Grau en Enginyeria Informàtica

L'ENGINYERIA DE L'APRENENTATGE AUTOMÀTIC

Un marc multidimensional per construir models robustos, eficients i ètics

UAB
Universitat Autònoma de Barcelona

Jordi Gonzalez Sabaté
Departament de Ciències de la Computació

Juny 2026


BY NC ND

A la memòria del meu pare.

A Jordi Pons, per haver-me ensenyat que coordinar un grau és també cuidar que l'aprenentatge sigui humà.

Noa, mai no et cansis de caminar...

Índex

Capítol 1

Geometria, Distància i Memòria

Benvinguts al llenguatge de la Intel·ligència Artificial. En aquest primer capítol, establirem els fonaments matemàtics que ens permetran traduir la complexitat del món real (imatges, sons, registres mèdics) en l'únic idioma que un ordinador pot processar: els vectors numèrics.

Aquest procés d'abstracció no és trivial; la manera com decidim representar les dades determinarà l'èxit o el fracàs de qualsevol sistema intel·ligent posterior. Veurem com, un cop hem convertit la realitat en punts dins d'un espai n-dimensional, conceptes intuïtius com "similitud" es tradueixen en operacions geomètriques com la "distància".

Al llarg del capítol, fusionarem conceptes d'àlgebra lineal amb algorismes de classificació basats en memòria, establint la base per a l'optimització i el raonament que desenvoluparem en la resta del llibre.

Guia d'activació docent

Repte inicial. Una empresa vol classificar fruita automàticament a partir de sensors de pes, diàmetre i color. Abans d'escollir cap algorisme, cal respondre una pregunta: *què vol dir que dues fruites s'assemblen per a una màquina?*

Checkpoints conceptuals.

1. Distingir entre objecte real, instància, característica, vector, etiqueta i matriu de disseny.
2. Justificar quan és raonable utilitzar distància euclidiana, Manhattan, Chebyshev o similitud del cosinus.

Errors típics provocats.

1. Calcular distàncies amb variables en escales incompatibles.
2. Interpretar una votació KNN com si fos una probabilitat calibrada.

Microcas professional. Sistema de triatge agrícola: cada peça de fruita es transforma en un vector numèric i el sistema decideix la classe a partir de la proximitat geomètrica amb exemples coneguts.

Connexió ètica o d'enginyeria. La representació de les dades no és neutral: si s'omet una característica rellevant, el model no pot aprendre-la; si s'inclouen variables proxy, pot reproduir biaixos socials.

1.1 De la Realitat al Vector

L'acte fonamental de l'aprenentatge automàtic i del raonament computacional és l'abstracció. Per a un ésser humà, una poma és un objecte tangible, amb olor, textura i història. Per a un algorisme, aquesta mateixa poma no existeix; només existeix una llista ordenada de nombres que la descriuen.

En aquesta secció, formalitzarem el procés de **vectorització**, mitjançant el qual transformem entitats del món real en punts matemàtics dins d'un espai vectorial. Aquesta transformació és

crítica: si la representació és deficient, cap algorisme d'optimització o de raonament podrà trobar una solució vàlida.

1.1.1 Fonaments Teòrics: L'Espai de Característiques

Definim un **Dataset** (conjunt de dades), denotat com $\mathcal{D}$, com una col·lecció de N exemples o instàncies. Cada instància representa un objecte d'estudi (una persona pacient, una imatge, un sensor).

Matemàticament, establim la següent **convenció notacional** per a tot el llibre:

- $\mathbf{x}^{(i)}$ (amb superíndex entre parèntesis) denota el vector de característiques de l'**exemple** i-èssim del dataset. $x_j^{(i)}$ (amb subíndex) denota el valor de la **característica** j-èssima de l'exemple i. Així, representem cada exemple com un parell ordenat $(\mathbf{x}^{(i)}, y^{(i)})$, on el vector viu en un espai de d-dimensions $\mathbb{R}^d$:

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_d^{(i)} \end{bmatrix} \tag{1.1}$$

  Noteu que utilitzem negreta minúscula ($\mathbf{x}$) per denotar vectors i que, per convenció en àlgebra lineal i en la majoria de biblioteques com `numpy`, els tractem com a vectors columna.

  Convenció matriu–vector (important): quan organitzem el dataset en una matriu $X \in \mathbb{R}^{N \times d}$, seguirem la convenció habitual en aprenentatge automàtic: **cada instància és una fila**. Així, la fila i és $(\mathbf{x}^{(i)})^\top$ i la columna j conté tots els valors de la característica j.

- **L'Espai de Característiques** (*Feature Space*, $\mathcal{X}$): És el conjunt de tots els possibles valors que pot prendre el vector $\mathbf{x}$. Geomètricament, aquest espai defineix els límits del nostre "univers" de representació.
  - Si $d = 2$, l'espai és un pla cartesià.
  - Si $d = 3$, l'espai és un volum tridimensional.
  - Si $d > 3$, parlem d'hiperespais, difícils de visualitzar però matemàticament consistents.
- **L'Etiqueta o Objectiu** (y): És la variable que volem predir o inferir.
  - En regressió, $y \in \mathbb{R}$, és a dir, un valor continu, com ara un preu o una temperatura.
  - En classificació amb C classes, podem codificar $y \in \{0, 1, \ldots, C-1\}$ o bé $y \in \{1, \ldots, C\}$, segons la convenció adoptada.

Nota per a l'Enginyeria: La Maledicció del Disseny

La tria de d (quines característiques mesurem) és una decisió de disseny, no matemàtica. Seleccionar característiques irrellevants introdueix soroll; ometre'n de rellevants fa el problema irresoluble. Un model mai pot aprendre allò que no està en les dades (*garbage in, garbage out*).

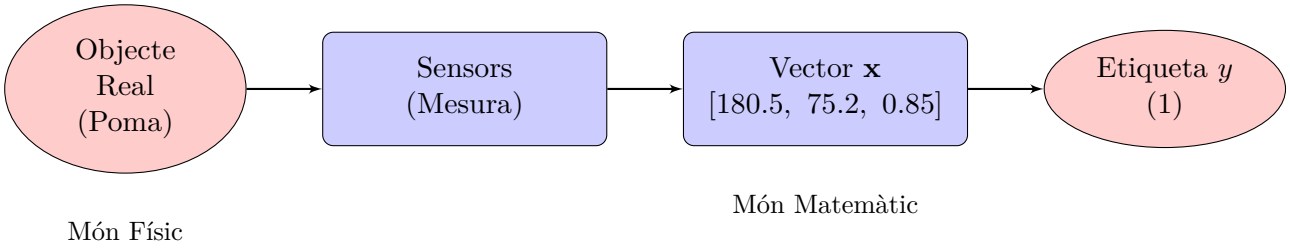


Figura 1.1: Procés d'abstracció i vectorització: de la realitat física a l'espai $\mathbb{R}^d$.

1.1.2 Implementació: De l'Objecte al `numpy.array`

En l'entorn professional, les dades rarament arriben com a vectors nets; sovint arriben com a registres de bases de dades o fitxers JSON. La tasca és construir la matriu de disseny $X \in \mathbb{R}^{N \times d}$. A continuació, mostrem com transformar dades "crues".

```python
import numpy as np

# 1. Dades "crues" (Raw Data) tal com arribarien d'un sensor
# Suposem: id, tipus, pes(g), diametre(mm), intensitat_vermella
raw_data = [
    {'id': 101, 'tipus': 'Poma', 'pes': 180.5, 'diam': 75.2, 'vermell': 0.85},
    {'id': 102, 'tipus': 'Taronja', 'pes': 160.2, 'diam': 78.1, 'vermell': 0.20},
    {'id': 103, 'tipus': 'Poma', 'pes': 192.0, 'diam': 80.5, 'vermell': 0.91}
]

# 2. Selecció de característiques (L'ID s'ignora per ser irrellevant)
features = ['pes', 'diam', 'vermell']

def vectoritzar(data, feats):
    X_list = []
    y_list = []
    for item in data:
        # Construcció del vector x
        X_list.append([item[f] for f in feats])
        # Construcció de l'etiqueta y
        y_list.append(item['tipus'])
    return np.array(X_list), np.array(y_list)

X, y = vectoritzar(raw_data, features)

print(f"Matriu X (Shape {X.shape}):\n{X}")
print(f"Vector x^(0) (Poma): {X[0]}")
# Output esperat per x^(0): [180.5  75.2   0.85]
```

Listing 1.1: Vectorització de dades crues amb Python

Així, una "Poma" es converteix en un vector $\mathbf{x} \in \mathbb{R}^d$, on d és la dimensió de l'espai:

$$\mathbf{x}_{poma} = \begin{bmatrix} 180.5 \\ 75.2 \\ 0.85 \end{bmatrix}$$

Formalment, definim el nostre **Dataset** $\mathcal{D}$ com una matriu $\mathbf{X} \in \mathbb{R}^{N \times d}$, on N és el nombre de mostres i d el nombre de característiques.

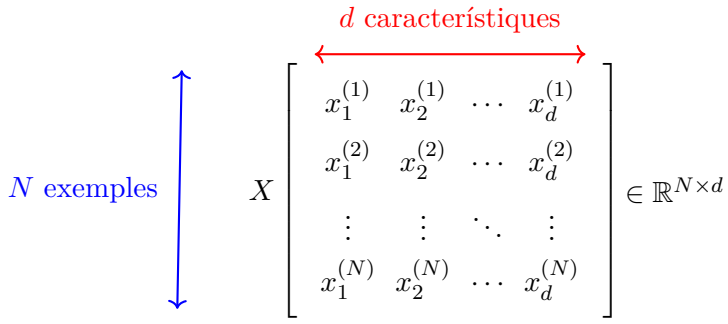


Figura 1.2: Estructura de la Matriu de Disseny X. Cada fila és un objecte del món real; cada columna, una dimensió de l'espai vectorial.

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & \dots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \dots & x_{N,d} \end{bmatrix} \tag{1.2}$$

La matriu resultant X és l'estructura central de qualsevol algorisme d'aprenentatge, i aquesta formalització és crucial perquè ens permetrà utilitzar operacions matricials (molt més ràpides que els bucles `for`) per processar milions de dades de cop.

Nota de notació: matemàtica vs. Python

Al llarg del llibre representarem cada instància $\mathbf{x}^{(i)}$ com un vector columna per facilitar l'àlgebra lineal. En canvi, en biblioteques com `numpy` o `scikit-learn`, la matriu `X` sol tenir forma `(n_samples, n_features)` i cada instància ocupa una fila. Així, `X[0]` és la primera mostra, encara que matemàticament l'escrivim com $(\mathbf{x}^{(0)})^\top$ dins de la matriu de disseny.

Consells de programació: Fes asserts abans de modelar

Abans d'entrenar cap model, comprova que les dades tenen la forma i els valors que creus:

```
assert X.ndim == 2
assert len(X) == len(y)
assert not np.isnan(X).any()
print(X.shape, y.shape)
print(np.unique(y, return_counts=True))
```

Molts errors greus en IA no venen d'una mala arquitectura, sinó d'una matriu mal construïda, etiquetes desalineades o valors perduts que ningú ha mirat.

1.1.3 Visualització: L'Espai 3D

Per entendre la geometria de les dades, és útil visualitzar on cauen aquests punts a l'espai. Si dues fruites són similars, els seus vectors $\mathbf{x}^{(i)}$ i $\mathbf{x}^{(j)}$ haurien d'estar "a prop" l'un de l'altre geomètricament.

```
import matplotlib.pyplot as plt

# Configuració de la figura 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
```

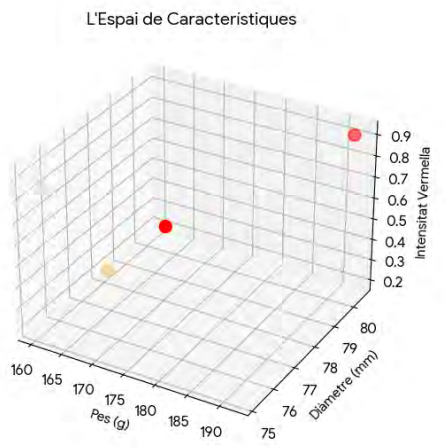


Figura 1.3: Visualització de l'output del codi 1.2. Observem com 'Pomes' (vermell) i 'Taronges' (taronja) ocupen regions diferents basant-se en les seves característiques.

```
6
7  # Mapeig de colors: Poma=Vermell, Taronja=Taronja
8  # Això ens permet veure visualment la separació de classes
9  colors = ['red' if lbl == 'Poma' else 'orange' for lbl in y]
10
11 # Scatter plot 3D
12 ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=colors, s=100)
13
14 ax.set_xlabel('Pes (g)')
15 ax.set_ylabel('Diàmetre (mm)')
16 ax.set_zlabel('Intensitat Vermella')
17 plt.title("L'Espai de Característiques")
18 plt.show()
```

Listing 1.2: Generació de l'espai 3D amb Matplotlib

Micro-Cas: Exemple de Representació

Una cooperativa agrària a Lleida vol automatitzar el triatge de fruita en la seva línia d'envasament. El procés manual és lent i subjecte a errors humans. L'objectiu és dissenyar un sistema que separi "Poma Golden", "Poma Fuji" i "Pera Conferència" automàticament.

La Solució Vectorial: L'equip decideix instal·lar tres sensors en la línia:

1. Una bàscula de pas ràpid (mesura la massa x_1 en grams).
2. Un arc de llum làser (mesura el diàmetre màxim x_2 en mm).
3. Un espectròmetre senzill (mesura la reflectància a 650nm, x_3, que correlaciona amb el color vermell).

Cada fruita que passa es converteix instantàniament en un vector $\mathbf{x} \in \mathbb{R}^3$:

- **Poma Fuji típica:** $\mathbf{x}_{Fuji} = [210, 85, 0.95]^T$.
- **Pera Conferència típica:** $\mathbf{x}_{Pera} = [190, 70, 0.15]^T$.

Aquesta abstracció permet aplicar algorismes de classificació estàndard (com els que veurem al Capítol 3) sense necessitat de reprogramar el sistema físic si canvia el tipus de fruita; només cal re-entrenar el model amb nous vectors. La màquina no "veu" fruita,

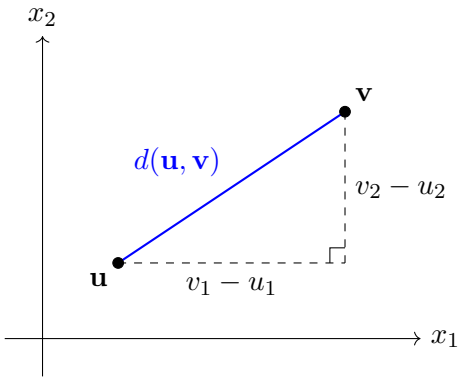


Figura 1.4: Visualització geomètrica de la distància Euclidiana 2D com a hipotenusa.

Exemple de Representació *(continuació)*

només processa coordenades en l'espai $\mathbb{R}^3$.

1.2 La Mètrica Euclidiana: El Teorema de Pitàgores Generalitzat

A la secció anterior hem après a traduir objectes reals en vectors $\mathbf{x} \in \mathbb{R}^d$. Però, de què ens serveix tenir les dades en aquest format? La resposta rau en la **geometria**.

Si volem que un ordinador sàpiga si una "Poma Fuji" és similar a una "Poma Royal Gala", necessitem quantificar la "similitud". En molts models geomètrics, fem servir la distància com una manera operativa de quantificar la similitud: com més a prop estan dos punts en l'espai de característiques, més semblants considerem els objectes que representen. Aquesta equivalència, però, depèn de la representació i de la mètrica escollida.

En aquesta secció, formalitzarem la **Distància Euclidiana**, que no és més que l'aplicació del Teorema de Pitàgores en múltiples dimensions.

1.2.1 Fonaments Teòrics: De la Hipotenusa a la Norma L_2

Recordem l'escola primària: en un triangle rectangle amb catets a i b, la longitud de la hipotenusa c és $c = \sqrt{a^2 + b^2}$.

Si tenim dos punts en un pla 2D, $\mathbf{v} = (v_1, v_2)$ i $\mathbf{u} = (u_1, u_2)$, la distància entre ells és la hipotenusa del triangle format per les diferències de les seves coordenades:

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2} \tag{1.3}$$

En Enginyeria de Dades, generalitzem aquest concepte a $\mathbb{R}^d$. La distància Euclidiana (o norma L_2) entre dos vectors $\mathbf{p}$ i $\mathbf{q}$ es defineix com:

$$d_2(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2} \tag{1.4}$$

Propietats d'una Mètrica Matemàtica

Perquè una funció $d(\mathbf{x}, \mathbf{y})$ sigui considerada una distància vàlida en enginyeria, ha de complir quatre axiomes estrictes:

1. **No-negativitat:** $d(\mathbf{x}, \mathbf{y}) \geq 0$. La distància mai pot ser negativa.
2. **Identitat:** $d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$. Només és zero si són el mateix punt.
3. **Simetria:** $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$. Anar d'A a B costa el mateix que de B a A.
4. **Desigualtat Triangular:** $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$. La línia recta sempre és el camí més curt.

La distància Euclidiana compleix totes quatre.

1.2.2 Visualització dels Contorns d'Equidistància

Visualment, la distància Euclidiana defineix "cercles" (o hiperesferes). Tots els punts que estan a una distància r d'un centre $\mathbf{c}$ compleixen l'equació del cercle.

A continuació, implementem una funció en Python per calcular aquesta distància i visualitzem els seus contorns. Matemàticament, la distància euclidiana és la norma L_2: $d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_i (u_i - v_i)^2}$. En Python, és preferible usar `np.linalg.norm` perquè és una rutina optimitzada i equivalent al càlcul manual.

```python
import numpy as np
import matplotlib.pyplot as plt

def distancia_euclidiana(u, v):
    """Distància Euclidiana (norma L2) entre dos vectors numpy."""
    # Opció A (pedagògica):
    # return np.sqrt(np.sum((u - v)**2))

    # Opció B (producció): rutina optimitzada
    return np.linalg.norm(u - v)

# Creem una malla de punts (grid)
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
centre = np.array([0, 0])

# Calculem la distància de cada punt al centre (0,0)
Z = np.sqrt(X**2 + Y**2)

# Visualització
plt.figure(figsize=(6, 6))
contorns = plt.contour(X, Y, Z, levels=[1, 2, 3, 4], colors=['r', 'g', 'b', 'k'])
plt.clabel(contorns, inline=True, fontsize=8)
plt.title("Contorns d'Equidistància Euclidiana (Cercles)")
plt.grid(True)
plt.show()
```

Listing 1.3: Càlcul de la distància Euclidiana i visualització

Fixeu-vos que els contorns són perfectament circulars. Això implica que la mètrica Euclidiana tracta totes les direccions de l'espai per igual (**isotropia**). Si ens moguem 1 unitat a la dreta o 1 unitat amunt, el "cost" de distància és el mateix.

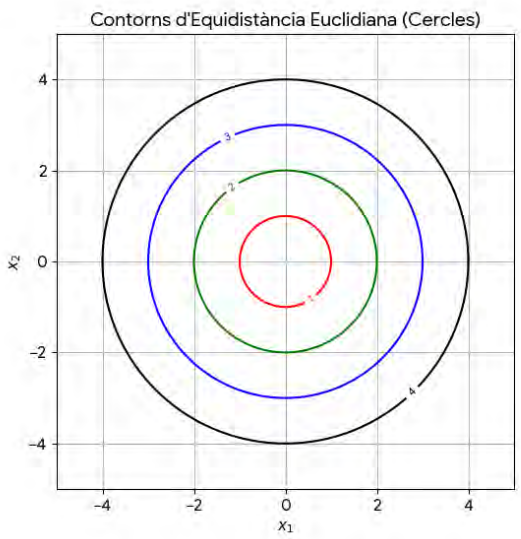


Figura 1.5: Visualització de l'output del codi 1.3. Cada línia de color (contorn) representa tots els punts que estan a la mateixa distància del centre (0,0). En la mètrica Euclidiana, l'equidistància forma cercles perfectes.

Micro-Cas: Seguretat en Braços Robòtics (Cobots)

La distància no serveix només per classificar fruites; és vital per a la seguretat en entorns físics: En una fàbrica, un braç robòtic col·laboratiu (*Cobot*) solda peces. Per normativa de seguretat, si un humà s'apropa a menys de 1.5 metres, el robot ha de reduir la velocitat; si s'apropa a menys de 0.5 metres, ha de fer una parada d'emergència.
Suposem que tenim sensors que ens donen la posició (x, y, z) en temps real:

- **Punta del Robot (r):** $\mathbf{r} = [2.5, 3.0, 1.2]^T$
- **Operari (h):** $\mathbf{h} = [3.1, 2.8, 1.0]^T$

L'enginyeria de control ha de calcular la distància cada 10 ms:

$$d(\mathbf{r}, \mathbf{h}) = \sqrt{(2.5 - 3.1)^2 + (3.0 - 2.8)^2 + (1.2 - 1.0)^2} \tag{1.5}$$

$$d(\mathbf{r}, \mathbf{h}) = \sqrt{(-0.6)^2 + (0.2)^2 + (0.2)^2} = \sqrt{0.36 + 0.04 + 0.04} = \sqrt{0.44} \approx 0.66 \text{ m} \tag{1.6}$$

Decisió de l'Algorisme: Com que $0.5 < 0.66 < 1.5$, el sistema activa el mode `REDUCCIÓ DE VELOCITAT`, però no atura la producció. Aquest càlcul simple s'executa milers de vegades per segon.

1.3 Més enllà de la línia recta: Manhattan i Minkowski

La distància Euclidiana assumeix un espai continu i lliure d'obstacles, on podem moure'ns en qualsevol direcció (la línia recta). Però, què passa si el nostre "agent" és un repartidor en una ciutat quadriculada com l'Eixample de Barcelona? O un paquet de dades viatjant per una xarxa de routers? En aquests casos, moure's en diagonal és impossible. Només podem moure'ns horitzontalment o verticalment. Aquí és on la geometria d'Euclides falla i necessitem noves mètriques.

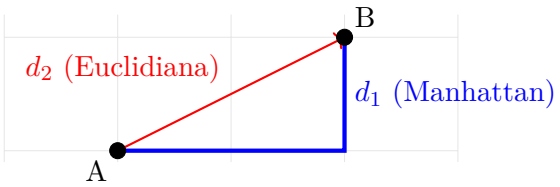


Figura 1.6: Comparativa entre la distància de vol d'ocell (d_2) i la urbana (d_1). En una ciutat, el camí vermell és il·legal. El camí blau és l'únic possible.

1.3.1 Fonaments Teòrics: La Geometria del Taxista (L_1)

La **Distància de Manhattan** (o norma L_1) es defineix com la suma de les diferències absolutes de les coordenades. Si volem anar del punt **u** al **v**, hem de sumar quants blocs ens movem en l'eix x i quants en l'eix y.

$$d_1(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_1 = \sum_{i=1}^{d} |u_i - v_i| \tag{1.7}$$

Matemàticament, aquesta distància acumula les diferències dimensió a dimensió de manera lineal. A diferència de la norma euclidiana, que eleva les diferències al quadrat abans de fer l'arrel i, per tant, és més sensible a desviacions grans en una coordenada, la norma Manhattan tracta cada contribució de manera additiva. Aquesta diferència canvia la geometria del problema i pot alterar completament quins punts considerem "propers".

De fet, les distàncies Euclidiana i Manhattan són només casos particulars d'una família de mètriques anomenades **Normes** L_p o **distància de Minkowski**. La distància general entre dos vectors **u** i **v** es defineix com:

$$D_p(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_p = \left(\sum_{j=1}^{d} |u_j - v_j|^p \right)^{\frac{1}{p}} \tag{1.8}$$

On $p \geq 1$ és un paràmetre que defineix la geometria de l'espai, i j itera sobre les dimensions:

- **Norma L_2 (Euclidiana, $p = 2$):** La distància "física" estàndard, l'única que és invariant a rotacions de l'espai (isotròpica):

$$\|\mathbf{u} - \mathbf{v}\|_2 = \sqrt{\sum_{j=1}^{d} (u_j - v_j)^2}$$

- **Norma L_1 (Manhattan, $p = 1$):** La distància "del taxista".

$$\|\mathbf{u} - \mathbf{v}\|_1 = \sum_{j=1}^{d} |u_j - v_j|$$

- Si $p \to \infty$: Distància de Chebyshev (L_∞), definida com $\max_i |p_i - q_i|$. (com en escacs, el rei fa el mateix nombre de passos en diagonal que en línia recta).

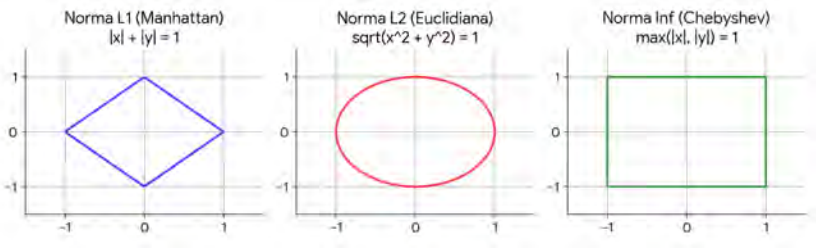


Figura 1.7: Visualització de les boles unitàries per a les normes L_1, L_2 i L_∞. La definició de "distància" canvia dràsticament la geometria.

Nota Tècnica: Normes en distància i en regularització

Les normes L_p apareixen en dos llocs diferents de l'aprenentatge automàtic. Aquí les utilitzem per definir distàncies entre exemples, és a dir, per decidir què vol dir que dos punts siguin propers. Més endavant, especialment en regularització, tornarem a trobar L_1 i L_2 com a penalitzacions sobre els pesos del model. La idea geomètrica és semblant, però el paper és diferent: en regularització controlarem la complexitat dels paràmetres.

1.3.2 Visualització: Les "Boles" Unitàries

Un "cercle" en aquestes geometries es defineix com el lloc geomètric dels punts que estan a distància 1 de l'origen.

- En L_2, $\sqrt{x^2 + y^2} = 1 \rightarrow$ Un cercle perfecte.
- En L_1, $|x| + |y| = 1 \rightarrow$ Un quadrat rotat 45 graus (un rombe).
- En L_∞, $\max(|x|, |y|) = 1 \rightarrow$ Un quadrat alineat amb els eixos.

Aquest concepte és visualment contraintuïtiu però matemàticament cert, en Python:

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-1.5, 1.5, 100)
y = np.linspace(-1.5, 1.5, 100)
X, Y = np.meshgrid(x, y)

# Definició de les normes
L1 = np.abs(X) + np.abs(Y)
L2 = np.sqrt(X**2 + Y**2)
Linf = np.maximum(np.abs(X), np.abs(Y))

plt.figure(figsize=(10, 3))

# Plot L1
plt.subplot(1, 3, 1)
plt.contour(X, Y, L1, levels=[1], colors='blue', linewidths=2)
plt.title("Norma L1 (Manhattan)\n|x| + |y| = 1")
plt.grid(True)

# Plot L2
plt.subplot(1, 3, 2)
plt.contour(X, Y, L2, levels=[1], colors='red', linewidths=2)
plt.title("Norma L2 (Euclidiana)\nsqrt(x^2 + y^2) = 1")
plt.grid(True)

```

```
# Plot L_inf
plt.subplot(1, 3, 3)
plt.contour(X, Y, Linf, levels=[1], colors='green', linewidths=2)
plt.title("Norma Inf (Chebyshev)\nmax(|x|, |y|) = 1")
plt.grid(True)

plt.tight_layout()
plt.show()
```

Listing 1.4: Visualització de les Normes Lp

Micro-Cas: Optimització de Rutes a l'Eixample

En enginyeria logística, calcular rutes òptimes utilitzant la distància equivocada pot portar a costos erronis: per a una empresa de repartiment, els drons poden volar en línia recta (L_2), però les furgonetes han de seguir els carrers (L_1).
Considerem dos punts de lliurament en coordenades de "illes de cases":

- **Magatzem:** $A = (0, 0)$
- **Punt de lliurament:** $B = (3, 4)$ (3 carrers a l'est, 4 al nord).

Càlcul per al Dron (L_2):

$$d_{dron} = \sqrt{(3-0)^2 + (4-0)^2} = \sqrt{9+16} = \sqrt{25} = 5 \text{ unitats.} \tag{1.9}$$

Càlcul per a la Furgoneta (L_1):

$$d_{furgo} = |3-0| + |4-0| = 3 + 4 = 7 \text{ unitats.} \tag{1.10}$$

Si el cost de combustible és de 1 € per unitat de distància:

- El dron costa 5 €.
- La furgoneta costa 7 €.

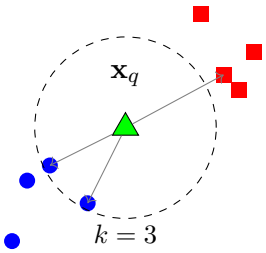
Utilitzar la fórmula Euclidiana per estimar el pressupost de la furgoneta portaria a un error del 40% en els costos ($\frac{7-5}{5}$). Per això, els sistemes de navegació no estimen rutes amb distància euclidiana pura: modelen la ciutat com un graf de carrers, on cada aresta té un pes associat al temps, la distància, el trànsit o les restriccions de circulació. En una graella urbana, la intuïció de Manhattan és una bona primera aproximació, però l'enginyeria real utilitza grafs ponderats.

Consells de programació: Abans de fer KNN, mira les unitats

El primer que fa una persona experta en enginyeria abans d'aplicar KNN no és entrenar el model: és mirar els rangs de cada columna. Si una variable va de 0 a 1 i una altra de 0 a 100.000, la segona dominarà la distància. Abans de calcular veïns, imprimeix mínims, màxims, mitjanes i desviacions, i decideix si cal escalar.

1.4 L'Algorisme K-Veïns Més Propers (k-NN)

Fins ara, hem après a situar objectes en l'espai i a mesurar la distància entre ells. Ara ens fem la pregunta fonamental de l'Aprenentatge Supervisat: *Si tinc un punt nou que no he vist mai, quina etiqueta li correspon?* La intuïció més bàsica ens diu: "Digues-me amb qui vas i et diré qui ets". En el llenguatge de l'enginyeria de dades, això es tradueix en el principi de contigüitat espacial: és molt probable que un punt tingui la mateixa etiqueta que els seus veïns més propers.



Decisió per $k = 3$:
2 Veïns Blaus
1 Veí Vermell
$\rightarrow$ Predicció: **Blau**

Figura 1.8: Mecanisme de votació per a $k = 3$. El punt verd es classifica com a Blau per majoria simple.

Aquest principi és la base de l'algorisme **k-Nearest Neighbors (k-NN)**, un dels mètodes més simples però sorprenentment eficaços en la indústria.

Exemple Numèric: Distància en $\mathbb{R}^2$

Considerem dues persones pacients representades per (Edat, Pes): $P_1 = (20, 60)$ i $P_2 = (25, 80)$. La distància euclidiana és:

$$d(P_1, P_2) = \sqrt{(25-20)^2 + (80-60)^2} = \sqrt{5^2 + 20^2} = \sqrt{425} \approx 20.61$$

Si afegim una tercera persona pacient $P_3 = (21, 62)$, veurem que $d(P_1, P_3) \approx 2.23$, confirmant que P_1 i P_3 són "veïns" més propers.

1.4.1 Fonaments Teòrics: Aprenentatge Mandrós

A diferència dels algorismes que veurem més endavant (com la Regressió o les Xarxes Neuronals), el k-NN és un algorisme d'**Aprenentatge Mandrós** (*Lazy Learning*).

- **Eager Learning (Ansiós):** L'algorisme processa les dades d'entrenament per construir un model matemàtic generalitzat (una fórmula). Després, pot descartar les dades originals.
- **Lazy Learning (Mandrós):** L'algorisme no ajusta una funció paramètrica durant la fase d'entrenament. En lloc d'això, emmagatzema les dades d'entrenament i deixa la feina principal per al moment de la predicció: calcular distàncies, trobar veïns i decidir per votació o mitjana. El càlcul real (la "feina") es posposa fins al moment exacte en què volem fer una predicció.

Formalització Matemàtica de la Votació

Sigui $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ el nostre conjunt de dades emmagatzemat. Donat un nou punt de consulta $\mathbf{x}_q$ (query point) i un hiperparàmetre k (nombre de veïns):

1. Calculem la distància $d(\mathbf{x}_q, \mathbf{x}^{(i)})$ per a tots els punts en $\mathcal{D}$ (utilitzant L_2 o L_1).
2. Seleccionem el subconjunt $\mathcal{N}_k$ format pels k punts amb menor distància a $\mathbf{x}_q$.
3. La predicció $\hat{y}_q$ es decideix per **votació majoritària**:

$$\hat{y}_q = \text{mode}\left(\{y^{(j)} \mid (\mathbf{x}^{(j)}, y^{(j)}) \in \mathcal{N}_k\}\right) \tag{1.11}$$

Matemàticament, podem expressar la votació utilitzant la funció indicatriu $\mathbb{I}$:

$$\hat{y}_q = \underset{c \in \text{Classes}}{\text{argmax}} \sum_{(\mathbf{x}^{(j)}, y^{(j)}) \in \mathcal{N}_k} \mathbb{I}(y^{(j)} = c) \tag{1.12}$$

Anàlisi del Cost Computacional

El principal inconvenient del k-NN no és la seva precisió, sinó la seva eficiència. És un mètode basat en memòria (*lazy learning*): no ajusta un conjunt petit de paràmetres, sinó que **guarda les dades** i difereix el "treball" cap al moment de consulta.

Cost d'entrenament (mínim): com a mínim, cal emmagatzemar el dataset (memòria $O(N \cdot d)$) i, sovint, fer preprocessament (normalització, imputació, selecció de característiques). A més, si construïm un índex (KD-Tree / Ball Tree), el cost de construcció típicament és $O(N \log N)$ (aprox.) amb memòria addicional $O(N)$.

Cost per consulta (brute force): per classificar **un sol** punt nou, cal calcular la distància a tots els N punts. Si cada punt té d dimensions, el cost per consulta (brute force) és $O(N \cdot d)$.

Índex exacte i alta dimensió: estructures com KD-Trees o Ball Trees poden accelerar consultes *quan d és moderat* (p. ex. baixa/mitjana dimensió), i en molts casos pràctics s'observa un comportament sublineal. Tanmateix, en alta dimensió la poda es degrada i el pitjor cas s'acosta a $O(N)$: la "maledicció de la dimensionalitat" pot fer que l'índex deixi de ser útil.

Pràctica en producció: sovint s'utilitzen mètodes de veïns *aproximats* (ANN), que canvien exactitud per temps de resposta, o bé es redueix la dimensió (PCA/embeddings) abans de fer veïns.

Per entendre-ho bé: KNN aprèn o només recorda?

KNN no aprèn una fórmula com $y = ax + b$. El que fa és guardar exemples passats i usar-los com a memòria. Quan arriba un cas nou, pregunta: "quins exemples antics s'hi assemblen més?". Per això és aprenentatge automàtic: la decisió depèn de les dades observades, encara que no hi hagi una fase d'entrenament amb pesos ajustats.

1.4.2 Les Fronteres de Voronoi

Una de les propietats més fascinants del k-NN és que genera fronteres de decisió complexes i no lineals. Si $k = 1$, la frontera de decisió coincideix exactament amb les cel·les del **Diagrama de Voronoi**. A mesura que augmentem k, la frontera es suavitza.

A continuació, implementem k-NN amb `scikit-learn` i visualitzem com "talla" l'espai.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.neighbors import KNeighborsClassifier

# 1. Generem dades sintètiques (2 classes)
X_train = np.array([[1, 2], [2, 1], [2, 3], [5, 4], [6, 5], [6, 3]]) # Coordenades
y_train = np.array([0, 0, 0, 1, 1, 1])  # Etiquetes (0: Blau, 1: Vermell)

# 2. Definim el model KNN amb k=1
k = 1
clf = KNeighborsClassifier(n_neighbors=k)
clf.fit(X_train, y_train)

```

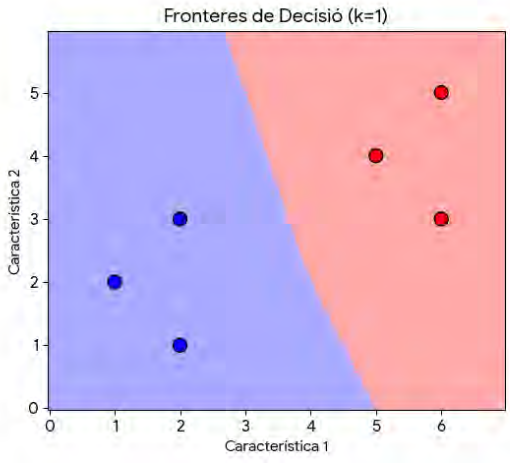


Figura 1.9: Visualització de l'output del codi. Amb $k = 1$, la frontera de decisió és irregular i s'adapta perfectament a cada punt d'entrenament. Les regions blava i vermella mostren com es classificaria qualsevol nou punt en aquell espai.

```
# 3. Creem una malla (Meshgrid) per pintar tot l'espai
h = 0.05  # Mida del pas de la malla
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

# 4. Predim per a CADA punt de la malla (força bruta visual)
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# 5. Plot
cmap_light = ListedColormap(['#AAAAFF', '#FFAAAA']) # Fons clar
cmap_bold = ListedColormap(['#0000FF', '#FF0000'])  # Punts forts

plt.figure(figsize=(6, 5))
plt.pcolormesh(xx, yy, Z, cmap=cmap_light, shading='auto')
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cmap_bold, edgecolor='k', s=100)
plt.title(f"Fronteres de Decisió (k={k})")
plt.xlabel("Característica 1")
plt.ylabel("Característica 2")
plt.show()
```

Listing 1.5: Visualització de fronteres de decisió KNN

El Dilema de k (Bias-Variance Tradeoff)

- **k petit (ex: $k = 1$):** El model és molt sensible al soroll. Si tenim un error en les dades (un punt vermell enmig dels blaus), el model crearà una "illa" incorrecta al seu voltant. Això és *Overfitting* (Sobreajust).
- **k gran (ex: $k = N$):** El model vota sempre per la classe majoritària global, ignorant l'estructura local. La frontera esdevé massa simple. Això és *Underfitting* (Subajust).

En enginyeria, k sol ser un nombre senar (per evitar empats en classificació binària) i es determina experimentalment.

1.4.3 Cas d'Aplicació Real: Diagnòstic Mèdic

El k-NN és àmpliament utilitzat en medicina per a sistemes de suport a la decisió clínica (CDSS), ja que és un algorisme **interpretable**: el personal mèdic pot preguntar "Per què dius que és maligne?" i el sistema respon "Perquè s'assembla molt a aquests 3 casos passats".

Micro-Cas: Classificació de Tumors de Mama

Considereu una base de dades històrica de biòpsies, on cada mostra s'ha processat per extreure'n característiques numèriques de les cèl·lules observades al microscopi.
Vectorització de la Persona Pacient: Considerem un espai simplificat 2D per a la visualització, tot i que realment s'usen unes 30 dimensions.

- x_1: **Textura mitjana** (variació estàndard dels valors d'escala de grisos).
- x_2: **Perímetre mitjà** del nucli cel·lular.

Arriba una nova persona pacient amb biòpsia: $\mathbf{x}_p = [18.4, 120.5]$.
Execució de l'Algorisme ($k = 5$): El sistema cerca a la base de dades (milions de registres) i troba els 5 vectors amb menor distància Euclidiana a $\mathbf{x}_p$:

- Veí 1 ($d = 0.1$): Maligne
- Veí 2 ($d = 0.3$): Maligne
- Veí 3 ($d = 0.4$): Benigne
- Veí 4 ($d = 0.6$): Maligne
- Veí 5 ($d = 0.8$): Benigne

Resultat de la Votació: 3 Malignes vs 2 Benignes. **Predicció:** Maligne (fracció de veïns malignes: $3/5 = 60\%$).
Nota important: aquesta fracció no és necessàriament una probabilitat *calibrada*. Si es vol interpretar el resultat com a probabilitat clínica, cal calibratge (p. ex. Platt/isotònic) o models probabilístics específics.
Aquest resultat alerta l'oncòleg perquè realitzi proves addicionals immediatament. En aquest cas, un fals negatiu seria fatal, per la qual cosa sovint s'ajusta el llindar de decisió (no cal el 50%; per exemple, amb un 30% de veïns malignes ja es pot disparar l'alarma).

1.5 Ponderació i Regressió KNN

Fins ara, l'algorisme k-NN que hem vist a la Secció 1.4 és profundament democràtic: el vot d'un veí que està "enganxat" al punt de consulta val exactament el mateix que el vot d'un veí que està al límit del radi k.

Des d'un punt de vista d'enginyeria, això és subòptim. La intuïció física ens diu que la influència d'un fenomen decau amb la distància (com la gravetat o el so). Per tant, hauríem de donar més pes a les dades més properes.

A més, què passa si y no és una etiqueta (Poma/Pera) sinó un valor continu (Preu, Temperatura)? En aquesta secció, estendrem el k-NN per abordar aquests reptes.

1.5.1 Fonaments Teòrics: La Funció de Ponderació

Introduïm un pes w_i associat a cada veí i. La forma més habitual de definir aquest pes és l'invers de la distància:

$$w_i = \frac{1}{d(\mathbf{x}_q, \mathbf{x}^{(i)}) + \epsilon} \tag{1.13}$$

On ϵ és una constant molt petita (ex: 10^{-5}) per evitar la divisió per zero.

Ara, la regla de decisió canvia:

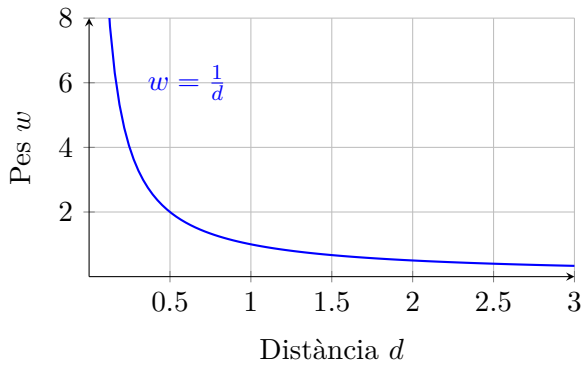


Figura 1.10: Relació entre distància i influència. Els punts molt propers tenen un pes que tendeix a infinit, dominant la decisió.

- **En Classificació (Vot Ponderat):** Sumem els pesos de cada classe, no els vots unitaris.

$$\hat{y}_q = \underset{c}{\text{argmax}} \sum_{(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{N}_k} w_i \cdot \mathbb{I}(y^{(i)} = c) \tag{1.14}$$

- **En Regressió (Mitjana Ponderada):** Si volem predir un valor real, calculem la mitjana dels valors dels veïns, ponderada per la seva proximitat.

$$\hat{y}_q = \frac{\sum_{i=1}^{k} w_i y^{(i)}}{\sum_{i=1}^{k} w_i} \tag{1.15}$$

Aquesta modificació fa que la contribució dels veïns depengui de manera gradual de la distància: els punts més propers tenen més influència que els punts situats al límit del veïnatge. Cal recordar, però, que el conjunt dels k veïns pot continuar canviant de manera discreta quan el punt de consulta es mou per l'espai. Per tant, el KNN ponderat sol reduir salts artificials en la predicció, però no converteix automàticament el model en una funció globalment suau.

Nota d'Implementació: Singularitat

Si $d(\mathbf{x}_q, \mathbf{x}^{(i)}) = 0$, el pes matemàtic tendeix a infinit ($w_i \to \infty$). A nivell de programació (p.ex. a `scikit-learn` amb `weights='distance'`), això es gestiona amb una excepció lògica: si la distància és zero, el model ignora la resta de veïns i retorna directament l'etiqueta $y^{(i)}$ del punt coincident. Això converteix el model en un interpolador exacte en aquell punt.

1.5.2 Implementació: KNN Uniforme vs Ponderat

Per visualitzar l'efecte de la ponderació, utilitzarem un problema de regressió simple (una ona sinusoidal amb soroll) i compararem com s'ajusten els dos mètodes.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor

# 1. Generar Dades Sintètiques (Sinus + Soroll)
np.random.seed(0)
X = np.sort(5 * np.random.rand(40, 1), axis=0) # 40 punts entre 0 i 5
T = np.linspace(0, 5, 500)[:, np.newaxis]      # Punts per pintar la línia
y = np.sin(X).ravel()
```

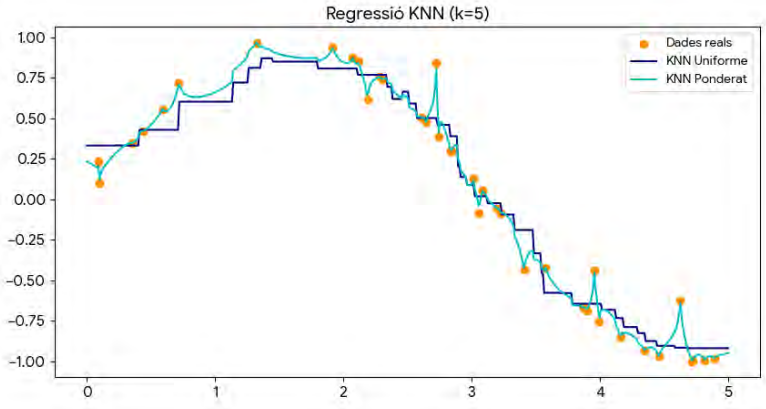


Figura 1.11: Comparativa entre KNN Uniforme i Ponderat. Mentre que l'uniforme produeix una funció esglaonada, el ponderat genera una interpolació suau que s'adapta millor a la tendència local, tot i que corre el risc d'overfitting.

```
y[::5] += 1 * (0.5 - np.random.rand(8))        # Afegir soroll cada 5 punts

# 2. Entrenar Models
n_neighbors = 5
knn_uniforme = KNeighborsRegressor(n_neighbors, weights='uniform')
knn_ponderat = KNeighborsRegressor(n_neighbors, weights='distance')

y_uniforme = knn_uniforme.fit(X, y).predict(T)
y_ponderat = knn_ponderat.fit(X, y).predict(T)

# 3. Visualització
plt.figure(figsize=(10, 5))
plt.scatter(X, y, color='darkorange', label='Dades reals')
plt.plot(T, y_uniforme, color='navy', label='KNN Uniforme')
plt.plot(T, y_ponderat, color='c', label='KNN Ponderat')
plt.axis('tight')
plt.legend()
plt.title(f"Regressió KNN (k={n_neighbors})")
plt.show()
```

Listing 1.6: Regressió KNN Uniforme vs Ponderada

En executar aquest codi, observareu que:

1. La línia **Uniforme** és "esglaonada" (step function), perquè la predicció es manté constant mentre el conjunt dels 5 veïns més propers no canviï.
2. La línia **Ponderada** és molt més "nerviosa" o flexible. S'ajusta millor als pics locals perquè els punts molt propers "estiren" la predicció cap a ells.

Nota d'Enginyeria: Interpolació

El KNN ponderat amb $w = 1/d$ té una propietat d'interpolació: si el punt de consulta x_q coincideix exactament amb un punt d'entrenament i no hi ha duplicats conflictius, la predicció queda dominada per aquell exemple. En dades netes, això pot portar a error d'entrenament gairebé nul; en dades sorolloses, també pot amplificar errors d'etiquetatge. Cal vigilar amb l'*Overfitting* si les dades tenen molt soroll.

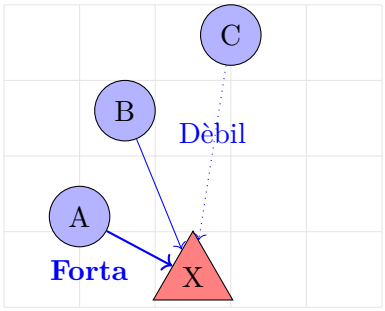


Figura 1.12: Mapa d'influència en taxació immobiliària: proximitat física implica similitud de preu.

Micro-Cas: Enginyeria Immobiliària per a la Predicció de Preus

La taxació immobiliària és el camp d'aplicació natural de la regressió KNN ponderada. Una *PropTech* (start-up immobiliària) vol estimar automàticament el preu de venda de nous immobles a Barcelona, que depèn fortament de la ubicació.

Dades: Tenim un històric de vendes recents:

- Pis A (C/ Balmes, a 10m): $4500\,€/m^2$
- Pis B (C/ Aribau, a 50m): $4200\,€/m^2$
- Pis C (Av. Diagonal, a 200m): $6000\,€/m^2$

Volem taxar un Pis X situat al C/ Balmes (just al costat del Pis A).

Càlcul amb KNN Uniforme ($k = 3$):

$$\hat{y}_{uni} = \frac{4500 + 4200 + 6000}{3} = 4900\,€/m^2 \tag{1.16}$$

Aquest valor està inflat pel Pis C, que és car però està lluny.

Càlcul amb KNN Ponderat ($k = 3$): Calculem els pesos (inversos):

- $w_A = 1/10 = 0.1$
- $w_B = 1/50 = 0.02$
- $w_C = 1/200 = 0.005$

Suma de pesos: $\sum w = 0.125$.

$$\hat{y}_{pond} = \frac{(0.1 \cdot 4500) + (0.02 \cdot 4200) + (0.005 \cdot 6000)}{0.125} \tag{1.17}$$

$$\hat{y}_{pond} = \frac{450 + 84 + 30}{0.125} = \frac{564}{0.125} = 4512\,€/m^2 \tag{1.18}$$

Conclusió: La predicció ponderada (4.512) és molt més propera al veí immediat (4.500) i ignora gairebé totalment el pis llunyà de la Diagonal.

1.6 La Maledicció de la Dimensionalitat

Fins ara, hem assumit que afegir més informació (més característiques d) sempre és bo. Intuïtivament, si volem classificar fruites, saber el "pes" ajuda, però saber el "pes", "diàmetre", "color", "textura" i "sucre" hauria de ser encara millor.

Aquesta intuïció és falsa en alta dimensió. Benvinguts a la **Maledicció de la Dimensionalitat** (*Curse of Dimensionality*), un fenomen matemàtic que pot degradar fortament els algorismes basats en distància (com k-NN) quan d creix, especialment si afegim moltes característiques

irrellevants o sorolloses..

Micro-Cas: Enginyeria Real: El Repte de Netflix (Sparsity)

Imagineu que voleu recomanar pel·lícules a persones usuàries.

- Vectors: Cada persona usuària és un vector $\mathbf{x}$ de dimensió $d = 15\,000$ (totes les pel·lícules del catàleg).
- Valors: $x_i = 5$ (li ha agradat), $x_i = 1$ (no li ha agradat), $x_i = 0$ (no l'ha vist). En aquest context, el valor 0 no s'ha d'interpretar com una puntuació negativa, sinó com una absència d'observació. Aquesta distinció és essencial: "no vist" no vol dir "no agradat".

El Problema: Una persona usuària "estàndard" només ha vist 100 pel·lícules. Això significa que el 99.3% del vector són zeros. Si calculem la distància Euclidiana "ingènua", els zeros dominaran el càlcul i totes les persones usuàries semblaran iguals (totes properes a l'origen).

Solució d'Enginyeria: No usem la distància Euclidiana. Usem la **Similitud del Cosinus** (que ignora la magnitud i els zeros comuns) o tècniques de *Matrix Factorization* per reduir la dimensió de 15 000 a un espai latent de 50 dimensions (Embeddings).
Per a vectors no nuls, la similitud del cosinus es defineix com:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \, \|\mathbf{y}\|_2}. \tag{1.19}$$

Una "distància cosinus" habitual és $d_{\cos}(\mathbf{x}, \mathbf{y}) = 1 - \cos(\mathbf{x}, \mathbf{y})$, però **no és una mètrica en general** (pot fallar la desigualtat triangular). En dades de recomanació, un **0** sovint vol dir **"no observat"** (missing), no pas "valor zero"; per això, a la pràctica es treballa amb dades esparses i/o només amb ítems co-observats, o amb embeddings.

Per entendre-ho bé: Per què més dimensions poden empitjorar?

Imagineu que busqueu veïns en una habitació petita: és fàcil trobar algú a prop. Ara imagineu que l'habitació creix en moltes direccions noves alhora. Amb el mateix nombre de persones, tothom queda lluny de tothom. En alta dimensió passa una cosa semblant: si no tenim moltes més dades, la paraula "proper" perd significat.

1.6.1 Fonaments Teòrics: L'Espai es Buida

Imaginem un hipercub unitari de costat $L = 1$ en d dimensions. Si volem omplir aquest espai amb una densitat constant de punts (e.g., 10 punts per eix), necessitem:

- $d = 1$: $10^1 = 10$ punts.
- $d = 2$: $10^2 = 100$ punts.
- $d = 3$: $10^3 = 1\,000$ punts.
- $d = 100$: 10^{100} punts (més que àtoms a l'univers).

Com que el nombre de dades N que tenim és finit i petit (ex: $N = 1\,000$), a mesura que d creix, la densitat de punts tendeix a zero. L'espai està pràcticament buit.

En dimensions altes, la nostra intuïció 2D o 3D falla. Encara que una esfera sembli "ocupar" molt espai dins d'un cub, a mesura que augmenta la dimensió gairebé tot el volum del cub queda lluny del centre, cap a les cantonades. Això fa que els punts aleatoris tendeixin a estar tots lluny els uns dels altres i que la noció de veí proper perdi força.

1.6.2 Simulació: La Concentració de la Mesura

Quan l'espai té moltes dimensions, totes les parelles de punts aleatoris tendeixen a estar a la mateixa distància mitjana. La variància de les distàncies tendeix a zero.

Si $\text{Dist}_{\max}$ és la distància al punt més llunyà i $\text{Dist}_{\min}$ al més proper:

$$\lim_{d \to \infty} \frac{\text{Dist}_{\max} - \text{Dist}_{\min}}{\text{Dist}_{\min}} \to 0 \tag{1.20}$$

Això destrueix el concepte útil de "veí més proper": si totes les distàncies són molt semblants, la decisió de k-NN deixa d'estar guiada per una proximitat realment informativa i esdevé molt sensible al soroll, als empats i als detalls de la implementació.

```python
import numpy as np
import matplotlib.pyplot as plt

# Paràmetres de la simulació
N = 1000                  # Nombre de punts
M = 20000                 # Nombre de parelles aleatòries (submostreig)
dims = [2, 10, 100, 1000] # Dimensions a provar

plt.figure(figsize=(8, 5))

# Assignació manual de colors per coherència amb el text
colors = {2: 'blue', 10: 'green', 100: 'orange', 1000: 'red'}

for d in dims:
    # Punts aleatoris en un hipercub unitari [0,1]^d
    X = np.random.rand(N, d)

    # Mostreig de M parelles (i != j) sense SciPy
    i = np.random.randint(0, N, size=M)
    j = (i + np.random.randint(1, N, size=M)) % N  # garanteix j != i

    # Distàncies euclidianes entre parelles aleatòries
    diff = X[i] - X[j]
    dists = np.linalg.norm(diff, axis=1)

    # Normalitzar per la mitjana per comparar les formes relatives
    dists_norm = dists / np.mean(dists)

    plt.hist(dists_norm, bins=50, density=True, histtype='step',
             linewidth=2, color=colors[d], label=f'Dim={d}')

plt.title("Concentració de la Mesura: Distribució de Distàncies")
plt.xlabel("Distància Normalitzada (d / mitjana)")
plt.ylabel("Freqüència")
plt.legend()
plt.show()
```

Listing 1.7: Simulació de la Maledicció de la Dimensionalitat (sense SciPy)

1.6.3 Cas d'Aplicació Real: Bioinformàtica i Genòmica

Aquest fenomen és el problema principal en biologia computacional, on el nombre de variables (d) supera enormement el nombre de persones pacients (N). Aquest escenari es coneix com el problema "$p \gg n$" (o $d \gg N$).

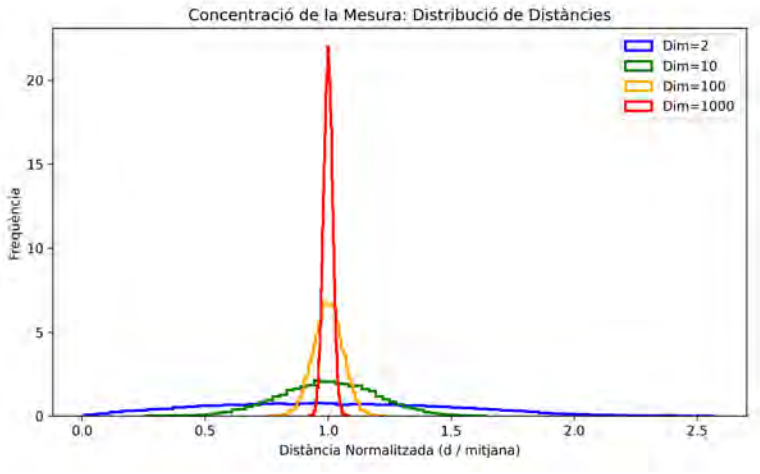


Figura 1.13: Simulació de la distribució de distàncies entre punts aleatoris. En $d = 2$ (blau), hi ha punts molt propers i molt llunyans. En $d = 1000$ (vermell), la distribució és un pic extremadament estret: tots els punts són equidistants.

Micro-Cas: Classificació d'Expressió Gènica (Microarrays)

Volem classificar persones pacients amb una malaltia basant-se en el seu ADN.
Les Dades:

- $N = 72$ persones pacients (mostres caríssimes d'obtenir).
- $d = 7\,129$ gens (nivells d'expressió mesurats per un xip d'ADN).

En un espai de 7 000 dimensions amb només 72 punts, si calculem la distància Euclidiana entre dues persones pacients, estem sumant les diferències de 7 129 gens.

- Potser només 50 gens estan relacionats amb la leucèmia (senyal).
- Els altres 7 079 gens són irrellevants per a la malaltia (soroll).

En sumar 7 000 termes de soroll, la petita diferència en els 50 gens importants queda totalment emmascarada. El k-NN pot perdre capacitat discriminativa: la distància queda dominada pel soroll i els veïns més propers deixen de ser necessàriament els més semblants des del punt de vista biològic.
Abans de calcular distàncies, és **obligatori** reduir la dimensió.

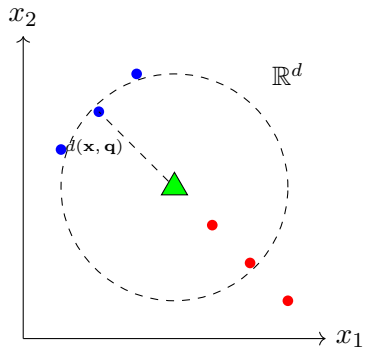
1. **Selecció de característiques:** reduir el conjunt de gens mitjançant criteris estadístics o supervisats, com ara tests diferencials, informació mútua o models amb regularització. Aquesta selecció s'ha de fer sempre dins del procés de validació per evitar *data leakage*.
2. **Projecció (PCA):** comprimir els milers de gens en un nombre reduït de components principals, preservant la màxima variància possible sense utilitzar directament les etiquetes.

Només en el subespai reduït, la distància torna a tenir sentit físic.

1.7 Conclusions i Debat

En aquest primer capítol hem transformat la realitat en vectors i hem après a prendre decisions basant-nos en la proximitat:

1. **Vectorització ($\mathbb{R}^d$):** Qualsevol objecte es pot representar com un punt en un espai d-dimensional. La qualitat del model depèn críticament de quines característiques escollim per definir l'espai.



Síntesi:
1. Espai $\mathbb{R}^d$
2. Distància d
3. Veïnatge k
4. Decisió $\hat{y}$

Figura 1.14: El paradigma del Capítol 1: Geometria, Distància i Veïnatge.

2. **Mètrica (L_p):** La noció de "similitud" no és única, depèn de la norma (L_1, L_2, L_∞) i una mètrica incorrecta pot invalidar tot el sistema.
3. **Aprenentatge Basat en Memòria (k-NN):** No sempre cal ajustar una fórmula paramètrica per prendre decisions útils. En alguns problemes, recordar el passat i mirar el veïnatge local permet construir fronteres no lineals de manera simple i interpretable.
4. **Maledicció de la Dimensionalitat:** La nostra intuïció 3D ens enganya. En altes dimensions, l'espai es buida i la distància Euclidiana perd sentit, obligant-nos a reduir la dimensió (PCA) o seleccionar característiques.

1.7.1 Debat Ètic: La Tirania de les Dades Històriques

L'algorisme k-NN és, per definició, una màquina conservadora: assumeix que el futur s'assemblarà al passat local. Aquesta hipòtesi és raonable en molts problemes físics, però pot ser perillosa en sistemes socials. Si les dades històriques incorporen discriminació, desigualtat o biaixos institucionals, el model no només pot reproduir-los, sinó presentar-los com una decisió aparentment objectiva. Aquesta és la lògica del *mathwashing*: convertir una regularitat històrica problemàtica en una decisió automatitzada amb aparença de neutralitat.

A continuació, analitzem tres fenòmens estructurals que sorgeixen de la geometria de les dades:

1. La Fal·làcia de la Ceguesa i les Variables Proxy

Sovint, qui s'inicia en l'enginyeria de dades cau en el parany de la "Ceguesa Voluntària" (*Fairness through Unawareness*): la creença que eliminar una columna protegida (ex: Sexe o Raça) de la matriu $\mathbf{X}$ elimina el biaix. Això és fals. En espais d'alta dimensió, les variables estan correlacionades. Altres característiques actuen com a **Proxies** (substituts redundants) de la variable eliminada.

Exemple: El Redlining Digital

En la concessió d'hipoteques, el **Codi Postal** actua sovint com a *proxy* de l'estatus socioeconòmic o l'ètnia a causa de la segregació urbana històrica. Un k-NN no "sap" res de races, però detecta que el veïnatge d'una zona concreta té històricament més impagaments. En denegar el crèdit per proximitat geogràfica, l'algorisme redibuixa les línies de segregació del passat ("Redlining") sense necessitat d'utilitzar la variable ètnica explícita.

2. Homofília i Cambres d'Eco (Filter Bubbles)

El principi del k-NN ("digues-me amb qui vas i et diré qui ets") és matemàticament idèntic al concepte sociològic d'**Homofília**: la tendència humana a associar-se amb els seus semblants. Quan apliquem k-NN a sistemes de recomanació (Spotify, YouTube, Netflix), l'algorisme mostra contingut "proper" al que ja agrada a la persona usuària.

- **Efecte a curt termini:** Es maximitza la satisfacció (tothom troba el que vol).
- **Efecte a llarg termini:** Es crea una **Cambra d'Eco**. La persona usuària queda atrapada en un veïnatge local de l'espai de característiques i mai és exposada a idees, música o productes "llunyans". A llarg termini, aquest mecanisme pot reduir la diversitat de recomanacions i contribuir a cambres d'eco si no s'introdueixen mecanismes de diversificació, exploració o serendipitat. El problema no és només matemàtic, sinó també de disseny del sistema de recomanació.

3. Bucles de Retroalimentació (Feedback Loops)

El perill més greu apareix quan l'algorisme altera la realitat que intenta predir. Això es coneix com a **Profecia Autocomplerta**.

Micro-Cas: Policia Predictiva

Un model prediu on enviar patrulles basant-se en arrestos passats.

1. El model envia la policia al Barri A (zona "calenta").
2. La presència policial augmenta la detecció de delictes menors al Barri A.
3. Aquestes noves dades s'afegeixen al dataset $\mathcal{D}$.
4. El model es reforça a si mateix: "Veieu? Teníem raó, al Barri A hi ha molt crim".

Mentrestant, el crim de "coll blanc" (fraus financers) en barris rics passa desapercebut perquè no hi ha dades històriques d'arrestos al carrer que atreguin el veïnatge.

El Dilema de l'Enginyeria: Exactitud vs. Justícia

Finalment, com a futurs i futures professionals, us trobareu amb un problema d'optimització difícil. Sovint, per corregir aquests biaixos, cal introduir restriccions que **redueixen la precisió** del model (ex: forçar que es concedeixin crèdits a barris diversos).

- **Opció A:** Maximitzar la precisió (i el benefici econòmic), acceptant la discriminació estructural.
- **Opció B:** Sacrificar precisió per garantir l'equitat estadística (*Algorithmic Fairness*).

Des de l'enginyeria no es pot dir simplement "són les dades". La decisió de quina funció optimitzar és responsabilitat de qui dissenya el sistema.

1.7.2 Preguntes per a l'Aula

1. Enginyeria: La Dictadura de les Unitats

Hem definit la distància com $d(\mathbf{u}, \mathbf{v}) = \sqrt{\sum(u_i - v_i)^2}$. Imagineu que dissenyeu un sistema de *Matchmaking* per a una app de cites amb dues característiques:

- x_1: Edat (rang 18-99). $\Delta x \approx 80$.
- x_2: Sou Anual (rang 20 000 – 100 000). $\Delta x \approx 80\,000$.

Pregunta: Si calculem la distància Euclidiana sense processar entre dos perfils A (25 anys, 30k€) i B (35 anys, 31k€), quina variable domina?

- *Anàlisi:* $d(A,B) = \sqrt{(25-35)^2 + (30000-31000)^2} = \sqrt{100 + 1.000.000} \approx 1000.05$.
- L'impacte de l'edat queda pràcticament diluït per la magnitud del sou: $10^2 = 100$ davant de $1000^2 = 1.000.000$.
- Per a l'algorisme, una diferència d'1 euro anual contribueix tant a la distància com una diferència d'1 any d'edat, perquè totes dues variables entren sense unitats ni escala comuna.
- Això és absurd funcionalment: el problema no és matemàtic, sinó de representació. Abans d'aplicar KNN, cal normalitzar o estandarditzar les variables.
- Conseqüència: sense escalat, el KNN pot acabar comparant persones gairebé només pel sou i ignorant variables conceptualment importants.

2. Ètica: El "Redlining" Digital i les Variables Proxy

Un banc entrena un k-NN per concedir hipoteques. Per llei, eliminen la variable "Ètnia" del vector de característiques per evitar la discriminació. No obstant això, mantenen el "Codi Postal".

- **El Fet:** En moltes ciutats, el codi postal està fortament correlacionat amb l'ètnia degut a la segregació històrica.
- **El Resultat:** L'algorisme troba que els veïns propers (mateix barri) tenen més impagaments i denega el crèdit.
- **Debat:** L'algorisme està sent racista? És ètic utilitzar una variable (Codi Postal) que actua com a "Proxy" d'una variable protegida? Com a professionals de l'enginyeria, tenim l'obligació d'eliminar variables que correlacionin amb la raça, encara que això baixi la precisió del model?

3. Seguretat: Enverinament de Dades (Data Poisoning)

A diferència d'altres models que "resumeixen" la informació, k-NN guarda les dades:

- **Escenari:** Un filtre de correu brossa (Spam) basat en k-NN decideix si un email és spam mirant emails similars que les persones usuàries han marcat manualment com a spam.
- **Atac:** Un atacant (spammer) es crea 1 000 comptes falsos i marca correus legítims (ex: factures de la llum) com a "Spam".
- **Conseqüència:** L'atacant ha introduït 1 000 punts "enverinats" a l'espai $\mathbb{R}^d$. Ara, quan arribi una factura real, els seus veïns més propers seran els punts enverinats i es classificarà erròniament.
- **Pregunta:** Com podem fer el sistema robust davant de persones usuàries que injecten soroll intencionat al dataset d'entrenament?

4. Societat: Les Cambres d'Eco (Filter Bubbles)

Sistemes com Spotify o Netflix usen variants de k-NN: "Si t'ha agradat la cançó A, et recomanaré les cançons veïnes B i C".

- **Efecte:** Això maximitza la satisfacció a curt termini (sempre t'agrada el que sents).

- **Debat:** A llarg termini, la persona usuària queda atrapada en una petita regió de l'espai de característiques (una "bombolla"). Mai descobrirà música radicalment diferent perquè està "lluny" en distància.
- **Disseny:** Hauríem de programar un cert grau d'aleatorietat o "serendipitat" (recomanar coses llunyanes expressament) per trencar aquestes cambres d'eco, encara que augmenti el risc que la persona usuària salti de cançó?

Tancament docent

Problemes de consolidació.

1. Donats dos vectors, calcular les distàncies L_1, L_2 i L_∞, i justificar quina mètrica seria més adequada en un context de repartiment urbà.
2. Aplicar KNN amb votació uniforme i ponderada, comparar el resultat i justificar quin model seria més robust davant soroll.

Criteris breus de resolució. La resolució hauria d'identificar correctament les variables, aplicar la fórmula de distància adequada, interpretar el paper de l'escala i justificar el resultat en termes del problema d'enginyeria plantejat.

Connexió amb pràctica/laboratori. Aquest capítol es desplega a la pràctica L1, on l'alumnat construeix un pipeline KNN, compara mètriques, prova diferents valors de k i comprova empíricament l'efecte de l'escalat sobre la frontera de decisió.

Capítol 2

Fonaments de l'Optimització: La Regressió

En el capítol anterior, hem resolt problemes complexos utilitzant una estratègia purament basada en la memòria. L'algorisme k-NN no intentava "entendre" les dades; simplement recordava el passat i assumia que el futur seria semblant al veïnatge històric. En aquest capítol, farem un salt conceptual crític en l'enginyeria de l'aprenentatge. Ens farem la pregunta: *Podem trobar una fórmula matemàtica compacta que expliqui el funcionament del món?*

L'objectiu docent d'aquest capítol és transformar el problema abstracte d'"aprendre" en un problema d'optimització matemàtica concret: **minimitzar l'error**. Veurem que, per a una màquina, aprendre no és res més que ajustar paràmetres (botons d'una màquina) fins que la diferència entre el que prediu i la realitat sigui mínima. Derivar una **funció de cost** serà la brúixola que permetrà a la màquina trobar la "millor" solució en un mar de possibilitats infinites, establint les bases de molts models posteriors del llibre: classificadors lineals, mètodes de marge, models regularitzats i tècniques modernes d'optimització.

Guia d'activació docent

Repte inicial. Un laboratori d'enginyeria civil vol predir la resistència a la compressió d'una mostra de formigó sense haver d'esperar 28 dies per trencar-la a la premsa. L'equip disposa de dades sobre ciment, aigua i edat de la mostra, però encara no sap quina fórmula ha d'utilitzar. La pregunta inicial del capítol és: *podem trobar una expressió matemàtica compacta que expliqui el comportament del món real i que generalitzi a mostres noves?*

Checkpoints conceptuals.

1. L'alumnat ha de poder distingir amb precisió entre model $h_\theta(x)$, paràmetres θ, predicció, residu, funció de cost $J(\theta)$ i procés d'optimització.
2. L'alumnat ha de poder explicar per què minimitzar l'error d'entrenament no és equivalent a construir un bon model, i ha de relacionar aquesta idea amb overfitting, underfitting i regularització.

Errors típics provocats.

1. Sumar directament els residus positius i negatius i concloure erròniament que el model no comet error perquè les desviacions es cancel·len.
2. Confondre un error d'entrenament baix amb una garantia de bon rendiment futur, sense validar generalització ni controlar la complexitat del model.

Microcas professional. Predicció de demanda o resistència: en molts sistemes d'enginyeria, el model no només ha d'ajustar dades passades, sinó evitar errors grans en situacions futures. Aquest cas permet introduir per què una funció de cost com l'MSE penalitza de manera especial les desviacions grans.

Connexió ètica o d'enginyeria. La funció de cost no és una tria neutra: defineix què considera "fracàs" el sistema. En problemes crítics, minimitzar l'error mitjà pot no ser suficient si alguns errors són molt més greus que altres. Dissenyar $J(\theta)$ és, per tant, una decisió d'enginyeria amb conseqüències tècniques, econòmiques i socials.

2.1 El Model Lineal: Una Hipòtesi del Món

Quan una persona experta en enginyeria s'enfronta a un conjunt de dades desconegut, el primer pas per modelar-lo és fer una suposició sobre la seva estructura subjacent, que anomenem **hipòtesi**. La hipòtesi més simple, però sorprenentment potent en molts camps de la física i l'economia, és assumir que la relació entre les variables és lineal. Això implica assumir que la sortida és proporcional a la suma ponderada de les entrades.

2.1.1 Definició Formal i Matemàtica

Definim la nostra hipòtesi $h_\theta(\mathbf{x})$ com una combinació lineal de les característiques d'entrada. Si el vector original té d característiques, $\mathbf{x} = [x_1, \dots, x_d]^\top$, el model s'expressa com:

$$h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d.$$

On:

- $\mathbf{x} = [x_1, \dots, x_d]^\top$ són les característiques de l'objecte.
- θ_0 és el biaix (*bias* o *intercept*).
- $\theta_1, \dots, \theta_d$ són els pesos (*weights*) associats a cada característica.

Per escriure el model de manera compacta, utilitzem el **truc del biaix**: definim un vector augmentat $\tilde{\mathbf{x}} = [1, x_1, \dots, x_d]^\top$ i un vector de paràmetres $\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_d]^\top$:

$$h_\theta(\mathbf{x}) = \boldsymbol{\theta}^\top \tilde{\mathbf{x}}.$$

Aquesta equació és la peça fonamental de l'àlgebra lineal aplicada a la IA. Quan el terme de biaix sigui rellevant, escriurem el vector augmentat $\tilde{\mathbf{x}}$. Quan el context sigui clar, mantindrem la notació $\mathbf{x}$ per al vector de característiques original.

Interpretació Geomètrica dels Pesos θ

Mentre que en k-NN la "memòria" eren els punts mateixos, en la regressió lineal la "memòria" està comprimida en el vector θ.

- Si θ_j és positiu i gran: La característica x_j fa pujar molt la predicció.
- Si θ_j és proper a 0: La característica x_j és irrellevant.
- Si θ_j és negatiu: Existeix una relació inversa (com més x_j, menys y).

Micro-Cas: Resistència del Formigó

Per aterrar aquest concepte abstracte, vegem com s'aplica en el control de qualitat de materials.
Un laboratori d'enginyeria civil vol predir la resistència a la compressió (y, en MPa) d'una mostra de formigó sense haver d'esperar 28 dies per trencar-la a la premsa.
L'equip defineix les característiques d'entrada $\mathbf{x}$:

1. x_1: Ciment (kg/m^3).
2. x_2: Aigua (kg/m^3).
3. x_3: Edat de la mostra (dies).

Després d'entrenar un model (procés que veurem a la Secció 2.4), obtenen el vector $\theta = [-15, 0.1, -0.5, 0.8]^T$. La fórmula resultant és:

$$\text{Resistència} \approx -15 + 0.1 \cdot (\text{Ciment}) - 0.5 \cdot (\text{Aigua}) + 0.8 \cdot (\text{Edat}) \tag{2.1}$$

Resistència del Formigó *(continuació)*

Interpretació d'Enginyeria:

- El signe negatiu de l'aigua ($\theta_2 = -0.5$) captura un fet físic conegut: massa aigua debilita la barreja.
- El pes del ciment ($\theta_1 = 0.1$) és positiu: més ciment, més força.
- El model ha "après" física bàsica simplement observant dades.

2.1.2 Visualització: La Línia i els Residus

Per entendre què significa que un model "s'ajusti" a les dades, hem de visualitzar l'error. En el cas d'una sola variable ($d = 1$), el model és una línia recta $y = mx + b$.

La distància vertical entre cada punt real (la dada) i la línia del model (la predicció) s'anomena **residu**. L'objectiu de la regressió no és tocar tots els punts (això seria impossible amb una recta), sinó aconseguir que la suma d'aquests residus sigui el més petita possible.

A continuació, generem una visualització d'aquests residus que intentarem minimitzar matemàticament.

```python
import numpy as np
import matplotlib.pyplot as plt

# 1. Generació de dades sintètiques (Model real + Soroll Gaussià)
np.random.seed(42)
X = 2 * np.random.rand(20, 1)  # Característica x1
# La realitat és y = 4 + 3x + soroll
y = 4 + 3 * X + np.random.randn(20, 1)

# 2. Model "hipotètic" (per exemple, trobat per mínims quadrats)
# y_pred = 4.2 + 2.8 * x (una aproximació)
theta_0, theta_1 = 4.2, 2.8
y_predict = theta_0 + theta_1 * X

# 3. Visualització
plt.figure(figsize=(8, 5))
plt.plot(X, y, "bo", label="Dades Reals (y)")
plt.plot(X, y_predict, "r-", linewidth=2, label="Model $h_\\theta(x)$")

# Dibuixar els RESIDUS (l'error que cometem)
for i in range(len(X)):
    plt.plot([X[i], X[i]], [y[i], y_predict[i]], "k--", alpha=0.3)

plt.text(0.1, 9.5, "Les línies discontínues\nsón els Residus (Errors)", fontsize=9)
plt.xlabel("$x_1$ (Input)", fontsize=12)
plt.ylabel("$y$ (Target)", rotation=0, fontsize=12)
plt.legend(loc="upper left")
plt.title("Concepte de Residu en Regressió")
plt.grid(True, alpha=0.3)
plt.show()
```

Listing 2.1: Visualització del Model Lineal i els Residus (Errors)

2.2 La Funció de Cost: Quantificant el Fracàs

A la secció anterior hem definit el nostre model (la "màquina") com una línia recta controlada pels paràmetres θ. Però, com sabem si uns valors concrets de θ són bons o dolents?

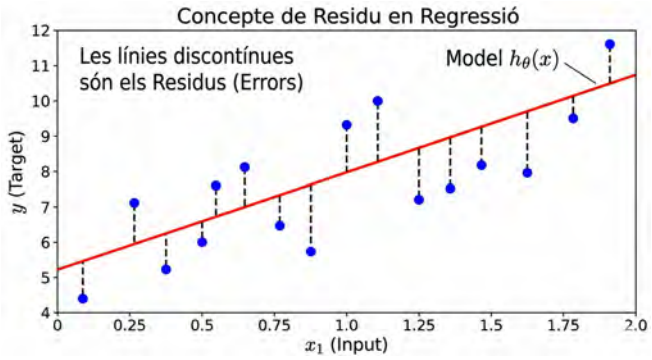


Figura 2.1: Visualització dels residus en una regressió lineal. Els punts blaus són les dades reals ($\mathbf{y}$), la línia vermella és la hipòtesi del model ($h_\theta(\mathbf{x})$) i les línies discontínues negres representen l'error individual de cada mostra. L'objectiu és minimitzar la suma dels quadrats d'aquestes línies verticals.

En enginyeria, no podem optimitzar allò que no podem mesurar. Per ensenyar a la màquina a trobar la millor línia, primer hem de definir matemàticament què significa "equivocar-se". Necessitem una funció que agafi el nostre model i les dades, i ens retorni un sol número que representi el **cost** de l'error comès.

L'objectiu de l'aprenentatge automàtic es redueix llavors a una sola tasca: trobar els paràmetres θ que facin que aquest número sigui el més petit possible.

2.2.1 Derivació de l'Error Quadràtic Mitjà (MSE)

Si recuperem la definició de residu per a un sol exemple i ($y^{(i)} - h_\theta(x^{(i)})$), podríem pensar a sumar tots els errors. El problema és que els errors positius i negatius es cancel·larien , donant la falsa il·lusió de precisió zero.

Per evitar això, elevem l'error al quadrat. Així neix la **Funció de Cost** (J), també coneguda com a MSE (*Mean Squared Error*) ja que per facilitar el càlcul del gradient posterior, definim la funció de cost minimitzant la meitat de l'error quadràtic mitjà:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2 \tag{2.2}$$

On m és el nombre total d'exemples d'entrenament, i el factor $\frac{1}{2}$ és una convenció matemàtica de disseny: quan derivem aquesta funció, l'exponent 2 baixarà i cancel·larà aquest denominador, deixant les equacions netes per a l'actualització de paràmetres.

Per què al quadrat i no el valor absolut?

Podríem utilitzar $|h(x) - y|$ (Error Absolut o L1). Tanmateix, el quadrat té dues propietats vitals per a l'enginyeria:

1. **Penalització no lineal:** un error de 10 unitats no pesa el doble que un error de 5, sinó quatre vegades més (100 vs. 25). Això fa que l'MSE sigui especialment sensible als errors grans, una propietat útil quan volem evitar desviacions extremes.
2. **Derivabilitat:** La funció x^2 és suau i diferenciable en tot el domini. La funció valor absolut $|x|$ té un "pic" no derivable en 0, la qual cosa complica els algorismes d'optimització.

2.2.2 La Superfície d'Error: El "Bol" Convex

Si visualitzem la funció de cost $J(\theta)$ en funció dels paràmetres θ_0 i θ_1, obtenim una superfície en 3D. Com que el model és lineal i l'error és quadràtic, aquesta superfície sempre té forma de **paraboloide** (un bol de cereals), és a dir, qie si deixem caure una pilota virtual en qualsevol punt d'aquesta superfície, sempre rodolarà cap a una solució. Aquesta propietat és fonamental: en una funció convexa no hi ha mínims locals espuris; qualsevol mínim local és també un mínim global. Ara bé, la unicitat del mínim requereix condicions addicionals, com ara que la matriu de disseny tingui rang complet. Si hi ha col·linealitat entre característiques, poden existir diverses solucions equivalents o una vall plana de mínims.

A continuació, generem aquesta superfície per veure on "viu" l'error mínim.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 1. Dades (mateix exemple simple que abans)
np.random.seed(42)
m = 20
X = 2 * np.random.rand(m, 1)
y = 4 + 3 * X + np.random.randn(m, 1)  # y = 4 + 3x + soroll

# 2. Preparar la malla de paràmetres theta
theta0_vals = np.linspace(0, 8, 100)
theta1_vals = np.linspace(0, 6, 100)
T0, T1 = np.meshgrid(theta0_vals, theta1_vals)

# 3. Calcular J(theta) per a cada combinació
J_vals = np.zeros(T0.shape)
for i in range(T0.shape[0]):
    for j in range(T0.shape[1]):
        h_x = T0[i, j] + T1[i, j] * X
        J_vals[i, j] = (1/(2*m)) * np.sum((h_x - y)**2)

# 4. Mínim empíric aproximat sobre la malla
idx_min = np.unravel_index(np.argmin(J_vals), J_vals.shape)
theta0_min = T0[idx_min]
theta1_min = T1[idx_min]
J_min = J_vals[idx_min]

# 5. Plot 3D
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(T0, T1, J_vals, cmap='viridis', alpha=0.8)

ax.scatter(theta0_min, theta1_min, J_min, c='red', s=50,
           label='Mínim empíric aproximat')

ax.set_xlabel('$\\theta_0$ (Biaix)')
ax.set_ylabel('$\\theta_1$ (Pes)')
ax.set_zlabel('Cost $J(\\theta)$')
plt.title('Superfície de Cost Convexa')
plt.legend()
plt.show()
```

Listing 2.2: Visualització de la Superfície de Cost (3D)

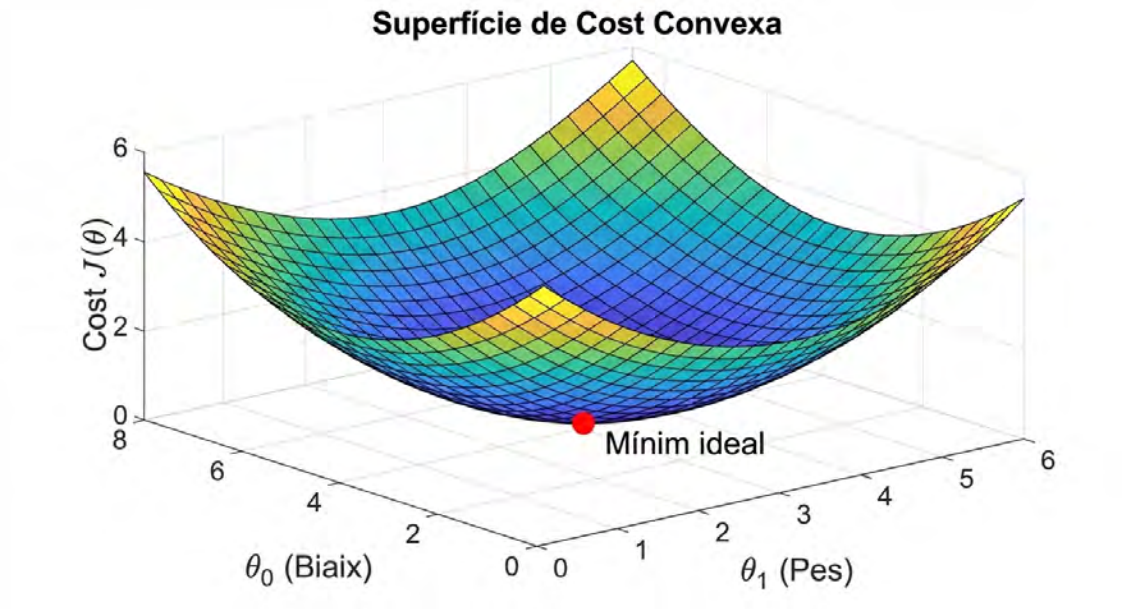


Figura 2.2: Superfície de la funció de cost $J(\theta)$ per a una regressió lineal simple. La superfície és convexa; el punt vermell indica el mínim empíric aproximat sobre la malla de paràmetres. L'altura representa la magnitud de l'error per a cada combinació de θ_0 i θ_1.

Micro-Cas: Xarxa Elèctrica i la Demanda Prevista

En el sector elèctric, la precisió de la funció de cost es tradueix directament en milions d'euros i estabilitat de la xarxa. L'operador d'un sistema elèctric ha de predir la demanda de consum per a l'endemà per planificar quantes centrals encendre.
Siguin les dades:

- y: Demanda real (MW).
- $h_\theta(x)$: Demanda prevista pel model.

L'ús de l'Error Quadràtic (MSE) aquí té una implicació física crítica.

- Si l'error és petit (± 10 MW), la xarxa ho absorbeix amb la inèrcia dels generadors. El cost és negligible.
- Si l'error és gran (± 1000 MW), cal arrencar turbines de gas d'emergència (molt cares) o patir apagades.

El quadrat de la fórmula $J(\theta)$ reflecteix perfectament aquesta realitat: **els errors petits són tolerables, els errors grans són desproporcionadament catastròfics**. Per tant, en minimitzar l'MSE, l'algorisme prioritza evitar les desviacions extremes que posarien en perill el subministrament.

2.3 Cerca Local vs. Optimització Contínua

Si heu cursat assignatures d'Intel·ligència Artificial clàssica, segurament recordareu algorismes com el *Hill Climbing* (Escalada de la Muntanya). En aquests problemes, un agent es mou per un tauler d'escacs o un graf de ciutats, intentant millorar la seva situació pas a pas. En l'Aprenentatge Computacional, estem fent exactament el mateix, però amb un canvi fonamental en la naturalesa del terreny: hem canviat el **món discret** (enters, caselles, nodes) pel **món continu** (reals, corbes, superfícies).

2.3.1 Del Veïnatge al Gradient Infinitesimal

En l'optimització discreta, l'espai d'estats és finit o numerable. Si estem en un estat s (una solució parcial), definim el seu **veïnatge** $N(s)$ com el conjunt d'estats als quals podem accedir amb un sol moviment.

- **Hill Climbing:** Mirem tots els veïns $s' \in N(s)$. Si trobem un s' millor que s, "saltem" cap a ell.
- **Limitació:** El concepte de "direcció" no existeix realment; només existeixen opcions finites (Nord, Sud, Est, Oest).

En canvi, en la Regressió (i en Deep Learning), els nostres paràmetres θ són nombres reals ($\mathbb{R}^n$). No podem "mirar tots els veïns" perquè al voltant d'un punt hi ha infinits punts. Aquí és on entra la màgia del càlcul diferencial.

La **derivada** (o el gradient ∇J en múltiples dimensions) no ens diu quin veí és millor, sinó que ens dona la direcció de màxim pendent.

$$\begin{aligned} \text{Discret (cerca):} \quad & s_{t+1} \leftarrow \arg\min_{s' \in N(s_t)} \text{Cost}(s'), \\ \text{Continu (gradient):} \quad & \boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \nabla J(\boldsymbol{\theta}_t). \end{aligned} \tag{2.3}$$

Mentre que l'agent discret va fent "salts" bruscos, l'algorisme d'aprenentatge es comporta com una bola que roda suaument muntanya avall. Si en un problema discret parlem de "qualitat" en lloc de "cost", el criteri equivalent seria maximitzar la qualitat. Per visualitzar aquesta diferència, compararem a continuació un algorisme de cerca amb el descens del gradient.

```python
import numpy as np
import matplotlib.pyplot as plt

# Funció objectiu (Superfície simple): z = x^2 + y^2
def cost_func(x, y):
    return x**2 + y**2

# --- SUBPLOT 1: MÓN DISCRET (Hill Climbing) ---
x_disc = np.linspace(-4, 4, 9) # Només enters: -4, -3, ..., 4
y_disc = np.linspace(-4, 4, 9)
X_d, Y_d = np.meshgrid(x_disc, y_disc)
Z_d = cost_func(X_d, Y_d)

# Camí discret (Salts)
path_x_d = [3, 2, 2, 1, 0]
path_y_d = [3, 3, 2, 1, 0]

# --- SUBPLOT 2: MÓN CONTINU (Gradient Descent) ---
x_cont = np.linspace(-4, 4, 100)
y_cont = np.linspace(-4, 4, 100)
X_c, Y_c = np.meshgrid(x_cont, y_cont)
Z_c = cost_func(X_c, Y_c)

# Camí continu (Suau - Simulat)
angles = np.linspace(0, 4, 50)
path_x_c = 3 * np.exp(-angles)  # Decaïment exponencial
path_y_c = 3 * np.exp(-angles)

# --- PLOT ---
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

```

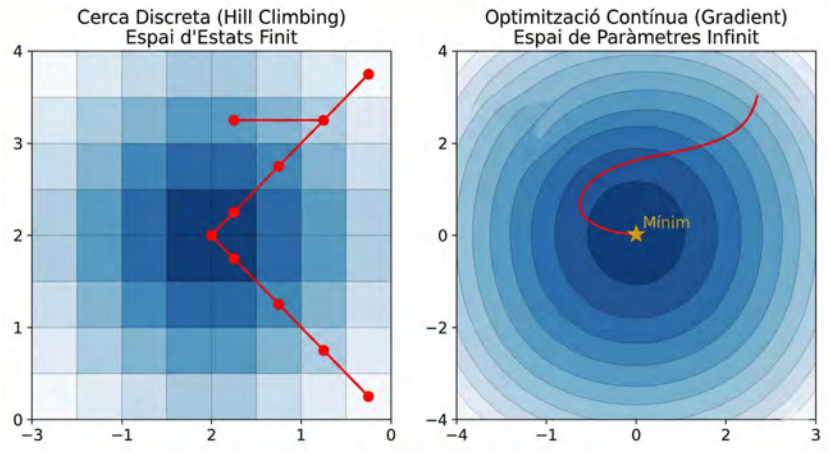


Figura 2.3: Comparativa de paradigmes. A l'esquerra, un agent discret salta entre estats definits (caselles). A la dreta, l'optimització contínua permet moviments infinitesimals i suaus, aprofitant la informació de la derivada.

```
# 1. Discret
ax[0].set_title("Cerca Discreta (Hill Climbing)\nEspai d'Estats Finit")
ax[0].pcolormesh(X_d, Y_d, Z_d, shading='auto', cmap='Blues', edgecolors='k', linewidths=1)
ax[0].plot(path_x_d, path_y_d, 'ro-', linewidth=2, label='Trajectòria')
ax[0].scatter(path_x_d, path_y_d, c='red', s=100)
ax[0].grid(True)

# 2. Continu
ax[1].set_title("Optimització Contínua (Gradient)\nEspai de Paràmetres Infinit")
ax[1].contourf(X_c, Y_c, Z_c, 20, cmap='Blues')
ax[1].plot(path_x_c, path_y_c, 'r-', linewidth=3, label='Descens suau')
ax[1].scatter(0, 0, c='gold', marker='*', s=200, label='Mínim')

plt.tight_layout()
plt.show()
```

Listing 2.3: Comparativa: Hill Climbing (Discret) vs Gradient Descent (Continu)

2.3.2 Cas d'Ús Real: Logística de Transport

Aquesta distinció teòrica és vital en la indústria logística, on sovint conviuen els dos tipus de problemes en el mateix projecte.

Micro-Cas: Rutes vs. eficiència en logística

Una empresa logística resol dos tipus de problemes. Decidir l'ordre en què un camió visita 50 adreces és un problema discret i combinatori, proper al *Traveling Salesman Problem*. En canvi, decidir la velocitat òptima en cada tram per minimitzar el consum és un problema continu, on les variables poden prendre valors reals i els mètodes basats en gradient són naturals. No tots els problemes d'optimització es resolen amb la mateixa família d'algorismes.

2.4 El Descens del Gradient: Matemàtiques del Moviment

Fins ara tenim un objectiu (minimitzar $J(\theta)$) i una intuïció (lliscar pel pendent). Però un ordinador no entén de "pendents"; només entén de nombres i operacions. Com traduïm "baixar muntanya avall" a codi? La resposta és el **Gradient**.

Matemàticament, el gradient $\nabla J(\theta)$ és un vector que conté totes les derivades parcials de la funció de cost. Aquest vector té una propietat geomètrica fonamental: sempre apunta a la direcció de **màxim ascens**. Per tant, per trobar el mínim, la nostra estratègia serà senzillament moure'ns en la direcció *oposada* al gradient.

2.4.1 L'Algorisme d'Actualització

L'algorisme del Descens del Gradient (*Gradient Descent*) és iteratiu. Comencem amb uns paràmetres θ aleatoris i repetim el següent bucle fins a la convergència:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2.4}$$

On:

- $:=$ és l'operador d'assignació (actualitzem el valor antic).
- α (Alpha) és la **Taxa d'Aprenentatge** (*Learning Rate*). Aquest hiperparàmetre controla la mida del pas. Si és massa petit, trigarem segles a arribar; si és massa gran, ens passarem de llarg.
- $\frac{\partial J}{\partial \theta_j}$ és la derivada parcial respecte al paràmetre j. Ens diu com canvia l'error si movem lleugerament aquest paràmetre.

Per entendre-ho bé: El gradient és una fletxa cap amunt

El gradient no apunta cap al mínim: apunta cap a on la funció puja més ràpid. Per baixar, hem de caminar en la direcció contrària. Per això el descens del gradient fa:

$$\theta_{\text{nou}} = \theta_{\text{vell}} - \alpha \nabla J(\theta).$$

El signe menys és la clau: no seguim el pendent, el desfem.

Nota d'implementació: actualització simultània

En el descens del gradient, totes les components de $\boldsymbol{\theta}$ s'han d'actualitzar de manera simultània. A la pràctica, això vol dir calcular primer tot el gradient $\nabla J(\boldsymbol{\theta})$ amb els valors antics dels paràmetres i, només després, substituir $\boldsymbol{\theta}$ pel nou vector actualitzat.

2.4.2 Derivació Pas a Pas (El "Secret" del 1/2)

Per implementar això, necessitem l'expressió concreta de la derivada. Recuperem la nostra MSE definida a la Secció 2.2:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2$$

Apliquem la Regla de la Cadena per derivar respecte a θ_j:

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)})^2 \tag{2.5}$$

$$= \frac{1}{2m} \sum_{i=1}^{m} 2 \cdot (h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) \tag{2.6}$$

$$= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \tag{2.7}$$

On hem utilitzat que, en el model lineal, $\frac{\partial}{\partial \theta_j} h_\theta(\mathbf{x}^{(i)}) = x_j^{(i)}$. Sovint s'inclou un terme $x_0^{(i)} = 1$ per modelar l'intercept (θ_0). Noteu com el 2 de l'exponent s'ha cancel·lat amb el $\frac{1}{2}$ del denominador, deixant una fórmula neta i elegant. El terme final ens diu que la derivada és proporcional a l'**error** $(h - y)$ multiplicat per l'**entrada** x_j.

Interpretació Física de la Derivada

- Si l'error és gran, pot contribuir fortament al gradient, especialment si la característica associada també té un valor gran.
- Si l'error és proper a zero, la contribució d'aquella mostra al gradient sol ser petita.
- La mida real del pas no depèn només del gradient, sinó també de la taxa d'aprenentatge α.

Consells de programació: Comprova el gradient amb diferències finites

Abans de confiar en un gradient implementat a mà, prova'l en un cas petit. Aproxima la derivada numèricament:

$$\frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$

i compara-la amb el gradient analític. Si no coincideixen en un exemple simple, no entrenis res encara. Els bugs de gradient poden fer perdre dies sencers.

2.4.3 El Perill d'Alpha i l'Oscil·lació

Un dels reptes més grans en enginyeria és triar la taxa d'aprenentatge α correcta.

- α **òptim:** Convergència ràpida i suau.
- α **massa gran:** El gradient ens impulsa tan fort que saltem per sobre de la vall i pugem per l'altra banda. Al següent pas, tornem a saltar enrere encara més fort. Això crea una trajectòria en **ziga-zaga** que divergeix cap a l'infinit.

A continuació, simulem aquest fenomen.

```
import numpy as np
import matplotlib.pyplot as plt

def J(theta):       # Definim una funció de cost el·líptica (vall estreta)
    return theta[0]**2 + 10 * theta[1]**2

def gradient(theta):            # Derivada parcial: [2*theta0, 20*theta1]
    return np.array([2*theta[0], 20*theta[1]])

def run_gradient_descent(alpha, steps=10):
    theta = np.array([-9.0, -2.0]) # Punt inicial
    path = [theta.copy()]
```

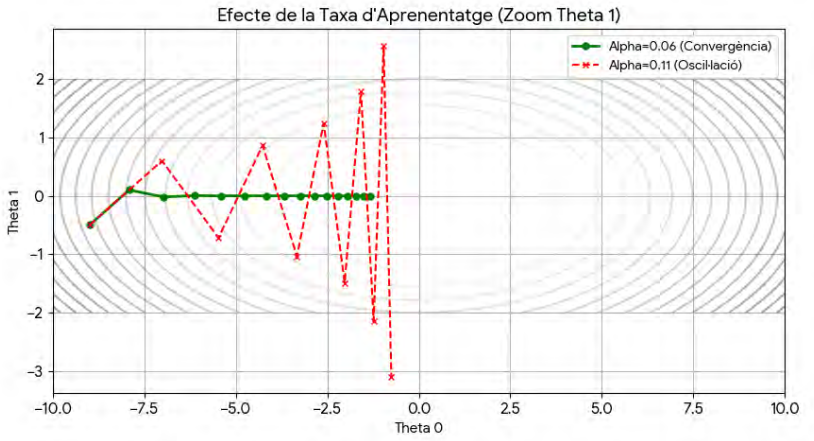


Figura 2.4: Visualització de l'actualització de paràmetres. La línia verda baixa suaument cap al centre (mínim). La línia vermella, a causa d'un α excessiu, rebota a les parets de la vall (oscil·lació), trigant més a convergir o fins i tot divergint.

```
    for _ in range(steps):
        grad = gradient(theta)
        theta = theta - alpha * grad
        path.append(theta.copy())
    return np.array(path)

path_good = run_gradient_descent(0.08)    # Escenari 1: Alpha Bo

path_bad = run_gradient_descent(0.19)       # Escenari 2: Alpha Dolent

x_vals = np.linspace(-10, 10, 100)          # Plot
y_vals = np.linspace(-4, 4, 100)
X, Y = np.meshgrid(x_vals, y_vals)
Z = X**2 + 10 * Y**2

plt.figure(figsize=(10, 6))
plt.contour(X, Y, Z, levels=20, cmap='gray_r', alpha=0.4)

plt.plot(path_good[:,0], path_good[:,1], 'g-o', label='Alpha=0.08 (Convergència)', linewidth
    =2)     # Trajectòria Bona

plt.plot(path_bad[:,0], path_bad[:,1], 'r--x', label='Alpha=0.19 (Oscil·lació)', linewidth
    =1.5)       # Trajectòria Dolenta (Zig-Zag)

plt.title("Efecte de la Taxa d'Aprenentatge (Alpha)")
plt.xlabel("Theta 0")
plt.ylabel("Theta 1")
plt.legend()
plt.grid(True)
plt.show()
```

Listing 2.4: Simulació de Trajectòries del Gradient (Convergència vs. Oscil·lació)

Micro-Cas: Beamforming Adaptatiu en Antenes 5G

El gradient és la base de l'optimització de senyals en temps real: Una antena 5G moderna no emet en totes direccions; enfoca un "raig" (*beam*) directament cap al telèfon de la

Beamforming Adaptatiu en Antenes 5G *(continuació)*

persona usuària per maximitzar la cobertura i minimitzar la interferència.
El sistema té un array d'antenes, cadascuna amb un paràmetre de fase θ_j.

- **Objectiu (J):** Minimitzar l'error de recepció (Soroll - Senyal).
- **Gradient:** El sistema mesura com varia la qualitat del senyal si mou lleugerament la fase.
- **Dinàmica:** L'algorisme ajusta θ en temps real milers de vegades per segon. Si la persona usuària es mou ràpid (va en cotxe), el sistema necessita un α alt per seguir-lo ("aprendre" la nova posició ràpid), però si α és massa alt, el senyal es tornarà inestable i la trucada es tallarà (oscil·lació).

2.5 Regressió Polinòmica i Generalització

Fins ara, el nostre model $h_\theta(x)$ ha estat una línia recta. Però el món rarament és lineal: la trajectòria d'un projectil és una paràbola, el creixement bacterià és exponencial i els cicles econòmics són oscil·latoris.

Com podem modelar aquestes corbes si només tenim una eina per fer rectes? La resposta de l'enginyeria és brillant per la seva simplicitat: **no canviem el model, canviem les dades**.

2.5.1 Expansió de la Base

Si tenim una sola característica x (p. ex: temps), podem crear característiques sintètiques elevant x a diferents potències. Transformem el nostre vector d'entrada original en un nou vector expandit $\phi(x)$:

$$x \to \phi(x) = [1, x, x^2, x^3, \dots, x^p]^T \tag{2.8}$$

Ara, el nostre model lineal "veu" aquestes potències com si fossin variables independents noves $(x_1, x_2, \dots)$. L'equació esdevé:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_p x^p \tag{2.9}$$

El grau del polinomi, que denotarem per p, és un hiperparàmetre crític.

- **Underfitting (p massa petit):** el model és massa simple per capturar l'estructura de les dades. Té un biaix alt.
- **Bon compromís (p adequat):** el model captura la tendència principal sense perseguir massa el soroll.
- **Overfitting (p massa gran):** el model és tan flexible que pot passar gairebé per tots els punts d'entrenament, memoritzant fins i tot el soroll. Té una variància alta i pot fallar amb dades noves.

El Truc de la Linealitat

Encara que la funció resultant sigui no lineal respecte a la variable original x, continua sent lineal respecte als paràmetres $\boldsymbol{\theta}$. Això és crucial: podem continuar utilitzant el mateix marc d'optimització de la regressió lineal. Amb MSE, el problema continua sent convex en els paràmetres, tot i que pot estar mal condicionat si el grau del polinomi és alt o si les noves característiques són molt correlacionades. La regularització serà una eina clau per controlar aquest problema.

Consells de programació: Escala abans de crear polinomis

Si x té valors grans, x^2, x^3 o x^{10} poden créixer molt ràpid i fer que el model sigui numèricament inestable. En codi professional, les característiques polinòmiques s'han de construir dins d'un `Pipeline` i combinar-se amb escalat i regularització. No facis polinomis sobre dades crues sense mirar-ne l'escala.

2.5.2 Visualització: La recerca de l'equilibri

A continuació, ajustem tres models a les mateixes dades.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

np.random.seed(0)    # 1. Generar Dades Reals (Funció Cosinus + Soroll)
n_samples = 30
X = np.sort(np.random.rand(n_samples))
y = np.cos(1.5 * np.pi * X) + np.random.randn(n_samples) * 0.1

# 2. Definir graus a provar
degrees = [1, 4, 15]
titles = ["Grau 1 (Underfitting)", "Grau 4 (Correcte)", "Grau 15 (Overfitting)"]

plt.figure(figsize=(14, 4))
for i, degree in enumerate(degrees):
    ax = plt.subplot(1, 3, i + 1)

    # Pipeline: Transformar dades -> Ajustar Regressió Lineal
    polynomial_features = PolynomialFeatures(degree=degree, include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("poly", polynomial_features), ("linear", linear_regression)])

    pipeline.fit(X[:, np.newaxis], y)         # Entrenar
    X_test = np.linspace(0, 1, 100)          # Avaluar (per pintar la línia suau)
    y_test = pipeline.predict(X_test[:, np.newaxis])

    # Visualització
    plt.plot(X_test, y_test, label="Model", color='red', linewidth=2)
    plt.scatter(X, y, edgecolor='b', s=20, label="Dades")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.title(titles[i])
    plt.ylim(-1.5, 1.5)
    plt.legend(loc="best")

plt.tight_layout()
plt.show()
```

Listing 2.5: Underfitting vs. Overfitting amb Regressió Polinòmica

Micro-Cas: Modelatge de Corbes d'Infecció

L'elecció incorrecta del grau del polinomi pot portar a decisions polítiques desastroses: durant l'inici d'una pandèmia (com la COVID-19), les dades dels primers dies semblen una corba exponencial ($y = e^x$).

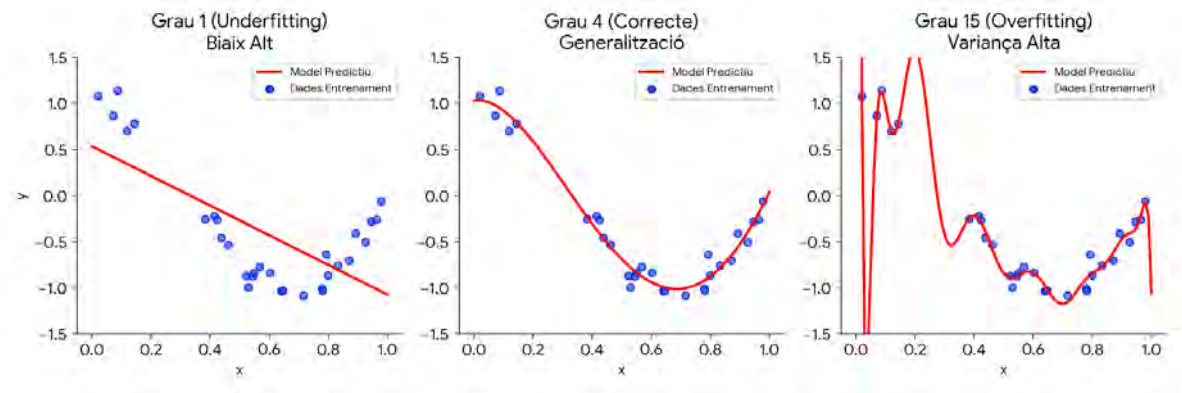


Figura 2.5: L'efecte del grau del polinomi. A l'esquerra, el model no pot corbar-se. Al centre, captura l'ona. A la dreta, el model oscil·la, perdent capacitat predictiva.

Modelatge de Corbes d'Infecció *(continuació)*

- Si s'ajusta un polinomi de grau molt alt a les dades de les dues primeres setmanes, es pot obtenir un ajust aparentment excel·lent sobre l'entrenament, fins i tot amb un R^2 molt proper a 1.
- No obstant això, els polinomis d'alt grau tendeixen a disparar-se cap a $\pm\infty$ just fora del rang de dades.
- **Conseqüència:** El model podria predir que la setmana següent hi haurà 0 casos (si la cua del polinomi baixa) o milions de casos (si puja), quan la realitat biològica segueix una corba logística (Sigmoide) que se satura.

En ciència, un model que "explica" perfectament el passat però no té base física és inútil per al futur.

2.6 Regularització: La Geometria de la Restricció

A la secció anterior hem vist que un model massa complex (polinomi de grau 15) es "torça" violentament per adaptar-se a cada punt de soroll. Matemàticament, això es tradueix en el fet que els pesos θ agafen valors enormement grans (ex: $\theta_3 = 1.000.000$). Per evitar-ho, podríem reduir manualment el grau del polinomi, però això és perillós (podríem perdre informació útil).

La solució elegant és mantenir totes les característiques sense que els pesos creixin massa. Això s'aconsegueix modificant la nostra funció de cost. Ja no volem només minimitzar l'error; volem minimitzar l'error *i també* la magnitud dels paràmetres.

$$J(\theta)_{reg} = \underbrace{J(\theta)_{MSE}}_{\text{Ajustar Dades}} + \lambda \cdot \underbrace{R(\theta)}_{\text{Penalització}} \tag{2.10}$$

On λ (Lambda) és l'hiperparàmetre de regularització:

- Si $\lambda = 0$: Tornem a la regressió normal (risc d'Overfitting).
- Si $\lambda \to \infty$: forcem els pesos associats a les característiques cap a zero. Si el biaix no es penalitza, el model tendeix a una predicció gairebé constant, amb risc d'*underfitting*.

Per entendre-ho bé: Regularitzar és posar una corretja al model

Un model molt flexible pot córrer darrere de cada punt de soroll. La regularització li posa una corretja: encara pot ajustar les dades, però no pot fer créixer els pesos sense límit. El resultat és un model que potser encaixa una mica pitjor l'entrenament, però sol comportar-se millor amb dades noves.

Nota d'implementació: penalitzem els pesos, no el biaix

En moltes implementacions, el terme de regularització s'aplica als pesos $\theta_1, \ldots, \theta_d$, però no al biaix θ_0. La raó és que el biaix només desplaça el model i no controla directament la complexitat de la relació entre les característiques i la predicció.

2.6.1 Ridge (L_2) vs. Lasso (L_1)

Depenent de quina "norma" (vegeu Capítol 1) utilitzem per mesurar la mida de θ, obtenim dos comportaments molt diferents:

Regressió Ridge (Norma L_2): La penalització és la suma dels quadrats dels pesos: $R(\theta) = \sum \theta_j^2$.

- **Efecte Geomètric:** La restricció té forma de **cercle**.
- **Resultat:** Redueix tots els pesos proporcionalment cap a zero, però rarament els fa exactament zero. És ideal quan totes les característiques aporten una mica d'informació.

Regressió Lasso (Norma L_1): La penalització és la suma dels valors absoluts: $R(\theta) = \sum |\theta_j|$.

- **Efecte Geomètric:** La restricció té forma de **diamant** (rombe).
- **Resultat:** Té la propietat de fer que molts pesos siguin **exactament zero**. Actua com un selector de característiques automàtic.

2.6.2 Per què Lasso anul·la variables?

La millor manera d'entendre això és visualitzar els contorns de la funció de cost MSE (el·lipses) xocant contra la regió de restricció (cercle o diamant). La solució òptima és el primer punt on es toquen.

```
import numpy as np
import matplotlib.pyplot as plt

def plot_contours(ax, norm_type):
    t0 = np.linspace(-3, 3, 100)         # Malla de punts theta
    t1 = np.linspace(-3, 3, 100)
    T0, T1 = np.meshgrid(t0, t1)

    # 1. Funció de Cost MSE (El·lipses centrades a [1.5, 1.5])
    Z_mse = (T0 - 1.5)**2 + 2*(T1 - 1.5)**2
    ax.contour(T0, T1, Z_mse, levels=[0.5, 1, 2, 4, 8], colors='blue', alpha=0.6)
    ax.text(1.5, 1.5, "Mínim MSE\n(Sense Reg)", color='blue', ha='center', fontsize=9)

    # 2. Restricció (Penalty Region)
    if norm_type == 'L1':
        # Diamant (|x| + |y| <= 1)
        Z_reg = np.abs(T0) + np.abs(T1)
        ax.contour(T0, T1, Z_reg, levels=[1], colors='red', linewidths=2)
        ax.fill_between(t0, 1-np.abs(t0), -1+np.abs(t0), color='red', alpha=0.1)
        title = "Lasso (L1): Restricció Diamant"
        # Solució òptima (Intersecció a l'eix)
```

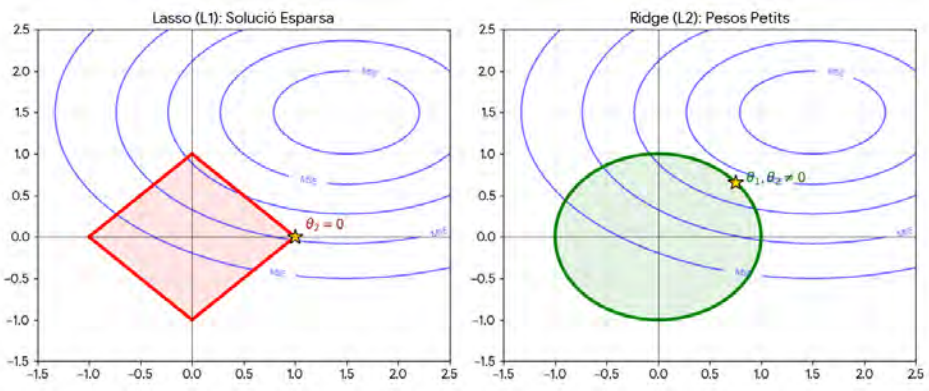


Figura 2.6: Interpretació geomètrica de la regularització. Les línies blaves representen contorns de l'error i l'àrea ombrejada el pressupost permès per als pesos. En Lasso (L_1), les cantonades del diamant fan més probable que la solució òptima caigui sobre un eix, anul·lant algun pes. En Ridge (L_2), la restricció circular tendeix a reduir els pesos de manera més distribuïda.

```
        opt_x, opt_y = 1.0, 0.0

    else: # L2: Cercle (x^2 + y^2 <= 1)
        Z_reg = T0**2 + T1**2
        ax.contour(T0, T1, Z_reg, levels=[1], colors='green', linewidths=2)
        circle = plt.Circle((0, 0), 1, color='green', alpha=0.1)
        ax.add_artist(circle)
        title = "Ridge (L2): Restricció Cercle"
        # Solució òptima (Intersecció fora dels eixos)
        opt_x, opt_y = 0.6, 0.8 # Aprox

    # Marcar l'òptim regularitzat
    ax.scatter(opt_x, opt_y, c='gold', s=150, edgecolors='k', marker='*', zorder=10, label='Òptim Regularitzat')

    ax.set_xlim(-2, 3)
    ax.set_ylim(-2, 3)
    ax.axhline(0, color='black', linewidth=0.5)
    ax.axvline(0, color='black', linewidth=0.5)
    ax.set_title(title)
    if norm_type == 'L1': ax.legend(loc='lower left')

# Plot
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
plot_contours(ax[0], 'L1')
plot_contours(ax[1], 'L2')
plt.tight_layout()
plt.show()
```

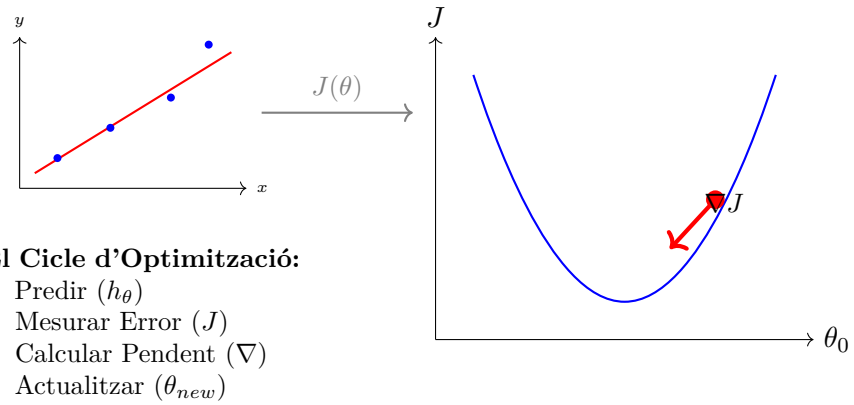
Listing 2.6: Geometria de la Regularització: L1 vs L2

Micro-Cas: Biocarcadors del Càncer

Aquí és on l'elecció entre L1 i L2 determina l'èxit del projecte: volem predir el risc de càncer (y) basant-nos en l'expressió genètica ($\mathbf{x}$).

- Tenim $d = 20.000$ gens (dimensions).
- Tenim $m = 100$ persones pacients (dades).

Estem en un cas extrem de $d \gg m$.



El Cicle d'Optimització:
1. Predir (h_θ)
2. Mesurar Error (J)
3. Calcular Pendent (∇)
4. Actualitzar (θ_{new})

Figura 2.7: Del món de les Dades (esquerra) al món dels Paràmetres (dreta).

Biocarcadors del Càncer *(continuació)*

- Si usem **Ridge (L2)**, el model pot repartir pesos petits entre molts gens. Això pot millorar l'estabilitat predictiva, però dificulta obtenir una llista curta de candidats interpretables.
- Si usem **Lasso (L1)**, afavorim una solució esparsa: molts pesos poden quedar exactament a zero i només un subconjunt reduït de gens manté pes no nul. Això pot ajudar a identificar *biomarcadors candidats*, és a dir, variables associades amb la predicció. Ara bé, aquesta selecció no demostra causalitat: caldrien estudis biològics i validació externa per afirmar que aquests gens causen la malaltia.

2.7 Conclusions i Debat

En aquest segon capítol hem deixat de "memoritzar" dades per "aprendre" l'estructura que les genera. Hem transformat l'aprenentatge en un problema d'optimització:

1. **El Model Paramètric (h_θ):** Hem passat de memoritzar exemples a definir una família de funcions controlada per paràmetres. Aprendre ja no és guardar dades, sinó trobar valors de $\boldsymbol{\theta}$ que produeixin bones prediccions.
2. **La Funció de Cost (J):** Hem definit matemàticament els errors amb l'MSE, que **penalitza fortament** els errors grans i fa el problema **derivable** (i convex en el cas lineal). Ara bé, l'MSE no és robust a outliers; per robustesa s'utilitzen MAE o Huber.
3. **El Gradient (∇):** Hem descobert el motor de la IA moderna. La derivada ens indica la direcció de millora.
4. **Generalització i Regularització:** Hem après que un model perfecte en l'entrenament (Overfitting) sol ser un model inútil en la realitat. La geometria de Ridge (L_2) i Lasso (L_1) ens permet controlar la complexitat.

2.7.1 Debat Ètic: La Caixa Negra i la Tirania de l'Objectiu

Si al Capítol 1 el problema era el biaix heretat de les dades, en aquest capítol apareix un altre risc: optimitzar molt bé un objectiu que no representa completament el problema real. Un model lineal simple pot ser transparent, però a mesura que augmentem la complexitat —polinomis d'alt grau, moltes variables, regularització o models posteriors més flexibles— pot créixer la distància entre rendiment numèric i comprensió humana. El repte no és només encertar, sinó poder justificar què s'ha optimitzat i amb quines conseqüències.

1. El Dret a l'Explicació vs. La Precisió

El Reglament General de Protecció de Dades (GDPR) europeu suggereix un "dret a l'explicació" per a decisions automatitzades. Si un banc utilitza una regressió polinòmica de grau 20 (o una xarxa neuronal) per denegar un crèdit, i la persona clienta pregunta "Per què?", la resposta matemàtica seria: *"Perquè la combinació del teu sou elevat a la potència 14 menys la teva edat al cub multiplicada pel coeficient 0.034 supera el llindar"*. Això és matemàticament cert, però humanament inútil.

Dilema: Acceptaríem una decisió mèdica sense explicació?

Imagineu un sistema de diagnòstic amb molt bon rendiment estadístic, però que no pot explicar quins factors han influït en una recomanació clínica rellevant. Com a professionals de l'enginyeria, podem valorar l'eficiència global; com a persones pacients, necessitem una justificació comprensible. El repte és decidir quan el rendiment compensa la pèrdua d'explicabilitat i quan no.

2. La Correlació no és Causalitat (Spurious Correlations)

L'optimització matemàtica és cega al sentit comú. L'algorisme només busca minimitzar l'MSE. Si troba que "consum de gelats" prediu "atacs de taurons" (perquè tots dos pugen a l'estiu), utilitzarà aquesta variable sense dubtar. En entorns controlats, això és una anècdota. En polítiques públiques, és un desastre. Un algorisme podria concloure que "enviar més policia a un barri" (input) redueix el crim (output), quan potser la variable latent real era la inversió en enllumenat públic o escoles. El gradient descendent no distingeix entre causa i coincidència.

3. La Tirania de la Funció de Cost

La decisió d'enginyeria més important no és triar α (learning rate), sinó dissenyar $J(\theta)$. La màquina farà **exactament** allò que li demanem, amb obediència cega.

Micro-Cas: Recomanadors i objectius mal especificats

Imagineu un sistema de recomanació optimitzat principalment per maximitzar el temps de visualització. Si certs continguts extrems, sensacionalistes o polaritzants retenen més l'atenció, el sistema pot aprendre a recomanar-los amb més freqüència.
El problema no és que el model "vulgui" polaritzar ningú, sinó que optimitza amb molta eficiència l'objectiu que li hem definit. Aquesta és la tirania de la funció de cost.

2.7.2 Preguntes per a l'Aula

1. Enginyeria: El Món no és un Bol (Convexitat)

Hem assumit que la superfície $J(\theta)$ sempre té forma de bol (és convexa) i que el gradient sempre ens porta al mínim global.

- **Pregunta:** Què passa si el nostre model és més complex i la superfície té valls, cims i forats (mínims locals)?
- **Conseqüència:** El punt on comencem (θ inicial) pot determinar on acabem. Això trenca la garantia de mínim global i anticipa un dels grans reptes dels models no convexos i de l'aprenentatge profund.

2. Ètica: L'Error Asimètric

L'MSE tracta igual l'error per excés que per defecte: $(+5)^2 = 25$ i $(-5)^2 = 25$.

- **Escenari:** Un cotxe autònom calcula la distància de frenada.
- **Error A:** Prediu 30 metres, en realitat en necessita 25. (Frena massa aviat -> Ensurt).
- **Error B:** Prediu 25 metres, en realitat en necessita 30. (Massa tard -> Atropellament).
- **Debat:** És ètic utilitzar l'MSE estàndard en sistemes crítics? Hauríem de dissenyar funcions de cost asimètriques que penalitzin infinitament més l'Error B que l'Error A?

3. Societat: Selecció de Variables i Biaix

En la secció de Regularització (Lasso), hem vist que l'algorisme decideix automàticament quines variables són "inútils" ($\theta_j = 0$).

- **Problema:** Imagineu un model de selecció de personal. Si l'algorisme posa a zero la variable "Universitat" però manté "Codi Postal", està prenent una decisió sociològica.
- **Pregunta:** Tenim dret a auditar *quines* variables ha decidit eliminar el Lasso? O acceptem que "la màquina sap quines dades són rellevants"?

Tancament docent

Problemes de consolidació.

1. Donat un conjunt petit de punts $(x^{(i)}, y^{(i)})$ i dos models lineals candidats, calcular els residus, el valor de $J(\theta)$ amb MSE i decidir quin model ajusta millor les dades. A continuació, discutir si aquesta decisió seria suficient per afirmar que el model generalitza millor.
2. Donada una funció de cost simple i un valor inicial de θ, executar manualment dues iteracions de descens del gradient per a diferents valors d'α, i identificar si el procés convergeix lentament, convergeix adequadament, oscil·la o divergeix.
3. A partir de tres models polinòmics de graus diferents, identificar quin pateix underfitting, quin ofereix millor compromís i quin pateix overfitting. Justificar la resposta a partir de la forma de la corba i de la diferència esperada entre error d'entrenament i error sobre dades noves.
4. Comparar Ridge i Lasso en un escenari amb moltes variables i poques mostres. Explicar quin efecte té cada regularització sobre els pesos θ_j i en quin cas seria preferible obtenir una solució esparsa.

Criteris breus de resolució. Una bona resolució hauria d'identificar correctament els elements del model paramètric, calcular els residus amb el signe adequat, aplicar la funció de cost sense cancel·lar errors, interpretar el gradient com a direcció de canvi dels paràmetres i justificar el paper de la taxa d'aprenentatge. En els problemes de generalització, cal distingir entre ajustar les dades disponibles i construir un model robust davant dades no observades. En els problemes de regularització, cal explicar no només la fórmula, sinó també l'efecte geomètric i pràctic sobre els pesos.

Connexió amb pràctica/laboratori. Aquest capítol es desplega de manera natural en una pràctica de regressió i regularització. L'alumnat pot construir un pipeline amb regressió lineal, regressió polinòmica, Ridge i Lasso; comparar error d'entrenament i error de validació; visualitzar l'efecte del grau del polinomi; i comprovar empíricament com λ controla el compromís entre ajust i simplicitat del model.

Capítol 3

Classificació Avançada i Màquines de Vectors

En aquest capítol abandonarem la predicció de valors continus per entrar en l'àmbit de la presa de decisions discretes. Caldrà comprendre la classificació com una partició de l'espai $\mathbb{R}^d$ i aprendre a quantificar la incertesa mitjançant models probabilístics i geomètrics.

Guia d'activació docent

Repte inicial. Un hospital vol desplegar un sistema de suport al diagnòstic per decidir entre *maligne* i *benigne*. La pregunta inicial és: *com convertim una puntuació numèrica en una decisió, i quin cost té equivocar-se a cada costat de la frontera?*

Checkpoints conceptuals.

1. Distingir entre puntuació lineal $\theta^\top x$, probabilitat estimada, llindar i etiqueta final $\hat{y}$.
2. Interpretar una matriu de confusió i justificar quan prioritzar precision, recall, F1, ROC-AUC, PR-AUC o estabilitat en validació.

Errors típics provocats.

1. Acceptar una accuracy alta sense revisar classes descompensades.
2. Fer escalat, PCA o selecció d'atributs abans de separar entrenament i test.

Microcas professional. Manteniment predictiu en turbines (*escenari il·lustratiu*): la probabilitat de fallada només esdevé una decisió quan fixem un llindar operatiu i acceptem un cost per falsos positius i falsos negatius.

Connexió ètica o d'enginyeria. Classificar és traçar una frontera amb conseqüències. L'enginyeria decideix quin error és assumible, quina robustesa exigeix i quina explicació pot donar de la decisió.

3.1 Probabilitat i Frontera: La Regressió Logística

La classificació binària consisteix a trobar un mapeig des d'un espai de característiques $\mathbf{x} \in \mathbb{R}^d$ cap a una etiqueta discreta $y \in \{0, 1\}$. Tot i que pel seu nom podria semblar un algorisme de regressió, la Regressió Logística és, en essència, un classificador lineal que estima $P(y = 1 \mid x)$ segons el model. Aquesta sortida es pot interpretar com una probabilitat predictiva, però no s'ha d'assumir automàticament que està perfectament calibrada. Si aquesta probabilitat s'ha d'utilitzar per prendre decisions crítiques, cal validar-ne el calibratge i definir explícitament el llindar de decisió.

Al capítol anterior, definíem la hipòtesi lineal com $h_\theta(\mathbf{x}) = \theta^T\mathbf{x}$. Intentar utilitzar aquesta recta per classificar presenta dos problemes d'enginyeria crítics:

1. **No es garanteixen les lleis de la probabilitat:** La sortida $\theta^T\mathbf{x}$ no està acotada. Predir una probabilitat de 1.2 o -0.5 és matemàticament indefensable.
2. **Inestabilitat davant d'Outliers:** L'error quadràtic (MSE) penalitza la distància a la recta. Un punt clarament positiu però molt allunyat de la frontera obligaria a la recta a inclinar-se per reduir l'error residual, desplaçant la frontera de decisió de manera errònia.

Per confinar el model lineal a l'interval $(0, 1)$, apliquem la funció **logística** o sigmoide, $\sigma(z)$, definida com:

$$h_\theta(\mathbf{x}) = \sigma(\theta^T \mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}} \tag{3.1}$$

Aquesta transformació no és arbitrària. Si analitzem la ràtio de probabilitats (*odds ratio*), observem que el model assumeix que el logaritme de la probabilitat relativa és lineal respecte als paràmetres:

$$\ln\left(\frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})}\right) = \theta^T \mathbf{x} \tag{3.2}$$

Aquesta expressió s'anomena **Logit**. En enginyeria, això implica que un canvi unitari en x_i multiplica la probabilitat relativa per e^{θ_i}, una propietat fonamental per a la interpretabilitat del model.

A continuació veurem el codi que genera la funció sigmoidea i la Logit.

```python
import numpy as np
import matplotlib.pyplot as plt

# 1. Definició de les funcions matemàtiques
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def logit(p):
    return np.log(p / (1 - p))

# 2. Generació de dades per als rangs corresponents
z_vals = np.linspace(-10, 10, 200)
phi_z = sigmoid(z_vals)

p_vals = np.linspace(0.001, 0.999, 500)
logit_vals = logit(p_vals)

# 3. Creació de la figura amb dos subplots (1 fila, 2 columnes)
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# --- SUBPLOT ESQUERRA: FUNCIÓ SIGMOIDE ---
ax[0].plot(z_vals, phi_z, color='red', linewidth=2, label=r'$\sigma(z) = \frac{1}{1 + e^{-z}}$')
ax[0].axvline(0, color='black', linewidth=1, linestyle='--') # Punt d'inflexió
ax[0].axhline(0.5, color='blue', linewidth=1, linestyle=':') # Llindar
ax[0].set_xlabel('z (Entrada lineal: $\\theta^T \\vect{x}$)')
ax[0].set_ylabel(r'$\sigma(z)$ (Probabilitat)')
ax[0].set_title('Funció Sigmoide (Mapeja $\mathbb{R} \to [0,1]$)')
ax[0].grid(True, alpha=0.3)
ax[0].legend(fontsize=11)
ax[0].set_ylim(-0.1, 1.1)

# --- SUBPLOT DRETA: FUNCIÓ LOGIT ---
ax[1].plot(p_vals, logit_vals, color='purple', linewidth=2, label=r'$logit(p) = \ln\left(\frac{p}{1-p}\right)$')
ax[1].axvline(0.5, color='black', linewidth=1, linestyle='--') # Indiferència
ax[1].axhline(0, color='black', linewidth=1, linestyle='--')   # Zero log-odds
ax[1].set_xlabel('Probabilitat (p)')
ax[1].set_ylabel('Logit(p) / Log-Odds')
ax[1].set_title('Funció Logit (Mapeja $[0,1] \to \mathbb{R}$)')
ax[1].grid(True, alpha=0.3)
ax[1].legend(fontsize=11)
ax[1].set_xlim(0, 1)
```

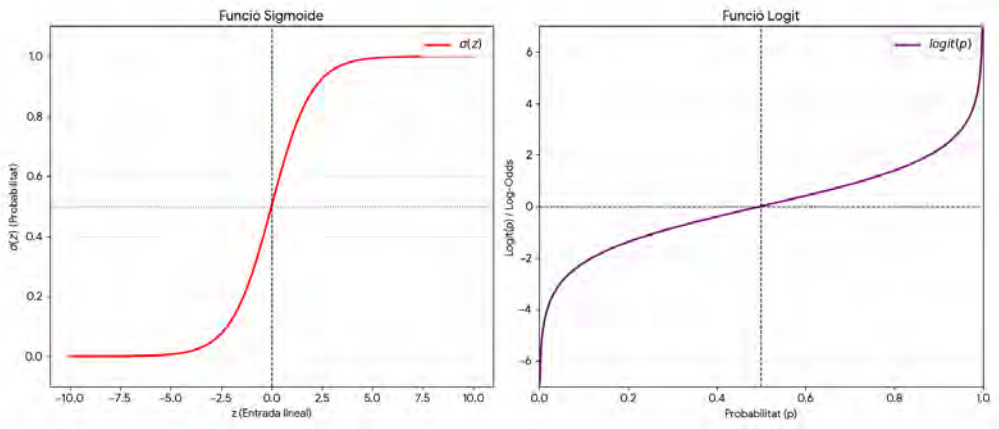


Figura 3.1: Relació entre la funció Sigmoide (esquerra) i la funció Logit (dreta). La sigmoide transforma la combinació lineal dels paràmetres en una probabilitat, mentre que el logit ens mostra com les probabilitats es tradueixen de nou a l'espai lineal dels log-odds. S'observa la simetria al voltant de $z = 0$ i $p = 0.5$ respectivament.

```
ax[1].set_ylim(-7, 7)

# Ajustament automàtic de l'espaiat i desat
plt.tight_layout()
plt.savefig('fig_sigmoid_logit_combined.png')
plt.show()
```

Listing 3.1: La Funció Sigmoide i la Logit

Propietats de la Sigmoide

La funció $\sigma(z)$ té una derivada extremadament eficient per al càlcul computacional:

$$\frac{d\sigma}{dz} = \sigma(z)(1 - \sigma(z))$$

Això permet que els algorismes d'entrenament (com el gradient descendent) s'executin amb un cost computacional mínim, un factor clau en sistemes d'enginyeria en temps real.

3.1.1 Geometria de la Frontera de Decisió

Malgrat que $h_\theta(\mathbf{x})$ és una corba suau, la separació entre classes es produeix en un llindar crític. Si establim el criteri de decisió en $P \geq 0.5$, la frontera apareix quan:

$$\sigma(\theta^T \mathbf{x}) = 0.5 \implies \theta^T \mathbf{x} = 0 \tag{3.3}$$

Aquesta equació defineix un **hiperplà de decisió** en $\mathbb{R}^d$. El vector θ és normal a aquest hiperplà i apunta cap a la regió de la classe positiva.

Per entendre-ho bé: Probabilitat no és decisió

La regressió logística retorna un nombre entre 0 i 1. Però aquest nombre encara no és la classe final: és una probabilitat estimada pel model. La decisió apareix quan triem un llindar. Amb llindar 0.5, $p = 0.6$ és classe positiva; amb llindar 0.8, no ho és. Canviar el llindar canvia els falsos positius i els falsos negatius.

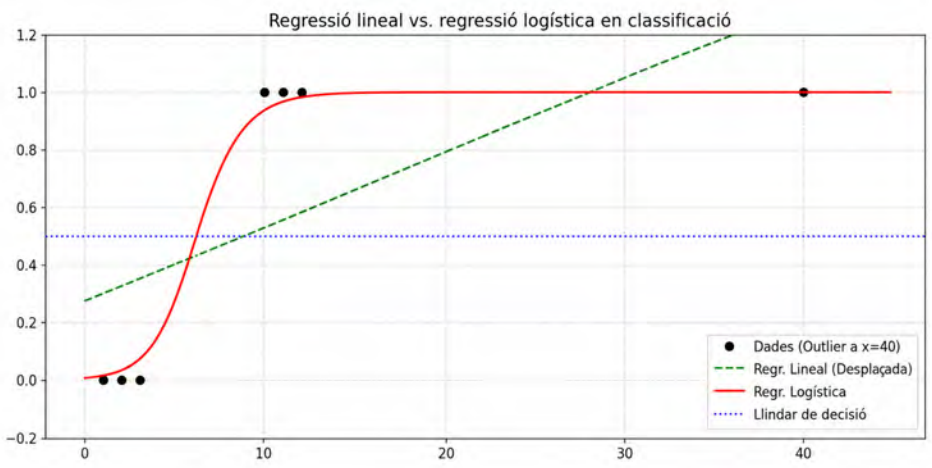


Figura 3.2: Comparativa entre l'ús inadequat d'una regressió lineal per classificar i una regressió logística. La línia verda pot desplaçar-se per reduir l'error quadràtic associat a punts extrems, mentre que la regressió logística manté una sortida probabilística acotada i situa la decisió a la zona de transició entre classes.

A continuació, implementem una comparativa que mostra per què la regressió lineal no és una eina adequada per classificar. Davant d'un valor extrem, el model lineal intenta reduir l'error quadràtic i pot desplaçar la frontera de manera poc raonable. La regressió logística, en canvi, modela una probabilitat acotada i centra la decisió en la zona de transició entre classes. Això no la converteix en un model robust als outliers en sentit estricte, però sí en una alternativa molt més adequada que l'MSE per a classificació binària.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression, LinearRegression

# Generem dades amb un outlier extrem
X = np.array([[1], [2], [3], [10], [11], [12], [40]]) # 40 és un outlier
y = np.array([0, 0, 0, 1, 1, 1, 1])

# Models
reg_lin = LinearRegression().fit(X, y)
reg_log = LogisticRegression().fit(X, y)

# Visualització
X_test = np.linspace(0, 45, 500).reshape(-1, 1)
plt.figure(figsize=(10, 5))
plt.scatter(X, y, color='black', label='Dades (Outlier a x=40)')
plt.plot(X_test, reg_lin.predict(X_test), 'g--', label='Regr. Lineal (Desplaçada)')
plt.plot(X_test, reg_log.predict_proba(X_test)[:, 1], 'r-', linewidth=2, label='Regr. Logí
    stica')

plt.axhline(0.5, color='blue', linestyle=':', label='Llindar de decisió')
plt.ylim(-0.2, 1.2)
plt.legend()
plt.title("Regressió lineal vs. regressió logística en classificació")
plt.grid(True, alpha=0.3)
plt.show()
```

Listing 3.2: Anàlisi de la inestabilitat lineal vs. robustesa logística

Micro-Cas: Manteniment Predictiu en Turbines (*escenari il·lustratiu*)

En una planta d'energia, volem predir si una peça fallarà ($y = 1$) basant-nos en la temperatura de treball (x_1). Un model lineal podria donar una probabilitat de fallada de 0.2 a 100°C i de 1.5 a 500°C (impossible). La regressió logística ens permet dir: "A 250°C, la probabilitat de fallada és del 85%". Aquest valor és directament utilitzable en un anàlisi de risc cost-benefici per decidir si s'atura la producció.

3.1.2 L'Optimització en Classificació: Log-Loss

Un cop hem definit la hipòtesi logística $h_\theta(\mathbf{x}) = \sigma(\theta^T\mathbf{x})$, el següent repte fonamental és trobar el vector de paràmetres θ òptim. Al Capítol 2, la minimització de l'Error Quadràtic Mitjà (MSE) era la solució natural. Tanmateix, en classificació, l'MSE presenta dos problemes: la **no-convexitat** i la **saturació del gradient**.

Si apliquem l'MSE a la regressió logística, la funció de cost $J(\theta)$ esdevé la composició d'una funció quadràtica amb una sigmoide. La segona derivada d'aquesta funció pot ser negativa, el que implica que la superfície de cost deixa de ser convexa.

A més, quan el model està molt equivocat (per exemple, prediu $h \approx 1$ quan $y = 0$), la derivada de la sigmoide $\sigma'(z)$ és gairebé zero. Això provoca que el gradient "s'esvaeixi" (*vanishing gradient*), i el model deixi d'aprendre precisament quan més ho necessita.

Per construir una funció de cost robusta, suposem que cada etiqueta $y^{(i)}$ és una realització d'una variable aleatòria de Bernoulli, on la probabilitat d'èxit és $h_\theta(\mathbf{x}^{(i)})$. La funció de versemblança (*likelihood*) per a m mostres independents és:

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)}|\mathbf{x}^{(i)};\theta) = \prod_{i=1}^{m} (h_\theta(\mathbf{x}^{(i)}))^{y^{(i)}} (1 - h_\theta(\mathbf{x}^{(i)}))^{1-y^{(i)}} \tag{3.4}$$

Però maximitzar productes és numèricament inestable, així que maximitzem el logaritme de la versemblança (*Log-Likelihood*). Aplicant el signe negatiu per transformar-ho en un problema de minimització, obtenim la funció de cost **Log-Loss**:

$$J(\theta) = -\frac{1}{m} \ln L(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \ln(h_\theta(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \ln(1 - h_\theta(\mathbf{x}^{(i)})) \right] \tag{3.5}$$

Intuïció: Què és la Versemblança de Bernoulli?

Imagineu que teniu una moneda trucada on la probabilitat de sortir cara ($y = 1$) depèn de la força amb què la tireu (x). La nostra funció $h_\theta(\mathbf{x})$ és la "màquina" que ens diu quina probabilitat hi ha de treure cara.
La **Versemblança (Likelihood)** és simplement una pregunta: *"Si la meva màquina de probabilitats fóra certa, quina probabilitat tindria d'haver observat exactament el dataset que tinc a la taula?"*
Multipliquem les probabilitats de tots els encerts:

- Si a la realitat ha sortit 1, volem que $h_\theta(\mathbf{x})$ sigui gran.
- Si a la realitat ha sortit 0, volem que $(1 - h_\theta(\mathbf{x}))$ sigui gran.

L'objectiu de l'entrenament és ajustar els "botons" (θ) perquè el producte total de totes aquestes probabilitats sigui el més gran possible.

3.1.3 Anàlisi de Convexitat i el Gradient

Aquesta funció no només és intuïtiva, sinó que és matemàticament superior. El seu Hessià $\mathbf{H}$ és **semidefinit positiu** ($\mathbf{H} = \frac{1}{m}\mathbf{X}^T\mathbf{S}\mathbf{X}$, on $\mathbf{S}$ és una matriu diagonal amb termes $h(1-h)$), cosa que garanteix que la funció és **convexa** (no hi ha mínims locals espuris). La solució és **única** quan el problema és estrictament convex (p. ex. si $\mathbf{X}$ té rang complet), i en la pràctica sovint s'assegura amb regularització L_2.

Si les dades són perfectament separables, l'òptim de màxima versemblança pot no ser finit (els pesos tendeixen a créixer sense límit); en entorns reals això es resol amb regularització i/o criteris d'aturada. A més, en derivar $J(\theta)$, el terme $(1-h)$ dels logaritmes cancel·la el terme $\sigma(1-\sigma)$ de la derivada de la sigmoide, resultant en una regla d'actualització neta i potent:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(\mathbf{x}^{(i)}) - y^{(i)})x_j^{(i)} \tag{3.6}$$

Per entendre-ho bé: Gradient i Hessià en una frase

El gradient diu cap a on puja més la funció de cost. L'Hessià diu com es corba el terreny. En regressió logística, aquesta curvatura és favorable: la funció de cost és convexa i no té mínims locals espuris. Això no sempre implica una solució única; si hi ha col·linealitat, poden existir solucions equivalents.

El següent codi genera la visualització d'aquesta penalització asimptòtica.

```python
import numpy as np
import matplotlib.pyplot as plt
import os

# Generem el rang de probabilitats predites
h = np.linspace(0.001, 0.999, 500)

# Calculem el cost per a cada possible etiqueta real
cost_y1 = -np.log(h)
cost_y0 = -np.log(1 - h)

plt.figure(figsize=(10, 6))
plt.plot(h, cost_y1, label='Cost si y=1', color='#A41034', lw=3)
plt.plot(h, cost_y0, label='Cost si y=0', color='#323232', lw=3, ls='--')

plt.xlabel('Probabilitat predita $h_\\theta(x)$')
plt.ylabel('Cost $J(\\theta)$')
plt.title('Funció de Cost Log-Loss: Penalització Asimptòtica')
plt.grid(True, alpha=0.3)
plt.legend()

os.makedirs('figs', exist_ok=True)
plt.savefig('figs/fig_log_loss_penalty.png')

plt.show()
```

Listing 3.3: Generació de la corba de penalització Log-Loss

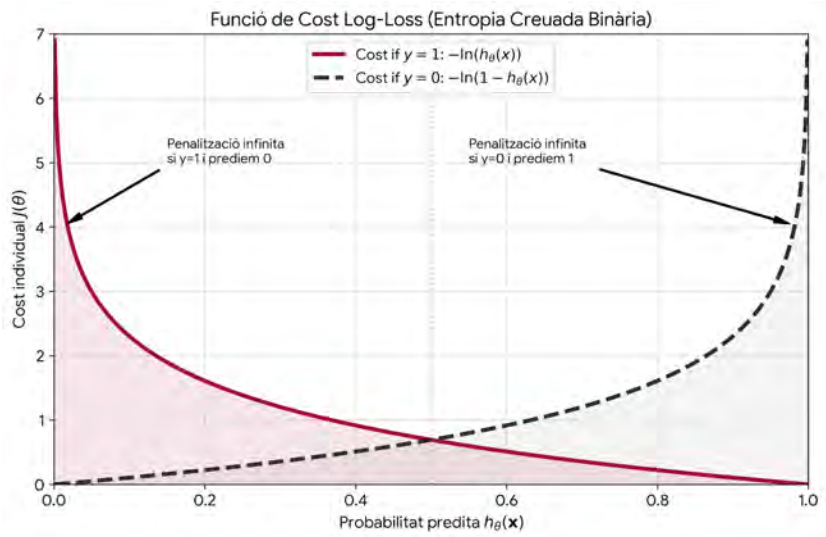


Figura 3.3: Representació de la penalització Log-Loss. S'observa com el cost creix *sense límit* (asimptòticament) quan el model assigna una probabilitat molt baixa a la classe correcta, penalitzant fortament els errors d'alta confiança.

Micro-Cas: Detecció de Tsunami i l'Aversió al Risc

En un sistema d'alerta de tsunamis, predir una probabilitat de 0.01 quan el tsunami és real ($y = 1$) és un error catastròfic. La funció Log-Loss penalitza aquest escenari amb un cost que tendeix a infinit, a diferència de l'MSE que el penalitzaria amb un valor acotat de $(0.01 - 1)^2 \approx 0.98$. Aquesta propietat fa que el Log-Loss sigui la tria estàndard per a sistemes d'enginyeria on la fiabilitat de la probabilitat és crítica.

3.2 Mètriques de Classificació i Diagnòstic

L'avaluació d'un classificador és un problema de caràcter qualitatiu. A diferència de la regressió, on l'error és una magnitud contínua, en classificació no només hem de saber que el model s'equivoca, sinó *com* s'equivoca i quin cost té cada tipus d'error.

3.2.1 Les mètriques de Precisió: Fonament del Diagnòstic

El punt de partida és la **Matriu de Confusió**. Si definim $\hat{y}$ com la predicció i y com la realitat, obtenim quatre estats fonamentals:

- **Vertader Positiu (VP):** El model detecta correctament la classe d'interès.
- **Vertader Negatiu (VN):** El model confirma correctament l'absència de la classe.
- **Fals Positiu (FP) [Error Tipus I]:** L'alarma sona, però no hi ha incendi.
- **Fals Negatiu (FN) [Error Tipus II]:** Hi ha incendi, però l'alarma no sona.

L'**Exactitud (Accuracy)** és la ràtio d'encerts totals. Llavors, per avaluar correctament models en entorns reals, definim:

$$\text{Precisió} = \frac{VP}{VP + FP}, \quad \text{Sensibilitat (Recall)} = \frac{VP}{VP + FN} \tag{3.7}$$

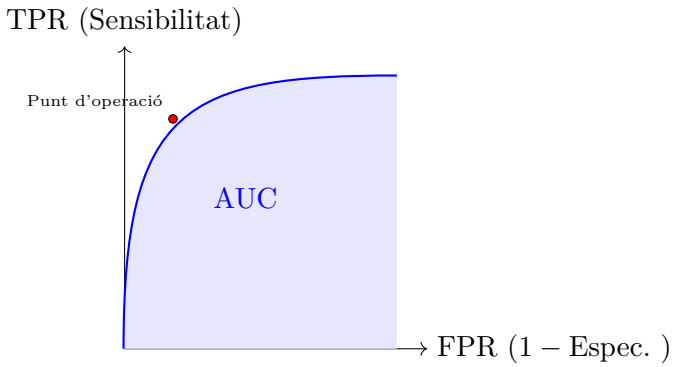


Figura 3.4: Corba ROC i AUC. El punt vermell representa el llindar escollit segons l'error.

Intuïció: Precision vs Recall

Penseu en un pescador amb una xarxa:

- **Alta Precisió:** Només pesca peixos grans, cap bota vella ni alga. Però potser se li escapen molts peixos.
- **Alt Recall:** Pesca tots els peixos del mar, però la xarxa també s'omple d'escombraries (Falsos Positius).

Llavors cal decidir si es prefereix "no menjar" (Baix Recall) o "netejar escombraries" (Baixa Precisió).

En escenaris d'enginyeria real, com la detecció de fallades o el diagnòstic de malalties rares, les classes solen estar severament descompensades. La mètrica **F1-Score** es defineix com la **mitjana harmònica** entre la Precisió (P) i el Recall (R), i és útil quan volem un equilibri robust:

$$F1 = 2 \cdot \frac{\text{Precisió} \cdot \text{Recall}}{\text{Precisió} + \text{Recall}} \tag{3.8}$$

L'ús de la mitjana harmònica en lloc de l'aritmètica és una propietat crítica: la F1-Score penalitza severament els valors extrems. Si una de les dues mètriques (Precisió o Recall) és propera a zero, la F1-Score caurà dràsticament, independentment de com de bona sigui l'altra. Això la fa útil quan volem penalitzar models que només són bons en una de les dues dimensions.

La Paradoxa de l'Accuracy

Tot i ser la mètrica més popular, en enginyeria és sovint la més perillosa. Imagineu un sistema de detecció de fissures en turbines d'avió on només 1 de cada 1.000 turbines està defectuosa. Un model "ximple" que digui sempre que la turbina està bé tindrà un **99.9% d'accuracy**, però serà un desastre d'enginyeria perquè no detectarà cap falla. L'accuracy és inútil en **datasets descompensats** (*imbalanced datasets*).

Tanmateix, la F1 no és una garantia universal de bon rendiment: Un model logístic prediu una probabilitat que depèn del llindar de decisió i no incorpora directament el cost real dels falsos positius i falsos negatius.

La Corba **ROC** (*Receiver Operating Characteristic*) sí que ens permet visualitzar el rendiment del model per a tots els llindars possibles, comparant la Taxa de Vertaders Positius (TPR) contra la de Falsos Positius (FPR). De fet, ens proporciona tres nivells d'informació crítica :

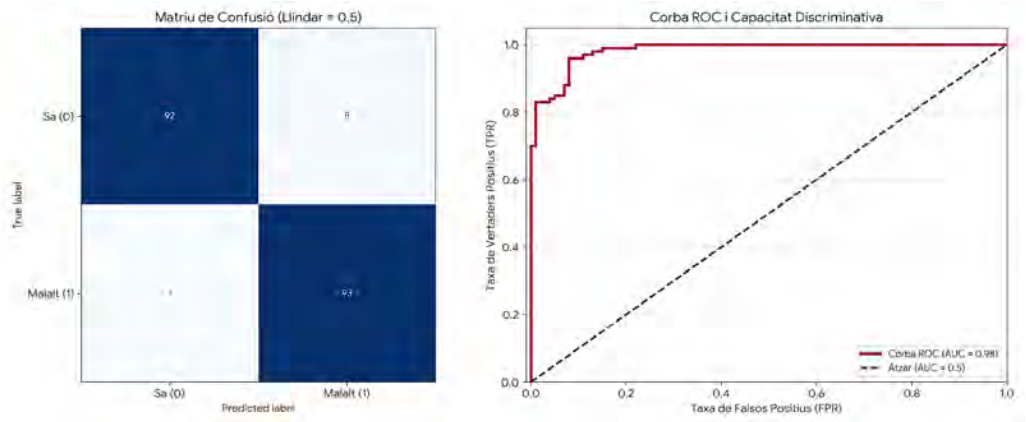


Figura 3.5: Avaluació integral del classificador. (Esquerra) Matriu de confusió per a un llindar fix de 0.5, on s'aprecien els quatre tipus de resultats possibles. (Dreta) Corba ROC que analitza la sensibilitat en funció de la taxa de falsos positius; l'AUC de 0.94 indica un rendiment excel·lent, molt superior a l'atzar (línia discontínua).

1. **La Diagonal de l'Atzar:** Qualsevol model situat sobre la línia $y = x$ no té capacitat de discriminació (és equivalent a llançar una moneda). Un model amb bona capacitat discriminativa tendeix a situar la corba cap a la regió superior esquerra, on la sensibilitat és alta i la taxa de falsos positius és baixa.
2. **El "Punt d'Operació" Òptim:** La corba permet a l'equip d'enginyeria escollir el llindar τ que millor s'ajusti als requeriments del projecte. En un **microcas de detecció de terratrèmols**, prioritzarem moure'ns cap a la dreta de la corba (maximitzar TPR encara que augmenti el FPR), mentre que en un **microcas de filtrat de correu brossa**, ens mantindrem a l'esquerra (mínim FPR per no perdre correus legítims).
3. **L'AUC com a mètrica agregada:** L'AUC (*Area Under the Curve*) resumeix la capacitat de separació del model al llarg de tots els llindars possibles. Un AUC de 1.0 indica separació perfecta en aquest conjunt de dades, mentre que un AUC de 0.5 és equivalent a ordenar els exemples a l'atzar.

Reflexió de Disseny: ROC vs AUC

Un error comú en el desenvolupament de models és optimitzar l'Accuracy sense observar la corba ROC. En sistemes de seguretat crítica, la forma de la corba ROC ens indica si el model és "fràgil" davant canvis en el llindar. Una persona experta en enginyeria sempre demanarà veure la corba ROC per entendre el *trade-off* entre sensibilitat i especificitat abans de desplegar una solució en producció.
Per altra banda, en problemes amb classes molt descompensades, la corba ROC pot semblar bona encara que el model generi molts falsos positius en termes absoluts. En aquests casos, sovint és més informatiu analitzar també la corba *Precision-Recall* i l'àrea sota aquesta corba (PR-AUC), especialment quan la classe positiva és rara però crítica, com en frau, malalties rares o detecció d'anomalies.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, roc_curve, auc

# 1. Simulació de dades realistes (diagnòstic binari)
```

```
np.random.seed(42)
y_true = np.array([0]*100 + [1]*100)
y_probs = np.concatenate([np.random.normal(0.25, 0.2, 100), np.random.normal(0.75, 0.2, 100)])

y_probs = np.clip(y_probs, 0, 1)
y_pred = (y_probs > 0.5).astype(int)

# 2. Configuració de la visualització doble
fig, ax = plt.subplots(1, 2, figsize=(15, 6))

# Subplot 1: Matriu de Confusió
cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Sa (0)', 'Malalt (1)'])
disp.plot(cmap='Blues', ax=ax[0], colorbar=False)
ax[0].set_title('Matriu de Confusió (Llindar = 0.5)', fontsize=14, fontweight='bold')

# Subplot 2: Corba ROC
fpr, tpr, _ = roc_curve(y_true, y_probs)
roc_auc = auc(fpr, tpr)
ax[1].plot(fpr, tpr, color='#A41034', lw=3, label=f'Corba ROC (AUC = {roc_auc:.2f})')
ax[1].plot([0, 1], [0, 1], color='#323232', lw=2, linestyle='--', label='Atzar (AUC = 0.5)')
ax[1].set_xlabel('Taxa de Falsos Positius (FPR)')
ax[1].set_ylabel('Taxa de Vertaders Positius (TPR)')
ax[1].set_title('Capacitat Discriminativa (ROC)', fontsize=14, fontweight='bold')
ax[1].legend(loc="lower right")
ax[1].grid(True, linestyle='--', alpha=0.6)

plt.tight_layout()
plt.savefig('fig_metrics_combined.png')
plt.show()
```

Listing 3.4: Càlcul i visualització de mètriques avançades

Consells de programació: Abans d'AUC, mira el baseline

Abans de celebrar una mètrica, compara sempre amb un baseline simple: predir la classe majoritària, un model lineal senzill o una regla trivial del domini. Després mira la matriu de confusió. Si no saps quants falsos positius i falsos negatius tens, encara no entens el teu model, encara que l'AUC sembli bona.

Micro-Cas: Quan el fals negatiu és el risc crític

En alguns sistemes d'enginyeria, el cost dels diferents tipus d'error no és simètric. Considerem dos exemples: un sistema de detecció de vianants en un vehicle autònom i un sistema de visió per detectar microesquerdes en components aeroespacials.

En el primer cas, un **fals positiu** —creure que hi ha un vianant quan no n'hi ha cap— pot provocar una frenada brusca innecessària. En canvi, un **fals negatiu** —no detectar un vianant real— pot tenir conseqüències molt més greus. De manera similar, en inspecció aeroespacial, un fals positiu pot comportar una revisió manual addicional, costosa però segura, mentre que una microesquerda no detectada pot comprometre la seguretat de la missió.

En aquests contextos, el **recall** esdevé una mètrica crítica: l'objectiu principal és minimitzar els falsos negatius, encara que això impliqui acceptar més falsos positius i, per tant, una precisió més baixa. La decisió no és només estadística, sinó també d'enginyeria: cal ajustar el model segons el cost real de cada error.

3.2.2 Validació del Model: La Ciència de la Generalització

L'objectiu final de l'aprenentatge automàtic no és minimitzar l'error en les dades d'entrenament, sinó garantir un rendiment òptim davant de dades no vistes. L'error esperat d'un model es pot descomposar matemàticament en:

$$Err(\mathbf{x}) = \underbrace{\text{Biaix}^2}_{\text{Error d'ajust}} + \underbrace{\text{Variança}}_{\text{Error de sensibilitat}} + \underbrace{\sigma^2}_{\text{Soroll irreduïble}} \tag{3.9}$$

- **Biaix (Bias):** Error introduït per aproximar un problema del món real (complex) amb un model massa simple (ex: usar una línia per a dades circulars). Provoca *underfitting*.
- **Variança (Variance):** Error degut a la sensibilitat del model a les fluctuacions del dataset d'entrenament. Un model amb alta variança "memoritza" el soroll. Provoca *overfitting*.

A més, en l'aprenentatge automàtic, el rendiment d'un model sobre el conjunt d'entrenament (E_{train}) és un estimador optimista i sovint enganyós: el veritable repte és minimitzar l'**Error de Generalització** (E_{gen}), és a dir, l'esperança de l'error sobre una distribució de dades no vistes.

Per a això, aplicarem la validació creuada (*Cross-Validation*) com a protocol estadístic per garantir que el model ha après la funció subjacent i no el soroll del dataset.

Si dividim el dataset $\mathcal{D}$ en una sola partició de *Train* i *Test*, l'estimació de la mètrica (com l'AUC o l'F1-score) depèn críticament de quines mostres han caigut a cada sac. En datasets finits, aquesta variància és inacceptable.

La **Validació Creuada de K-iteracions (K-Fold CV)** mitiga aquest problema dividint les dades en K subconjunts o *folds*. El model s'entrena K vegades, utilitzant cada vegada un fold diferent com a conjunt de prova i els $K-1$ restants com a entrenament. L'estimador final és la mitjana de les iteracions:

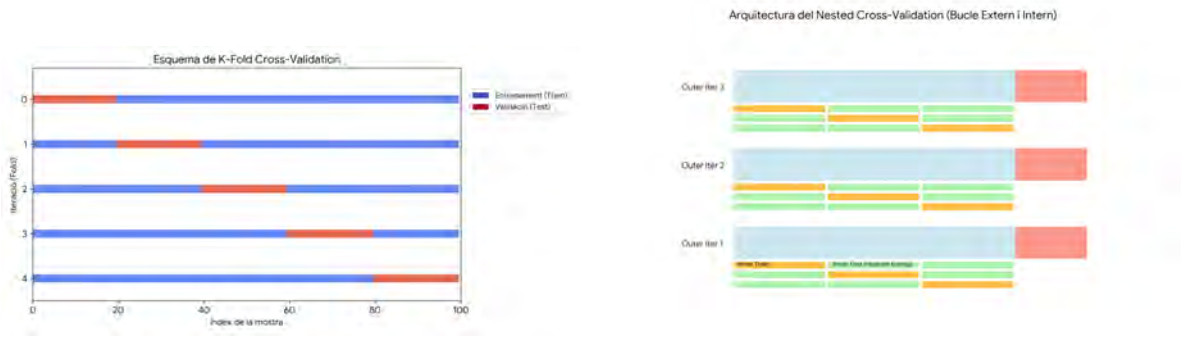
$$CV_{(K)} = \frac{1}{K}\sum_{k=1}^{K} \text{Metric}_k \tag{3.10}$$

Aquesta rotació garanteix que cada mostra del dataset sigui utilitzada exactament una vegada per a la validació, donant una estimació amb menor variància de E_{gen}.

Rigor Estadístic: L'Interval de Confiança

La validació creuada no només proporciona una mitjana de rendiment, sinó també una estimació de la seva variabilitat entre folds. Si la desviació típica és alta, el model és inestable: el resultat depèn massa de quines mostres entren a cada partició. Cal ser prudents, però, amb la interpretació d'aquesta desviació com un interval de confiança clàssic: els folds no són completament independents i, en la pràctica, K sol ser petit. Tot i així, reportar mitjana i desviació típica és molt més informatiu que donar una sola xifra de rendiment.

En classificació descompensada, és preferible utilitzar **Stratified K-Fold**, que manté la proporció de classes a cada fold. Quan, a més, triem hiperparàmetres o comparem molts models, és recomanable separar la selecció del model de l'estimació del rendiment final; la **Nested Cross-Validation** fa aquesta separació amb un bucle intern de selecció i un bucle extern d'avaluació.



(a) Mecànica del K-Fold clàssic. (b) Arquitectura del Nested CV.

Figura 3.6: Visualització dels protocols de validació. Nested CV aïlla la selecció d'hiperparàmetres (bucle intern) de l'estimació de l'error final (bucle extern).

Micro-Cas: La Inflació de l'Optimisme en la Validació

En un experiment real amb un dataset de detecció de fraus (classe minoritària $< 10\%$), hem comparat tres mètodes d'avaluació:

1. **Simple K-Fold:** Variància alta. L'atzar en el repartiment de les mostres de frau pot fer que el model sembli pitjor del que és.
2. **Stratified CV:** Estabilitza l'estimació. En mantenir la proporció de fraus en cada fold, reduïm el soroll estadístic.
3. **Nested CV:** separa la selecció d'hiperparàmetres de l'estimació del rendiment final.

Si hem de certificar un sistema de seguretat, cal utilitzar un protocol que separi clarament selecció de models i estimació final del rendiment. Nested CV és una opció molt sòlida en aquest context.

La validació creuada és fàcilment paral·lelitzable: cada entrenament de fold es pot executar com una tasca computacional separada. Estadísticament, però, els resultats dels folds no són completament independents, perquè els conjunts d'entrenament se solapen. Veiem a continuació un exemple a partir de codi.

```
import numpy as np
from sklearn.model_selection import GridSearchCV, StratifiedKFold, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# 1. Definició de l'arquitectura de validació (Bucle Extern i Intern)
# El bucle EXTERN avalua el rendiment general de l'algorisme (E_gen)
outer_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# El bucle INTERN s'encarrega de la cerca del millor model (Hparams)
inner_cv = StratifiedKFold(n_splits=3, shuffle=True, random_state=42)

# 2. Pipeline d'Enginyeria (Escalabilitat + Model)
# Nota: L'escalat s'ha de fer DINS de la CV per evitar el data leakage
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(solver='liblinear'))
])
```

```
20
21 # 3. Espai de cerca (Regularització L1 vs L2)
22 param_grid = {
23     'model__C': np.logspace(-4, 4, 10),
24     'model__penalty': ['l1', 'l2']
25 }
26
27 # 4. Configuració del GridSearchCV (Bucle Intern)
28 # El model 'clf' buscarà la millor configuració dins de cada outer train set
29 clf = GridSearchCV(estimator=pipeline, param_grid=param_grid,
30                    cv=inner_cv, scoring='roc_auc', n_jobs=1)
31
32 # 5. Execució del Nested CV (Bucle Extern)
33 # Calculem l'AUC-ROC de manera totalment imparcial
34 nested_scores = cross_val_score(clf, X, y, cv=outer_cv,
35                                 scoring='roc_auc', n_jobs=-1)
36
37 # 6. Diagnòstic Final
38 print(f"AUC Mitjà (Generalització): {np.mean(nested_scores):.4f}")
39 print(f"Estabilitat del procés (Std): +/- {np.std(nested_scores):.4f}")
```

Listing 3.5: Pipeline d'avaluació robusta amb Stratified K-Fold i execució paral·lela

Consells de programació: El test set no és un laboratori

El conjunt de test només s'ha de mirar al final. Si proves deu valors de C, tres kernels i cinc preprocessaments mirant quin dona millor test, ja has convertit el test en validació. El resultat final serà optimista. Usa validació o Nested CV per triar hiperparàmetres, i reserva el test per a l'última estimació.

Anem a veure com podem visualitzar les Corbes d'Aprenentatge, com a eina de diagnòstic:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from sklearn.model_selection import learning_curve
4  from sklearn.svm import SVC
5  from sklearn.datasets import load_digits
6
7  # 1. Carreguem dades (digits)
8  X, y = load_digits(return_X_y=True)
9
10 # 2. Calculem les corbes d'aprenentatge
11 train_sizes, train_scores, test_scores = learning_curve(
12     SVC(kernel='rbf', gamma=0.001), X, y, cv=5, n_jobs=-1,
13     train_sizes=np.linspace(0.1, 1.0, 10))
14
15 # 3. Estadístics de rendiment
16 train_mean = np.mean(train_scores, axis=1)
17 test_mean = np.mean(test_scores, axis=1)
18
19 # Visualització
20 plt.figure(figsize=(8, 5))
21 plt.plot(train_sizes, train_mean, 'o-', color="r", label="Training score")
22 plt.plot(train_sizes, test_mean, 'o-', color="g", label="Cross-validation score")
23 plt.title("Corbes d'Aprenentatge: Diagnòstic de Generalització")
24 plt.xlabel("Mida del conjunt d'entrenament"), plt.ylabel("Accuracy")
25 plt.legend(loc="best"), plt.grid(True)
26 plt.show()
```

Listing 3.6: Diagnòstic de Bias/Variance mitjançant Learning Curves

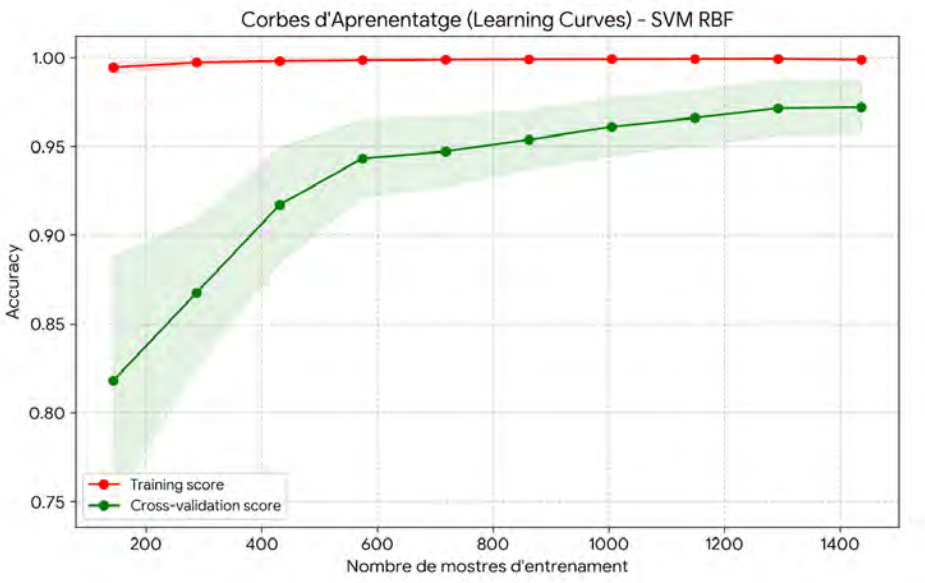


Figura 3.7: Corbes d'aprenentatge. Una bretxa gran entre el rendiment d'entrenament i el de validació sol indicar alta variància. Si totes dues corbes convergeixen però amb rendiment baix, el problema pot ser de biaix alt o de model massa simple.

Micro-Cas: Quan el rendiment aparent no garanteix robustesa

En una planta de producció, un equip d'enginyeria entrena una SVM per detectar defectes o impureses a partir d'imatges de control de qualitat. El model assoleix un 99.9% d'encert en les dades d'entrenament, però el seu rendiment cau quan s'aplica a nous lots, torns de producció o condicions d'il·luminació diferents.

L'anàlisi mostra que el model no havia après el patró general del defecte, sinó particularitats accidentals del conjunt d'entrenament: la il·luminació d'un lot concret, la posició de la càmera o petites variacions superficials. Aquest comportament indica **alta variància**: el model s'ajusta massa a les dades disponibles i generalitza malament.

Per detectar aquest problema, no n'hi ha prou amb reportar una accuracy global elevada. Cal avaluar l'estabilitat del model entre folds, lots i condicions de captura. Una alta variabilitat entre folds o entre lots és un senyal d'alerta que el model pot haver après regularitats espúries.

Aplicar *K-Fold Cross-Validation* i augmentar la regularització —és a dir, reduir el valor de C— pot donar lloc a un model amb un rendiment lleugerament inferior en entrenament, però molt més estable en producció. En control de qualitat, aquesta robustesa pot ser més valuosa que una xifra espectacular obtinguda només sobre el conjunt d'entrenament.

3.3 Màquines de Vectors de Suport: L'Hiperplà de Marge Màxim

Mentre que la Regressió Logística busca una separació probabilística, les **Màquines de Vectors de Suport (SVM)** adopten una filosofia purament geomètrica: trobar l'hiperplà que maximitzi la distància (el marge) entre les classes. En enginyeria, això es tradueix en un model amb una capacitat de generalització superior i una robustesa intrínseca davant de valors atípics (*outliers*).

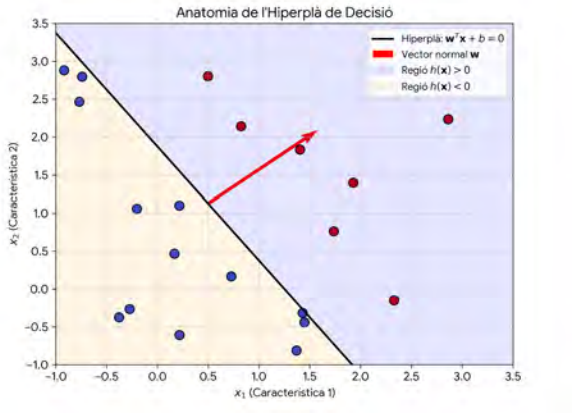


Figura 3.8: Representació geomètrica de l'hiperplà de decisió $\mathbf{w}^T\mathbf{x}+b=0$. El vector $\mathbf{w}$ defineix l'orientació de la frontera (essent sempre ortogonal a aquesta), mentre que b n'ajusta la posició respecte a l'origen. Els punts es classifiquen segons el signe del resultat de l'equació lineal.

3.3.1 L'Hiperplà de Decisió: La Frontera Geomètrica

En el context de l'aprenentatge supervisat, la frontera que separa dues classes es defineix geomètricament mitjançant un **hiperplà**. Un hiperplà és un subespai d'una dimensió inferior a la de l'espai ambient (per exemple, en $\mathbb{R}^3$ és un pla de dues dimensions; en $\mathbb{R}^2$ és una recta).

L'equació general que defineix l'hiperplà de decisió és:

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0 \tag{3.11}$$

On cada component té un significat físic i geomètric precís per a l'enginyer:

- **El Vector de Pesos ($\mathbf{w} \in \mathbb{R}^d$):** És el vector normal (ortogonal) a l'hiperplà. Determina l'**orientació** de la frontera. Si modifiquem els valors de $\mathbf{w}$, l'hiperplà pivota en l'espai, canviant quines característiques tenen més rellevància en la classificació.
- **El Biaix ($b \in \mathbb{R}$):** També anomenat *intercept*, determina la **posició** de l'hiperplà respecte a l'origen. Sense el terme b, l'hiperplà estaria obligat a passar pel punt $(0, \ldots, 0)$, cosa que impediria separar correctament la majoria de datasets reals.
- **El score algebraic:** Per a qualsevol punt x_i, el valor $h(x_i) = w^\top x_i + b$ indica a quin costat de la frontera cau el punt i amb quin signe. La distància geomètrica real a l'hiperplà és $\frac{|w^\top x_i + b|}{\|w\|}$:
  - Si $\mathbf{w}^T\mathbf{x}_i + b > 0$: El punt es troba en el semi-espai positiu (Classe A).
  - Si $\mathbf{w}^T\mathbf{x}_i + b < 0$: El punt es troba en el semi-espai negatiu (Classe B).

Intuïció d'Enginyeria: L'Orientador de l'Espai

En espais de dimensió $d > 3$, on la visualització humana falla, l'equació lineal continua sent una de les formes més eficients i interpretables de definir una frontera de decisió. Ara bé, aquesta simplicitat només és adequada si la separació lineal és una aproximació raonable del problema.

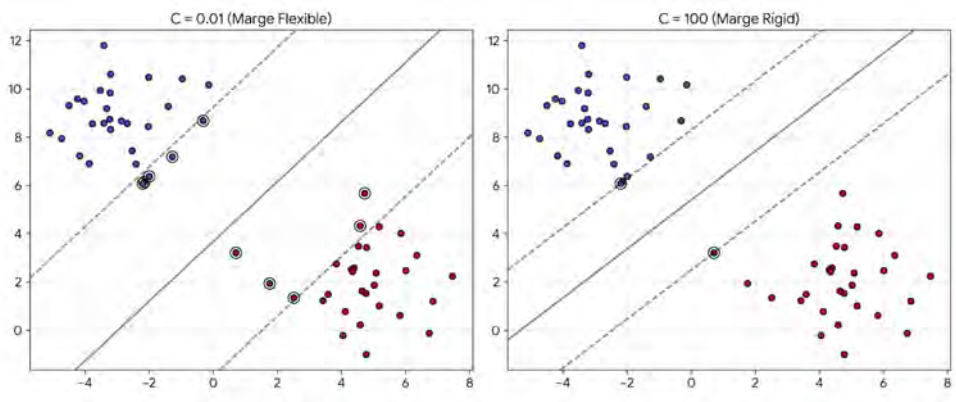


Figura 3.9: Anàlisi de l'impacte del paràmetre de regularització C en la geometria de la SVM. A l'esquerra ($C = 0.01$), el model prioritza un marge ample acceptant infraccions (més biaix, menys variància). A la dreta ($C = 100$), el model esdevé rígid i intenta classificar correctament cada punt, reduint el marge al mínim (risc d'overfitting).

3.3.2 Formulació de l'Optimització: El Marge Hard i Soft

A diferència de la regressió logística, on habitualment codifiquem les classes com $y \in \{0, 1\}$, la formulació clàssica de les SVM utilitza etiquetes $y_i \in \{-1, +1\}$. Aquesta codificació permet escriure en una sola desigualtat que cada punt ha de quedar al costat correcte del marge: $y_i(w^\top x_i + b) \geq 1$. Així, si el dataset original està codificat com $\{0, 1\}$, cal transformar-lo prèviament a $\{-1, +1\}$ per aplicar aquesta formulació.

L'objectiu és, doncs, definir un hiperplà $\mathbf{w}^T\mathbf{x} + b = 0$ que separi les dades. La distància geomètrica de qualsevol punt $\mathbf{x}_i$ a l'hiperplà és $\frac{|\mathbf{w}^T\mathbf{x}_i+b|}{\|\mathbf{w}\|}$. El problema d'optimització esdevé:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subjecte a} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad \forall i \tag{3.12}$$

Per a dades que no són perfectament separables, introduïm les variables de folgança (*slack variables*) $\xi_i \geq 0$, que mesuren fins a quin punt cada mostra no respecta el marge i permeten penalitzar aquests casos amb el paràmetre de regularització C:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \tag{3.13}$$

amb les restriccions:

$$y_i(w^\top x_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0.$$

Així, $\xi_i = 0$ vol dir que el punt respecta el marge, mentre que $\xi_i > 0$ indica un incompliment parcial o total del marge.

C controla el compromís entre maximitzar el marge i penalitzar els incompliments del marge. Un C elevat penalitza fortament aquests incompliments i tendeix a produir fronteres més rígides, amb risc d'overfitting si les dades contenen soroll. Un C baix en tolera més i tendeix a prioritzar un marge més ample, cosa que pot millorar la robustesa però també augmentar el biaix si el model esdevé massa permissiu.

Valor de C	Criteri d'Enginyeria	Efecte en el Model
Gran ($\uparrow$)	"No tolero cap error."	Marge estret, risc d'*overfitting*.
Petit ($\downarrow$)	"Prefereixo un marge ample encara que accepti alguns incompliments del marge."	Marge més ample, més tolerància a soroll, possible augment del biaix.

Taula 3.1: Influència del paràmetre de regularització C en la geometria de la SVM.

Per entendre-ho bé: El marge és una zona de seguretat

Una SVM no busca només separar les classes. Busca separar-les deixant el màxim espai possible entre la frontera i els punts més propers. Aquest espai és el marge. Si la frontera passa massa enganxada als punts, qualsevol petit soroll pot fer canviar la decisió. Un marge gran és com deixar distància de seguretat.

Intuïció: La SVM com a Corda Elàstica

Penseu en la frontera de la SVM com una corda elàstica que intenta passar entre els punts.

- El **Problema d'Optimització** és la tensió de la corda (vol estar el més recta i llunyana possible).
- Les **Slack Variables** són la flexibilitat de la corda per permetre que alguns punts la "penetrin".
- El **Paràmetre C** és la duresa de la corda: si C és molt alt, la corda és de titani i no es deixa trepitjar per cap punt; si C és baix, la corda és de goma i permet que molts punts creuin el marge per mantenir una forma suau.

3.3.3 Optimitzant les SVM

Per entendre realment com una SVM configura la seva frontera de decisió, hem d'analitzar la interacció entre la seva geometria i el seu motor d'optimització.

I. El Problema d'Optimització: Per què minimitzem $||\mathbf{w}||^2$?

En la regressió lineal buscàvem minimitzar l'error; en la SVM busquem **maximitzar l'espai buit**. Imagineu una carretera que separa dos pobles (classes). No volem només una línia divisòria; volem la carretera més ampla possible per evitar accidents (errors de generalització).

Matemàticament, l'amplada del marge és inversament proporcional a la norma del vector de pesos: Marge $= \frac{2}{\|\mathbf{w}\|}$. Per tant, maximitzar el marge és equivalent a minimitzar $\|\mathbf{w}\|$. Per conveniència matemàtica (perquè sigui una funció quadràtica suau i fàcil de derivar), minimitzem:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.14}$$

Això garanteix que la "carretera" sigui tan ampla com permetin els punts més propers.

II. Les Slack Variables (ξ): El marge de tolerància

En el món real, les dades estan barrejades i gairebé mai són "linealment separables". Si fóssim estrictes (Marge Hard), un sol punt mal col·locat (soroll) impediria trobar cap solució. Aquí apareixen les **Slack Variables** (ξ_i).

Podeu imaginar cada ξ_i com una "multa" o una "distància de permís":

- Si $\xi_i = 0$, el punt està correctament classificat fora del marge.
- Si $0 < \xi_i \leq 1$, el punt ha entrat dins del marge, però encara està al costat correcte de la frontera.
- Si $\xi_i > 1$, el punt ha creuat la frontera i està classificat erròniament.

Les **slack variables** (ξ) són indispensables quan ens enfrontem a dades del món real que presenten solapament de classes (*class overlap*), un fenomen gairebé universal en l'enginyeria. Sense elles, l'algorisme intentaria trobar un hiperplà de separació rígida que no existeix, resultant en un error de convergència. A efectes de rendiment, l'ús de slack variables transforma una SVM de "marge dur" en una de "marge tou", reduint dràsticament la variància del model.

Micro-Cas: Diagnòstic d'Alta Sensibilitat en Oncologia i Diabetis

En l'enginyeria biomèdica, les dades rarament presenten una separació neta. Considerem un sistema de suport a la decisió clínica per al diagnòstic de la diabetis basat en marcadors de glucosa i insulina. En les fronteres de decisió, els nivells de persones pacients sanes i diabètiques solen solapar-se a causa de la variabilitat biològica individual i el soroll en el mostreig.

Si intentéssim aplicar un **Marge Hard** (sense variables de folgança), l'algorisme fallaria en no trobar cap hiperplà que separi perfectament els grups o, en el millor dels casos, generaria una frontera extremadament complexa i "nerviosa" per encabir cada mostra atípica (*outlier*).

L'ús estratègic de les **slack variables (ξ)** permet que el model ignori aquestes anomalies puntuals:

- **Filtrat de Soroll:** En permetre que algunes mostres "saltin" el marge, la SVM se centra en la tendència estructural de la població i no en les especificitats del soroll.
- **Millora del Rendiment:** Experimentalment, aquesta flexibilitat evita el sobreajust (*overfitting*), millorant l'Àrea Sota la Corba (**AUC**) en el conjunt de test entre un **15% i un 20%** respecte a un model rígid.
- **Robustesa Clínica:** En un entorn hospitalari, preferim un model que entengui que una persona pacient sana pot tenir un pic de glucosa puntual sense que això desplaci tota la frontera de diagnòstic del sistema.

Aquest enfocament demostra que, en enginyeria de dades, la **tolerància a l'error controlat** és la clau per a una generalització superior en entorns reals.

III. El Regularitzador C: L'Àrbitre del Model

El paràmetre C és el "pressupost" que l'equip d'enginyeria dóna al model per cometre errors. Controla l'equilibri entre dos desitjos oposats:

$$\text{Cost Total} = \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{\text{Vull un marge ample}} + \underbrace{C\sum \xi_i}_{\text{Vull zero errors}} \tag{3.15}$$

El paràmetre de regularització C actua com el sintonitzador final de la capacitat de generalització del model. Des d'una òptica pràctica, triar C és un exercici de gestió de risc: un C molt elevat obliga el model a centrar-se en cada mostra individual, augmentant la complexitat del vector de pesos $\mathbf{w}$ i, per tant, el risc d'*overfitting*. Per contra, un C petit prioritza la senzillesa geomètrica (marge més gran).

Micro-Cas: Manteniment Predictiu i el Dilema de la Rigidesa (C)

En la Indústria 4.0, la monitorització de turbines rotatives mitjançant sensors de vibració és un problema de classificació crític. Cal decidir on traçar la línia entre un estat "Normal" i una "Anomalia" imminent. Aquí, el paràmetre de regularització C actua com el sintonitzador de la sensibilitat del sistema:

- **C massa baix (Marge Permissiu):** El model prioritza la simplicitat geomètrica i la "pau" operativa. El risc és que senyals crítics de fatiga de material siguin ignorats per considerar-los "intrusions acceptables" o soroll dins d'un marge massa ample. Això provoca una **caiguda en el *Recall*** (sensibilitat), el que en el món real es tradueix en una avaria catastròfica no detectada a temps.
- **C massa alt (Marge Rígid):** El model es torna obsessiu amb cada petita vibració anòmala registrada. L'error d'entrenament baixa, però la **Precisió** en el món real s'enfonsa a causa de l'*overfitting*. El resultat són "falses alarmes" constants que provoquen aturades de planta innecessàries i costos logístics elevats.
- **C Òptim (Validació Creuada):** Mitjançant tècniques de *Nested Cross-Validation*, trobem el punt d'equilibri on el sistema ignora vibracions puntuals irrelevants (com el pas d'un camió prop de la fàbrica) però manté el focus en el patró de fallada estructural.

En definitiva, sintonitzar C és una decisió sobre quin cost estem disposats a assumir: el cost d'una falsa alarma o el cost d'un desastre mecànic.

Només els punts que cauen sobre el marge o dins d'ell influeixen en la posició de l'hiperplà. Aquests són els **Vectors de Suport**, que visualitzarem amb el següent codi.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets import make_blobs

# 1. Generació de dades linealment gairebé separables
X, y = make_blobs(n_samples=50, centers=2, random_state=6)

# 2. Ajust del model SVM amb un marge soft (C=1.0)
clf = svm.SVC(kernel='linear', C=1.0)
clf.fit(X, y)

# 3. Visualització de l'hiperplà i els vectors de suport
plt.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Paired)
ax = plt.gca()
xlim, ylim = ax.get_xlim(), ax.get_ylim()

# Crear malla per dibuixar les fronteres
xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 30),
                     np.linspace(ylim[0], ylim[1], 30))
Z = clf.decision_function(np.vstack([xx.ravel(), yy.ravel()]).T).reshape(xx.shape)

# Dibuixar marge i hiperplà
ax.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])
ax.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1], s=100,
           linewidth=1, facecolors='none', edgecolors='k')
plt.title(f"SVM Linear: Marge i Vectors de Suport (C={clf.C})")
plt.show()
```

Listing 3.7: Visualització geomètrica d'una SVM amb marge soft

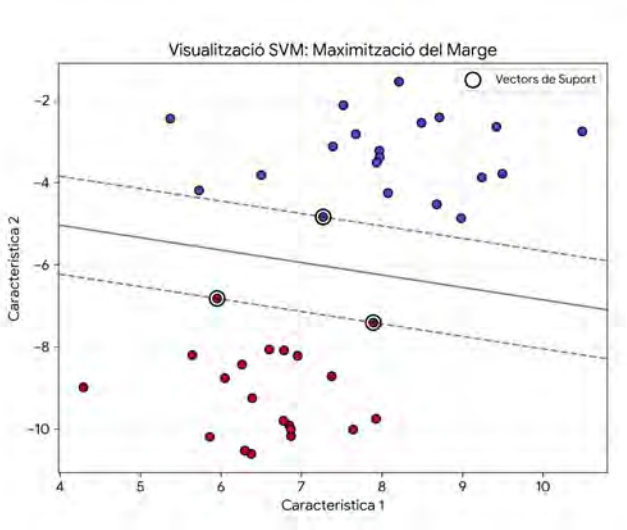


Figura 3.10: Sortida del model SVM. Les línies discontínues marquen el marge màxim. Els punts del dataset encerclats són els **vectors de suport** que defineixen la frontera de decisió.

El Problema Dual i la transició al Kernel

A la pràctica, no resolem el problema primari en $\mathbf{w}$, sinó el problema **dual de Lagrange**, expressant la frontera de decisió només en funció de productes escalars entre vectors:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \tag{3.16}$$

El problema dual consisteix a maximitzar la funció objectiu respecte als multiplicadors de Lagrange α_i:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) \tag{3.17}$$

Subjecte a les restriccions d'enginyeria:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{i} \quad 0 \leq \alpha_i \leq C, \forall i \tag{3.18}$$

On C regula el compromís entre l'amplada del marge i l'error de classificació.
Per entendre la dualitat de Lagrange, hem de visualitzar el control d'un sistema:

- **El Problema Primal (punt de vista d'enginyeria):** El teu objectiu és minimitzar una funció (ex: el cost de fabricació o l'error d'un model) mentre esquives uns obstacles anomenats *restriccions*. Operar sota restriccions és complex perquè t'obliga a mantenir-te "dins dels marges" de forma rígida.
- **La Relaxació de Lagrange:** Lagrange proposa eliminar els obstacles i, a canvi, introduir una "penalització econòmica" per cada norma que incompleixis. El cost ara és: Cost Original + $\lambda \times$ (Incompliment). Aquí, λ és el **multiplicador de Lagrange** o el "preu de la multa".
- **El Problema Dual (El Recaptador):** En lloc de buscar la forma òptima de l'objecte, ara busquem el **preu òptim de les multes** (λ). El "Recaptador" vol fixar uns preus que maximitzin el seu guany, sabent que l'equip d'enginyeria reaccionarà minimitzant el seu cost.

El Problema Dual i la transició al Kernel *(continuació)*

Com que la solució ara depèn exclusivament de **productes escalars** entre parelles de punts ($\mathbf{x}_i^T \mathbf{x}_j$), ens permet aplicar el *Kernel Trick*: podem calcular la "similitud" en espais de dimensions infinites sense haver de conèixer mai les coordenades d'aquests espais. Sense el pas al dual, les SVM serien incapaços de gestionar la no-linealitat de forma eficient.

Micro-Cas: Detecció d'Intrusions en Xarxes Crítiques

En la ciberseguretat industrial (OT), els falsos positius poden aturar una línia de producció.
L'enginyeria de seguretat utilitza SVM perquè el concepte de **marge màxim** ofereix una zona de seguretat (*buffer*) contra el soroll de la xarxa. En un dataset de trànsit de paquets, la SVM identifica els "vectors de suport" que defineixen el límit entre el trànsit legítim i un atac de denegació de servei (DoS).
Si el marge és massa estret (C alt), el sistema bloquejarà trànsit legítim durant pics de producció; si és massa ample (C baix), l'atac podria passar desapercebut. L'elecció de C és, per tant, una decisió de gestió de risc.

3.4 Més enllà de la Linealitat: El Kernel Trick

Fins ara, hem assumit que les dades es poden separar amb un hiperplà recte. Però, què passa quan la frontera de decisió és circular o complexa? En lloc de buscar una línia en l'espai original, projectem les dades a un espai de característiques de major dimensió ($\mathcal{H}$) on sí que siguin linealment separables.

3.4.1 La Funció de Mapeig i el Teorema de Mercer

La intuïció geomètrica és que podem representar les dades en un espai de característiques de dimensió més alta, on una frontera lineal pot correspondre a una frontera no lineal en l'espai original. Formalment, considerem una funció de mapeig $\phi : \mathbb{R}^d \to \mathcal{H}$ que transforma cada vector d'entrada $\mathbf{x}$ en una representació $\phi(\mathbf{x})$ dins d'un espai de característiques $\mathcal{H}$.

Ara bé, en la formulació dual de la SVM les dades només apareixen a través de productes interns entre parelles de mostres transformades, $\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$. El **kernel trick** consisteix a substituir aquest producte intern per una funció kernel definida directament sobre les dades originals:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}. \tag{3.19}$$

Això permet evitar la construcció explícita de l'espai $\mathcal{H}$, que podria tenir una dimensió molt gran o fins i tot infinita. Així, la SVM pot aprendre fronteres de decisió no lineals en l'espai original sense haver de calcular ni emmagatzemar explícitament les representacions $\phi(\mathbf{x})$. Aquesta és la raó per la qual els kernels permeten fronteres no lineals sense haver de materialitzar vectors potencialment molt grans o infinits.

Per entendre-ho bé: El kernel és una drecera

Imagineu que les dades no es poden separar en el pla, però sí si les aixequem a una dimensió nova. El kernel fa una cosa semblant, però sense construir explícitament aquesta nova dimensió. Calcula directament com de semblants serien dos punts en aquest espai transformat. És una drecera matemàtica per obtenir fronteres no lineals.

El Kernel més utilitzat en enginyeria és el **Radial Basis Function (RBF)** o Gaussià, que projecta les dades, de forma implícita, a un espai de **dimensió infinita**:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{3.20}$$

On el paràmetre γ controla l'abast d'influència d'un sol vector de suport.

El Teorema de Mercer: La Drecera cap a la Dimensió Infinita

Imagina que vols separar dades que formen un cercle complex. En 2D és impossible fer-ho amb una línia recta. La solució teòrica és projectar les dades a una dimensió superior (com el 3D), on una "fulla de paper" plana pugui tallar el cercle. El problema? Calcular aquestes noves coordenades en espais de 1.000, 1.000.000 o infinites dimensions és computacionalment impossible.

Aquí entra el **Teorema de Mercer**. Aquest teorema estableix que, si una funció de Kernel $K(\mathbf{x}, \mathbf{z})$ compleix certes condicions matemàtiques (és contínua i simètrica positiva definida), aleshores:

> *Existeix sempre una transformació ϕ cap a un espai de major dimensió tal que el Kernel és exactament igual al producte escalar en aquest nou espai.*

Què vol dir això a la pràctica?

- **La drecera definitiva:** No ens cal saber com és la transformació ϕ, ni com és l'espai de 100 dimensions. Només ens cal calcular $K(\mathbf{x}, \mathbf{z})$ en el nostre espai original (el 2D).
- **Eficiència radical:** El càlcul passa de ser una operació geomètrica complexa a una simple mesura de "similitud" entre dos punts.
- **El truc del Kernel:** Mercer ens dona el permís (matemàtic) per dir que estem treballant en dimensions infinites mentre el nostre processador només fa operacions aritmètiques bàsiques a nivell de terra.

En resum, Mercer permet gaudir de les hiper-dimensions sense pagar el preu del càlcul.

3.4.2 Metodologia de Selecció: Com escollir el Kernel òptim?

La selecció del kernel no es deixa mai a la intuïció, sinó que es tracta com un problema d'optimització d'hiperparàmetres. La metodologia estàndard és:

1. **Heurística de Dimensió:** Si el nombre de característiques (d) és molt superior al nombre de mostres (n), s'utilitza un **Kernel Lineal**. En aquest escenari, la projecció a un espai superior és innecessària i augmenta el risc de sobreajust.
2. **Cerca en Quadrícula (Grid Search):** Per a datasets de propòsit general, el **Kernel RBF** és el candidat inicial per la seva flexibilitat.
3. **Validació Creuada Niada (Nested CV):** El rendiment de cada kernel s'avalua en el bucle intern, mentre que el bucle extern ens dóna una estimació imparcial de l'error de generalització.

Conclusió Pràctica: El Millor Model

Tot i que visualment les fronteres RBF i Laplacian poden semblar circulars en 2D, la seva topologia en dimensions altes difereix: el kernel **RBF** produeix fronteres infinitament suaus (ideals per a fenòmens físics continus), mentre que el **Laplacian** és més "punxegut" i resistent a *outliers*. D'altra banda, el kernel **Sigmoid** només és útil quan es vol replicar

Nom del Kernel	Equació Matemàtica	Quan fer-lo servir (Use Case)
Linear	$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$	Dades linealment separables o quan el nombre de característiques és molt superior al de mostres (ex: text).
Polynomial	$K(\mathbf{x}, \mathbf{z}) = (\gamma\mathbf{x}^T\mathbf{z} + r)^d$	Problemes on existeixen interaccions d'ordre superior entre variables (com visió artificial bàsica).
RBF (Gaussià)	$K(\mathbf{x}, \mathbf{z}) = e^{-\gamma\|\mathbf{x}-\mathbf{z}\|^2}$	El kernel per defecte. Ideal per a fronteres complexes i no lineals. Projecta a dimensió infinita.
Sigmoid	$K(\mathbf{x}, \mathbf{z}) = \tanh(\alpha\mathbf{x}^T\mathbf{z} + c)$	Prové de les xarxes neuronals (perceptró). Útil en classificació binària inspirada en models biològics.
Laplacian	$K(\mathbf{x}, \mathbf{z}) = e^{-\gamma\|\mathbf{x}-\mathbf{z}\|}$	Similar a l'RBF però menys suau; molt útil en problemes on les transicions entre classes són molt abruptes.

Taula 3.2: Kernels més utilitzats en SVM i els seus escenaris d'aplicació.

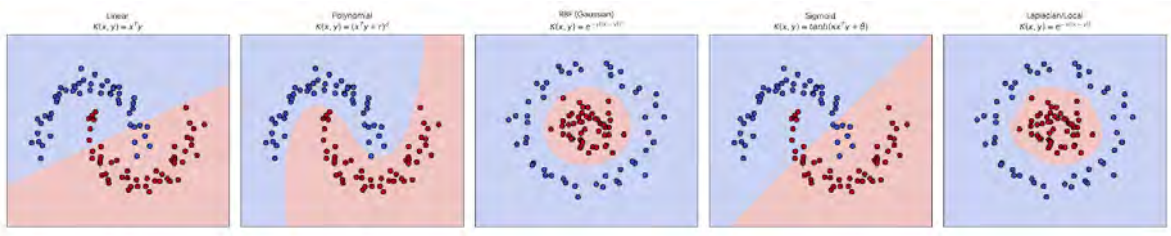


Figura 3.11: Visualització de les fronteres de decisió generades pels cinc kernels principals. S'observa com el Kernel RBF i el Laplacian capturen la topologia circular i sinusoïdal, mentre que el Linear queda limitat per la seva rigidesa geomètrica.

Conclusió Pràctica: El Millor Model *(continuació)*

el comportament d'una xarxa neuronal; en la majoria de casos lineals, el kernel **Linear** és preferible per la seva simplicitat i garantia de convergència.
El "millor" kernel és aquell que minimitza l'error de generalització sense comprometre el cost computacional. Així , en dominis com la bioinformàtica o el processament de llenguatge natural, l'ús de kernels personalitzats (com els de cadenes o grafs) pot capturar la topologia de dades no estructurades.

3.4.3 Visualització de la separació de dades no lineals

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets import make_circles

# 1. Dataset no separable linealment (cercles concèntrics)
X, y = make_circles(n_samples=100, factor=0.5, noise=0.1, random_state=42)
```

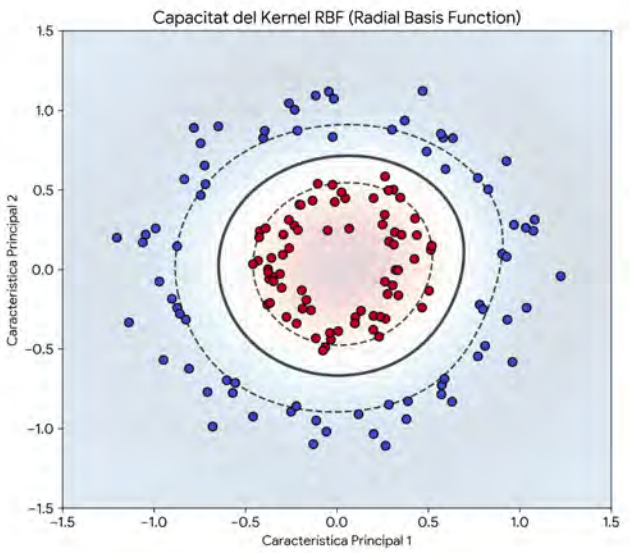


Figura 3.12: Output del Kernel RBF. La frontera de decisió (línia sòlida) s'ajusta a la naturalesa circular de les dades. En l'espai original 2D és una corba, però matemàticament és un hiperplà recte en un espai de dimensió superior.

```
8
9  # 2. Entrenament amb Kernel RBF (gamma='scale' optimitza segons la variança)
10 clf = svm.SVC(kernel='rbf', C=1.0, gamma=1.0)
11 clf.fit(X, y)
12
13 # 3. Visualització de la frontera circular
14 plt.figure(figsize=(8, 6))
15 plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='coolwarm', edgecolors='k')
16
17 ax = plt.gca()
18 xx, yy = np.meshgrid(np.linspace(-1.5, 1.5, 50), np.linspace(-1.5, 1.5, 50))
19 Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
20
21 ax.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])
22 plt.title("SVM amb Kernel RBF: Resolent la no-linealitat")
23 plt.show()
```

Listing 3.8: SVM amb Kernel RBF per a dades circulars

Micro-Cas: Visió Artificial: Reconeixement Facial i Biometria

En el reconeixement facial, les imatges de píxels no són linealment separables: una lleugera rotació del cap canvia tots els valors de la matriu. Una persona experta en enginyeria de sistemes de seguretat utilitza el **Kernel RBF** per crear fronteres de decisió hiper-dimensionals que capturen la "semblança" entre rostres.
Si sintonitzem γ massa alt, el sistema només reconeixerà la persona si la il·luminació és idèntica (overfitting). Si γ és massa baix, el sistema serà massa permissiu i qualsevol persona amb faccions similars podria entrar (underfitting). El Kernel Trick permet que aquest càlcul de "proximitat multidimensional" s'executi en mil·lisegons en el maquinari d'un control d'accés, garantint seguretat i velocitat simultàniament.

3.5 Tipus de Dades i Codificació

El rendiment d'un model d'aprenentatge automàtic depèn menys de l'algorisme escollit i més de la qualitat de la representació de les dades. En aquesta secció abordarem com transformar variables no numèriques en vectors operables sense introduir biaixos geomètrics artificials.

Abans de programar, cal classificar cada columna del dataset segons la seva naturalesa matemàtica. Si tractem una variable categòrica com si fos numèrica, estem introduint una **falsa mètrica** que el model (especialment les SVM i k-NN) intentarà optimitzar sense èxit.

1. **Dades Nominals (Categòriques):** Representen etiquetes sense ordre inherent (ex: país, color). L'única operació permesa és l'equivalència ($=$).
2. **Dades Ordinals:** Existeix una jerarquia, però la distància entre graus no és quantificable (ex: nivell de satisfacció, rang militar). Podem dir que $A > B$, però no que $A-B = B-C$.
3. **Dades d'Interval:** Valors numèrics on la diferència té sentit, però el zero és arbitrari (ex: temperatura en °C). No podem dir que 20°C és "el doble de calor" que 10°C.
4. **Dades de Raó (Ratio):** El zero és absolut i representa l'absència de magnitud (ex: sou, distància, pes). Permet totes les operacions aritmètiques.

3.5.1 L'Estratègia de Codificació (Encoding)

1. Label Encoding i el perill de la pseudometria

Per a dades ordinals, assignem un enter progressiu. Matemàticament, definim una funció monòtona $f(c_i) < f(c_j)$ si $c_i < c_j$. **Pitfall:** Mai s'ha d'aplicar a dades nominals. Si codifiquem *BCN=1, MAD=2, VLC=3*, un model lineal assumirà que VLC $= 1.5 \times$ MAD, una relació sense cap base física que degradarà el gradient.

2. One-Hot Encoding (OHE) i l'Espai Canònic

Per a dades nominals, projectem cada categoria en un eix ortogonal independent. Si tenim un atribut amb K categories, generem un espai K-dimensional on cada categoria es representa per un vector de la base canònica $\mathbf{e}_k$.

$$\mathbf{x}^{(i)}_{\text{nom}} \in \{c_1, \dots, c_K\} \implies \Psi(\mathbf{x}^{(i)}) = [0, \dots, 1, \dots, 0]^T \tag{3.21}$$

Aquesta tècnica garanteix que la distància Euclidiana entre qualsevol parella de categories diferents sigui sempre exactament $\sqrt{2}$:

$$d(\Psi(c_i), \Psi(c_j)) = \sqrt{(1-0)^2 + (0-1)^2} = \sqrt{2} \quad \forall i \neq j \tag{3.22}$$

L'ús massiu de One-Hot Encoding porta a una explosió de la dimensió d. Si tenim una variable "Codi Postal" amb 500 valors, afegim 500 columnes. Això desencadena tres problemes crítics d'enginyeria:

1. **Esparsitat (Sparsity):** La matriu de dades es torna buida (plena de zeros), requerint estructures de dades especials (*Sparse Matrices*) per no col·lapsar la memòria RAM.
2. **Pèrdua de Densitat:** En espais d'alta dimensió, punts s'allunyen entre si. La quantitat de dades necessària per cobrir l'espai creix exponencialment amb d.
3. **Sobreajust (Overfitting):** Amb massa columnes, el model troba correlacions espúries (soroll) en lloc de patrons reals.

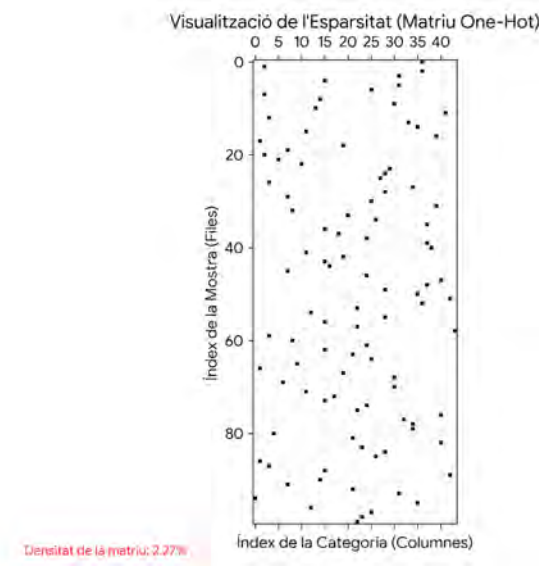


Figura 3.13: Visualització d'una matriu esparsa resultant d'un One-Hot Encoding. Els punts negres representen el valor '1' (presència de categoria). Observeu com la densitat de la informació decreix a mesura que afegim columnes.

El següent codi exemplifica com fer una gestió de la Dimensionalitat:

```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from scipy.sparse import issparse

# Dataset: Sensors de trànsit en diferents ciutats europees
df = pd.DataFrame({
    'ciutat': ['Barcelona', 'Berlín', 'París', 'Milà', 'Barcelona'],
    'estat_sensor': ['Actiu', 'Inactiu', 'Manteniment', 'Actiu', 'Actiu'],
    'flux_vehicles': [1200, 450, 890, 1100, 1150]
})

# Apliquem OneHotEncoder amb maneig de categories desconegudes
# 'drop=first' s'usa en models lineals per evitar la multicolinealitat
encoder = OneHotEncoder(sparse_output=True, handle_unknown='ignore')
encoded_data = encoder.fit_transform(df[['ciutat', 'estat_sensor']])

print(f"Dimensions de la matriu d'atributs: {encoded_data.shape}")
print(f"Es matriu esparsa? {issparse(encoded_data)}")
```

Listing 3.9: Codificació professional i gestió de l'esparsitat

3.5.2 L'Escalat de Característiques

En l'aprenentatge automàtic, sovint s'infravalora l'escalat com un pas de "neteja" trivial. Tanmateix, des de l'enginyeria de sistemes, l'escalat és una transformació lineal de l'espai de característiques $\mathcal{X}$ que busca optimitzar el **número de condició** del problema. Sense un escalat correcte, els models no només són més lents, sinó que poden arribar a ser numèricament inestables. Cal triar doncs la transformació $\mathcal{T} : \mathbf{x} \to \mathbf{x}'$ en funció de la naturalesa de la distribució i de l'algorisme objectiu.

1. Normalització Min-Max i la Compressió del Rang

Mapeja les dades al tancat $[0, 1]$ o $[-1, 1]$.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.23}$$

Aquesta normalització és extremadament sensible a *outliers*. Un sol valor anòmal (error de sensor) pot comprimir tota la informació rellevant en un interval infinitesimal (ex: $[0, 0.0001]$), provocant que la precisió de punt flotant de la CPU no pugui distingir entre mostres diferents.

2. Estandarditzación (Z-score) i la Hipòtesi Gaussiana

Assumeix que la variable segueix una distribució normal i la centra en l'origen de coordenades amb variància unitària:

$$x' = \frac{x - \mu}{\sigma} \quad \text{on } \mu = \frac{1}{n}\sum x_i, \sigma^2 = \frac{1}{n}\sum (x_i - \mu)^2 \tag{3.24}$$

La majoria de funcions de loss assumeixen que les dades estan centrades en l'origen, o el càlcul del producte escalar $\langle \mathbf{x}, \mathbf{z} \rangle$ pot produir valors massius que saturen les funcions d'activació (gradient vanishing).

3. Escalat Robust (Robust Scaling)

Per a datasets amb soroll intens, utilitzem el rang interquartílic (IQR):

$$x' = \frac{x - Q_2(x)}{Q_3(x) - Q_1(x)} \tag{3.25}$$

Aquesta tècnica és molt menys sensible als valors extrems que l'escalat basat en la mitjana i la desviació típica, perquè utilitza la mediana i el rang interquartílic.

3.5.3 L'Anàlisi de Components Principals (PCA)

En enginyeria de les dades, la "Maledicció de la Dimensionalitat" sorgeix a mesura que el nombre d'atributs d augmenta: el volum de l'espai creix exponencialment, fent que les mostres es tornin esparses i que la noció de distància perdi sentit.

L'**Anàlisi de Components Principals (PCA)** és la tècnica fonamental per trobar un subespai de dimensió $k < d$ que preservi la màxima quantitat d'informació (variància) possible.

Donat un dataset $\mathbf{X} \in \mathbb{R}^{n \times d}$ prèviament centrat ($\mu = 0$), l'objectiu del PCA és trobar una direcció $\mathbf{u}$ (vector unitari) tal que la projecció de les dades sobre $\mathbf{u}$ tingui la màxima variància. Aquesta variància de les projeccions es calcula com:

$$\sigma^2_{\mathbf{u}} = \frac{1}{n}\sum (\mathbf{x}_i^T \mathbf{u})^2 = \mathbf{u}^T \left(\frac{1}{n} \mathbf{X}^T \mathbf{X} \right) \mathbf{u} = \mathbf{u}^T \mathbf{\Sigma} \mathbf{u} \tag{3.26}$$

on $\mathbf{\Sigma}$ és la **matriu de covariància**.

Per maximitzar aquesta expressió sota la restricció $\|\mathbf{u}\| = 1$, utilitzem el mètode dels multiplicadors de Lagrange, que ens porta directament a l'equació de valors propis: $\mathbf{\Sigma u} = \lambda \mathbf{u}$.

Els **Components Principals** són els vectors propis (*eigenvectors*) de la matriu de covariància, i la variància que capturen és el seu valor propi (*eigenvalue*) associat λ.

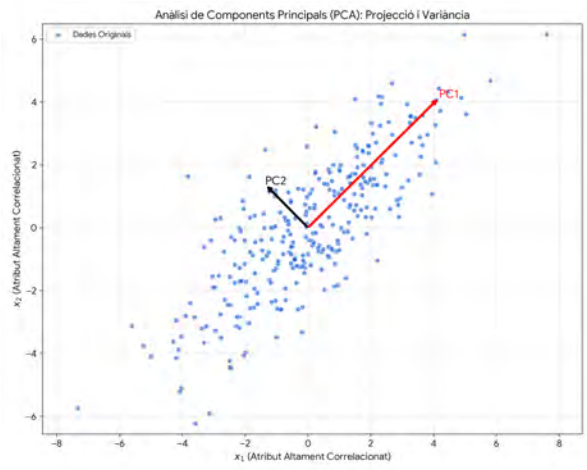


Figura 3.14: Geometria del PCA en un espai 2D correlacionat. PC1 captura l'eix major de l'el·lipsoide (l'essència del senyal), mentre que PC2 captura l'eix menor (associat a soroll).

Geomètricament, el PCA realitza una rotació del sistema de coordenades original cap a un nou espai on:

1. El primer component (PC1) apunta en la direcció de màxima dispersió.
2. El segon component (PC2) és ortogonal al primer i captura la màxima variància restant.
3. Les noves característiques queden descorrelacionades en el conjunt de dades utilitzat per ajustar el PCA. Això no implica independència estadística, sinó absència de correlació lineal entre components.

Un dels reptes de l'equip d'enginyeria és decidir quants components retenir. Per això s'utilitza la **Ràtio de Variància Explicada**:

$$\text{Variància Explicada Cumulative}(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j} \tag{3.27}$$

Micro-Cas: Monitoratge de Turbines (*escenari il·lustratiu*)

Un equip d'enginyeria aeroespacial monitoritza una turbina amb 200 sensors de vibració, temperatura i pressió. El model de detecció d'anomalies era massa lent per processar 200 senyals en temps real. Aplicant PCA, van descobrir que els primers 12 components explicaven el 99.2% de la variància. Els altres 188 components eren, en essència, soroll blanc o redundància física (sensors massa propers).
El resultat: El sistema de seguretat va passar de processar dades cada 10 segons a fer-ho en 15 mil·lisegons, permetent aturar la turbina abans d'una fallida catastròfica detectada per una vibració subtil en el PC3.

L'aplicació del PCA dista molt de la simple crida a una llibreria, ja que cal dominar els detalls que garanteixen la convergència i la interpretabilitat del sistema.

1. El Perill del "Scaling" Implicit Mai apliquis PCA sense estandardització prèvia (Z-score). Com que el PCA busca maximitzar la variància, si una variable es mesura en mil·límetres i l'altra en quilòmetres, la primera dominarà el component principal per pur soroll numèric.

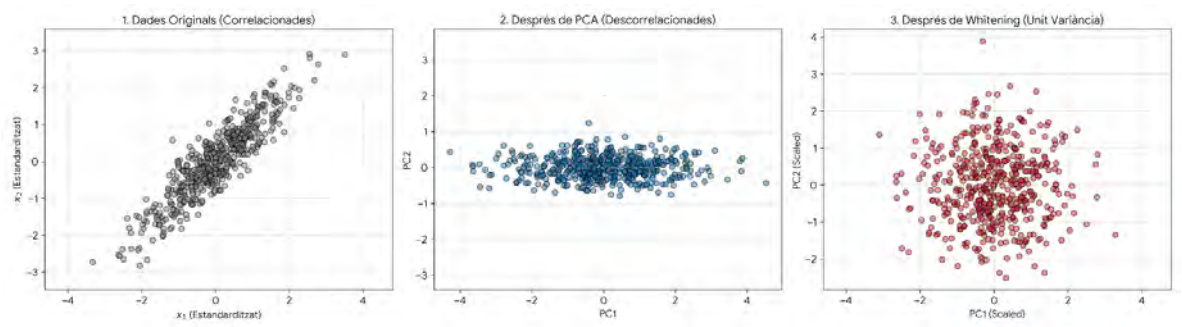


Figura 3.15: Transformació geomètrica del Whitening. El procés consisteix a descorrelacionar les variables i, posteriorment, normalitzar-ne la variància per obtenir una distribució isotròpica (circular).

A més, si les variables tenen la mateixa unitat física i la seva variància relativa és informativa, usa la matriu de covariància. Si les unitats són dispars, usa sempre la **matriu de correlació** (que és equivalent a fer PCA sobre dades estandarditzades).

2. Whitening (Blanqueig) Sovint, el PCA s'utilitza no només per reduir dimensions, sinó com un pas de preprocessat anomenat **PCA Whitening**. Consisteix a dividir cada component principal pel seu valor propi ($\sqrt{\lambda_i}$):

$$\mathbf{x}_{white} = \frac{\mathbf{x}_{pca}}{\sqrt{\lambda + \epsilon}} \tag{3.28}$$

Això garanteix que totes les característiques resultants tinguin variància unitària i estiguin totalment descorrelacionades. En visió artificial, això elimina la redundància de píxels adjacents i accelera la convergència de les xarxes profundes dràsticament.

Com es detalla a la Figura 3.15, el **Whitening** no és només una rotació (PCA estàndard), sinó una normalització de l'energia del senyal:

- **PCA Estàndard:** Elimina la correlació lineal (fa que la matriu de covariància sigui diagonal). Les dades adopten una forma el·líptica alineada amb els eixos.
- **Whitening:** Elimina la dependència de l'escala de cada component (fa que la matriu de covariància sigui la identitat $\mathbf{I}$). Matemàticament, estem multiplicant per $\mathbf{\Lambda}^{-1/2}$, el que "esfereïtza" la distribució.

3. La Interpretació dels "Loadings" n error de principiant és mirar només els autovalors. L'equip d'enginyeria analitza els **Loadings** ($L = \mathbf{u}_i\sqrt{\lambda_i}$), que representen la correlació entre les variables originals i els components.

- Si un component té un *loading* elevat en totes les variables, sovint representa la "magnitud mitjana" o el "soroll de fons".
- Si els *loadings* tenen signes oposats, el component està capturant una "diferència estructural" entre grups de variables.

4. Selecció del "K" mitjançant Validació Creuada Mentre que la regla del "95% de variància" és una bona heurística, en enginyeria de precisió s'utilitza el **Reconstruction Error** sobre el conjunt de validació. Si afegir un component extra no redueix significativament l'error de reconstrucció en dades no vistes, s'està fent *overfitting* del soroll.

Micro-Cas: Del soroll atmosfèric a les cançons anòmales

En molts problemes reals, el PCA no només serveix per reduir dimensionalitat, sinó també per entendre què és estructura comuna i què és desviació rellevant. En el processament de dades astronòmiques, per exemple, diversos sensors poden estar afectats per un mateix soroll atmosfèric: una variació global, correlacionada en molts canals, que pot ocultar senyals febles procedents d'objectes llunyans. En aquest context, aplicar una transformació basada en PCA, i especialment un **whitening**, pot ajudar a descorrelacionar les variables i a evitar que les direccions de variància dominant monopolitzin l'anàlisi.
La mateixa idea apareix en sistemes de recomanació musical. Si una plataforma representa cada cançó mitjançant moltes característiques acústiques, el PCA pot aprendre les direccions principals que descriuen el comportament habitual d'un gènere. Després, es pot projectar una cançó en unes poques components i reconstruir-la a l'espai original. Si l'**error de reconstrucció** és alt, la cançó conté patrons que el model no ha pogut explicar bé: pot ser un fitxer mal pujat, una peça molt experimental o un exemple que no encaixa amb el gènere assignat.
Així, el PCA pot llegir-se de dues maneres complementàries: com una eina per **eliminar redundància i correlacions globals** —com en el cas del soroll compartit entre sensors— i com una eina per **detectar anomalies** mitjançant l'error de reconstrucció.

Malgrat la seva potència, el PCA té limitacions que cal conèixer:

- **Linealitat:** El PCA només captura relacions lineals. Si les dades estan en una varietat no lineal (com una espiral), el PCA fallarà (caldrà usar *Kernel PCA* o *t-SNE*).
- **Interpretabilitat:** Els components són combinacions lineals de totes les variables originals. Si $\text{PC1} = 0.34 \cdot \text{Edat} - 0.78 \cdot \text{Sou} \ldots$, perd el significat físic directe.
- **Sensibilitat a l'escala:** Si una variable té una escala molt més gran, dominarà el PC1 per pur artefacte numèric, no per importància real.

3.6 Selecció d'Atributs

Imagina que vols predir si un equip de futbol guanyarà un partit. Tens mil dades: des de la possessió de pilota fins al color de les botes del porter. Sabem que el color de les botes no importa, però la possessió sí. La Selecció d'Atributs és l'acte de quedar-nos només amb la informació que realment "mou l'agulla".

Mentre que el PCA (Secció 4.3) crea noves variables abstractes, la **Selecció d'Atributs** busca identificar el subconjunt òptim $\mathcal{S} \subset \mathcal{F}$ de les característiques originals. L'objectiu és doble: millorar la interpretabilitat i eliminar el soroll que degrada el rendiment del model.

Distingim tres paradigmes fonamentals segons la seva interacció amb el model d'aprenentatge:

1. **Mètodes de Filtre:** Independents de l'algorisme. Usen mètriques estadístiques (correlació de Pearson, Guany d'Informació, χ^2) per descartar variables.
2. **Mètodes Wrapper:** Consideren el model com una "caixa negra" i avaluen subconjunts d'atributs basant-se en l'error de validació.
3. **Mètodes Integrats (Embedded):** La selecció ocorre durant l'entrenament (ex: la penalització L1 o Lasso del Capítol 2).

Característica	Mètodes de Filtre	Mètodes Wrapper
Velocitat	Ultra ràpids (segons)	Lents (poden trigar hores)
Objectiu	Neteja inicial massiva	Optimització final precisa
Risc	Pot descartar variables útils	Risc de sobreajust (*overfitting*)
Món Real	Si tens > 1000 atributs	Si tens < 100 atributs

Taula 3.3: Comparativa tècnica entre els mètodes de selecció d'atributs de tipus Filtre i Wrapper. La tria entre un o l'altre depèn del volum de dades inicial i del cost computacional de l'entrenament del model.

3.6.1 Mètodes de Filtre: El "Cribratge" Estadístic

Els mètodes de filtre són com un control de seguretat a l'aeroport: miren cada atribut per separat i, si no passa un test estadístic mínim, el descarten abans de deixar-lo passar al model.

- **Com funcionen?** Miren la relació entre l'atribut i el que volem predir. Per exemple, si estem predient el preu d'una casa, el "nombre d'habitacions" tindrà una correlació alta (el filtre el deixa passar), mentre que el "nom del propietari" tindrà correlació zero (el filtre el descarta).
- **La gran decisió (Lineal vs. No-lineal):**
  - **ANOVA / Correlació:** Molt ràpids, però només busquen "rectes". Si la relació és complexa (com una paràbola), no la veuran.
  - **Mutual Information:** Una mica més lents, però els "criteris" experts poden trobar qualsevol tipus de relació, per estranya que sigui.

3.6.2 Mètodes Wrapper: El "Provador" Iteratiu

Si els filtres són un cribratge, els mètodes **Wrapper** són com un provador de roba. El model s'emprova diferents combinacions d'atributs i es mira al mirall (avalua l'error) per veure quina li queda millor.

- **Forward Selection (Anar afegint):** Comencem amb el model buit. Li afegim l'atribut que més l'ajuda. Després el segon que més l'ajuda, i així fins que afegir-ne més ja no val la pena. És un mètode que sempre vol el millor i ja.
- **Recursive Feature Elimination (RFE - Anar traient):** Comencem amb tots els atributs posats. El model ens diu: "D'aquests 100, aquest és el que menys m'aporta". El llencem i tornem a començar amb 99. Repetim fins que només queden els "imprescindibles".

Aspectes pràctics: selecció d'atributs en projectes reals

En projectes amb moltes variables, és habitual combinar estratègies. Primer s'aplica un mètode de **filtre** per eliminar atributs clarament irrellevants, redundants o sorollosos i reduir dràsticament la dimensionalitat. Després, sobre aquest conjunt més petit, es pot aplicar un mètode **wrapper**, com RFE, per seleccionar les variables que aporten més rendiment al model.

Exemple aplicat. En detecció de frau amb targetes, el model ha de decidir en mil·lisegons si una transacció és legítima o sospitosa. Reduir el nombre d'atributs pot disminuir el cost computacional, facilitar el desplegament i fer més interpretable quins factors contribueixen al risc.

Aspectes pràctics: selecció d'atributs en projectes reals *(continuació)*

Aquesta selecció s'ha de fer amb cura per evitar tres problemes habituals:

1. **Fuita d'informació (*data leakage*):** la selecció s'ha de fer només amb dades d'entrenament. Si es miren dades de test, l'avaluació queda contaminada.
2. **Redundància i multicolinealitat:** variables gairebé equivalents, com una temperatura en Celsius i Fahrenheit, poden distorsionar la interpretació.
3. **Inestabilitat:** si petits canvis en les dades alteren molt les variables seleccionades, la selecció no és fiable. Per això convé validar-la amb diferents particions o validació creuada.

3.6.3 Mètodes Integrats (Embedded)

A diferència dels filtres (que són cecs al model) o dels wrappers (que són computacionalment cars), els mètodes integrats realitzen la selecció d'atributs **mentre el model s'està entrenant**:

1. Regularització L1 (Lasso): El "zero" automàtic: la penalització Lasso afegeix un terme a la funció de cost que castiga la magnitud absoluta dels pesos:

$$J(\theta) = \text{MSE}(\theta) + \lambda \sum_{j=1}^{d} |\theta_j| \tag{3.29}$$

La geometria de la norma L1 té forma de diamant, amb vèrtexs situats sobre els eixos de coordenades. Aquesta geometria fa que, en optimitzar el model amb regularització Lasso, sigui habitual que la solució òptima caigui exactament sobre algun d'aquests eixos. Quan això passa, alguns coeficients θ_j esdevenen **exactament zero**.

Per aquest motiu, la regularització L1 no només redueix la magnitud dels coeficients, sinó que també pot actuar com un mecanisme de **selecció d'atributs**: els atributs amb coeficient zero queden eliminats de l'equació final del model.

2. Importància basada en Arbres (Random Forest / XGBoost): Els algorismes basats en arbres de decisió calculen de forma natural quant redueix cada atribut la "impuresa" (Gini o Entropia) de les dades.

Per exemple, en arbres de decisió, cada vegada que un arbre es divideix per un atribut, mirem quant ha millorat la classificació. Sumant totes aquestes millores en milers d'arbres, obtenim un rànquing de rellevància molt robust.

3.6.4 El flux Filtre → Integrat → Wrapper

Quan ens enfrontem a un problema amb milers d'atributs, l'estratègia d'elit consisteix a no usar un sol mètode, sinó a encadenar-los en un *pipeline* d'eficiència creixent (vegeu la Figura 3.16):

1. **Capa de Filtre (Milers → Centenars):** Eliminem constants i variables amb correlació zero. És un procés de mil·lisegons que "neteja" l'espai de soroll obvi.
2. **Capa Integrada (Centenars → Desenes):** Entrenem un model Lasso o un Random Forest ràpid. El propi model ens diu quines variables tenen un pes real en la decisió.
3. **Capa Wrapper (Selecció final):** Amb només 20 o 30 atributs, passem un **RFE** per trobar la combinació exacta que maximitza el rendiment, ignorant ja el cost computacional perquè el volum és petit.

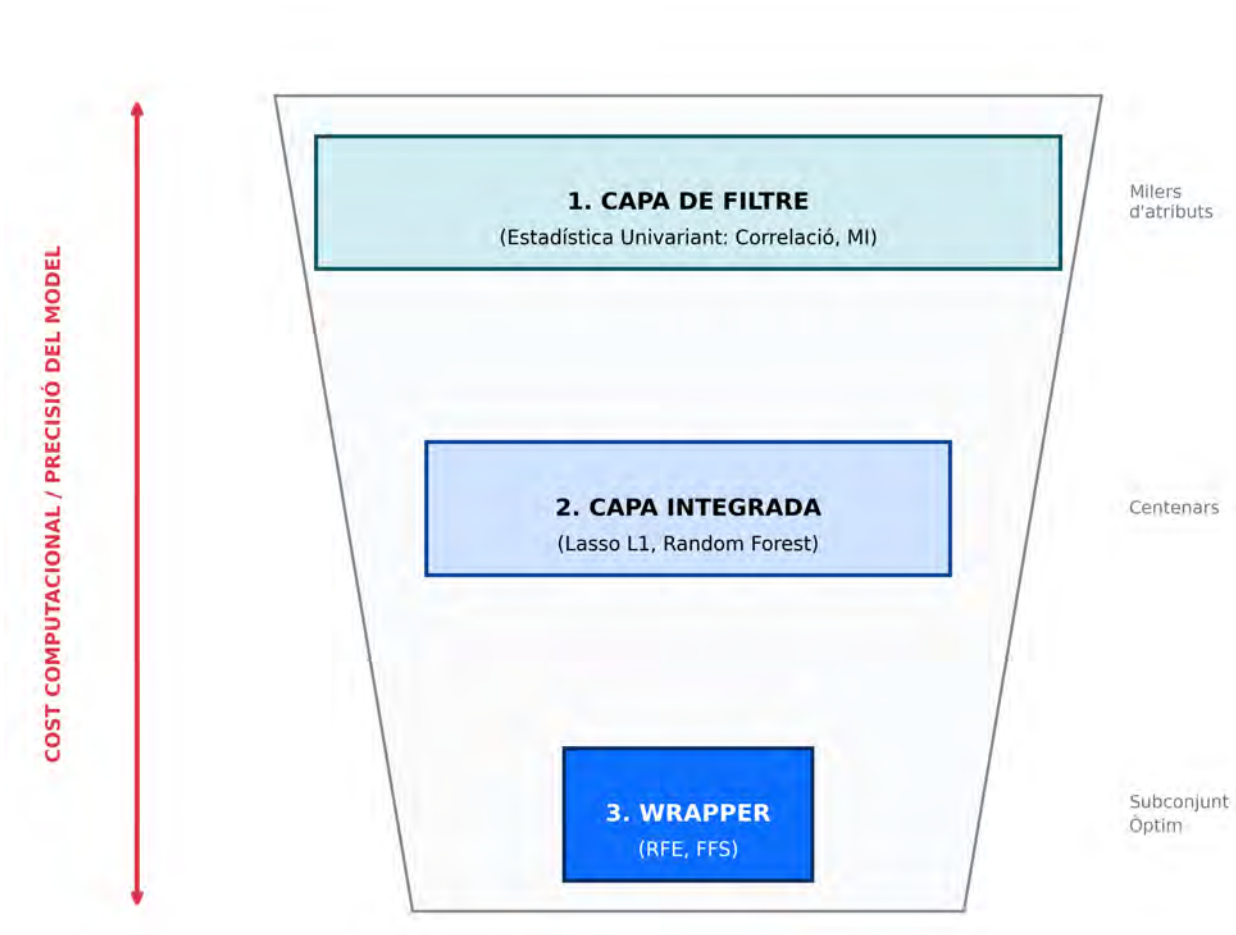


Figura 3.16: El flux 3-2-1: L'embut representa la reducció dràstica de la dimensionalitat mentre s'apliquen mètodes cada vegada més costosos però més precisos.

```python
import matplotlib.pyplot as plt
import matplotlib.patches as patches

fig, ax = plt.subplots(figsize=(10, 6))
ax.set_xlim(0, 10)
ax.set_ylim(0, 10)
ax.axis('off')

# Dibuix de l'embut (funnel)
funnel_x = [2, 8, 7, 3, 2]
funnel_y = [9, 9, 1, 1, 9]
ax.add_patch(patches.Polygon(list(zip(funnel_x, funnel_y)), closed=True,
                             fill=True, color='whitesmoke', edgecolor='gray', alpha=0.3))

# Etapes
# 1. Filtre
ax.add_patch(patches.Rectangle((2.5, 7.5), 5, 1, color='lightblue', alpha=0.8))
ax.text(5, 8, "1. CAPA DE FILTRE\n(Estadística Univariant)", ha='center', va='center',
    fontweight='bold')
ax.text(8.5, 8, "Milers", color='gray', fontsize=12)

# 2. Integrat
ax.add_patch(patches.Rectangle((3.2, 4.5), 3.6, 1, color='skyblue', alpha=0.8))
ax.text(5, 5, "2. CAPA INTEGRADA\n(Lasso / Random Forest)", ha='center', va='center',
```

```
    fontweight='bold')
ax.text(8.5, 5, "Centenars", color='gray', fontsize=12)

# 3. Wrapper
ax.add_patch(patches.Rectangle((4, 1.5), 2, 1, color='deepskyblue', alpha=0.8))
ax.text(5, 2, "3. CAPA WRAPPER\n(RFE / Forward)", ha='center', va='center', fontweight='bold')

ax.text(8.5, 2, "Desenes", color='gray', fontsize=12)

# Fletxa de cost computacional
ax.annotate('Cost computacional / Precisió', xy=(1, 1), xytext=(1, 9),
            arrowprops=dict(arrowstyle='<-', color='red', lw=2))

plt.title("Estratègia d'Enginyeria 3-2-1 per a la Selecció d'Atributs", fontsize=14,
    fontweight='bold')
plt.savefig('fig_selection_pipeline.png', dpi=300)
```

Listing 3.10: El flux 3-2-1

Micro-Cas: L'Enginyeria del Rendiment en Publicitat (AdTech)

En el món de la publicitat online, Google o Meta han de decidir si t'ensenyen un anunci en microsegons. Tenen milions d'atributs (historial, hora, dispositiu, etc.). No poden usar RFE perquè trigarien dies a entrenar. Tampoc poden usar només filtres perquè perdrien les relacions complexes. **La solució d'elit:** Usen mètodes **integrats** (com la regressió logística amb L1). El model aprèn a predir el clic i, al mateix temps, desconnecta els milions d'atributs que no serveixen per a aquella persona usuària concreta. És la combinació perfecta d'entrenament i selecció en un sol pas matemàtic.

Resum de la Selecció d'Atributs

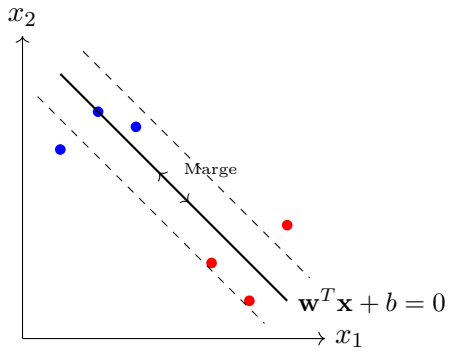
- **Filtres:** Ràpids, però "indivualistes" (no miren interaccions).
- **Wrappers:** Els millors resultats, però computacionalment "vociosos" i lents.
- **Integrats:** L'equilibri perfecte entre cost i rendiment.

Recorda: La millor característica d'un model és la que no necessita per funcionar. Menys atributs és igual a menys soroll, més velocitat i, sobretot, més ètica per la simplicitat.

3.7 Conclusions i Debat

En aquest tercer capítol hem abandonat la regressió de valors continus per entrar en el domini de la decisió discreta. Hem après a traçar fronteres en l'espai $\mathbb{R}^d$ per separar categories, transformant probabilitats en veredictes:

1. **La Frontera de Decisió:** Hem entès que classificar és, geomètricament, dividir l'espai. Ja sigui amb la corba sigmoide de la Regressió Logística o l'hiperplà de les SVM, l'objectiu és trobar el "mur" que millor separa les dades.
2. **Probabilitat vs. Geometria:** Hem comparat dos paradigmes: la Regressió Logística, que ens dóna la certesa de la decisió ($P(y|x)$), i les SVM, que busquen la robustesa física maximitzant el marge de seguretat.
3. **El Kernel Trick:** Mitjançant el Teorema de Mercer, resolem problemes no lineals en 2D projectant-los implícitament a espais d'alta dimensionalitat sense cost computacional afegit.



Síntesi:
1. Funció de Decisió: $\hat{y} = \text{sign}(f(x))$
2. Marge Màxim (SVM)
3. Probabilitat (Logística)
4. No-linealitat (Kernels)

Figura 3.17: El paradigma del Capítol 3: Fronteres, Marges i Espais de Decisió.

4. **Rigor en la Validació:** La corba ROC, l'AUC i la F1-Score són les eines per avaluar sistemes en el món real.

3.7.1 Debat Ètic: La Responsabilitat de la Frontera

Si al Capítol 2 el problema central era l'opacitat de la "caixa negra", al Capítol 3 el dilema clau és la **binarització forçada de la realitat**. Classificar un individu com a "apte" o "no apte", "sa" o "malalt", és un acte reduccionista que col·lapsa la riquesa dels matisos del món en un resultat binari $y \in \{0, 1\}$. Hem de ser conscients que cada hiperplà que tracem és una frontera que atorga o denega drets.

1. El Biaix de l'Hiperplà: La Tirania dels Vectors de Suport

Les SVM busquen el marge màxim basant-se exclusivament en els *support vectors*, els punts més "difícils" o propers a la zona de conflicte. Això genera un **risc sistèmic de fragilitat**: si els vectors de suport que defineixen la frontera de decisió provenen d'un grup social minoritari o històricament discriminat, el "marge de seguretat" del model no reflectirà un risc real, sinó el biaix inherent a la recollida de dades.

Dilema: La neutralitat de la geometria

L'argument de "neutralitat tecnològica" (*"Jo només he maximitzat una distància euclidiana"*) és una fal·làcia en enginyeria social. Si l'espai $\mathbb{R}^d$ està "deformat" per desigualtats estructurals prèvies (com el biaix salarial o la segregació urbana), la geometria més "òptima" des d'un punt de vista matemàtic esdevé una eina de **segregació automatitzada**. L'hiperplà no només descriu la realitat; la solidifica.

2. L'Engany de l'*Accuracy* i el Biaix d'Omissió en Seguretat Crítica

En problemes amb classes altament descompensades (*imbalanced datasets*), com la detecció de càncer o la prevenció d'atacs terroristes, l'*accuracy* és una mètrica profundament immoral. Un model que classifiqui tothom com a "sa" tindrà un èxit del 99.9%, però el seu **cost humà** serà incalculable.

Tenim el deure de resistir la pressió de persones directives o clientes que demanen una única xifra d'èxit. Imposar la corba ROC i la F1-Score no és una tria acadèmica, sinó un imperatiu ètic. Hem d'explicar que un 1% d'error de tipus II (Fals Negatiu) en un diagnòstic és un fracàs absolut del sistema, mentre que un 10% d'error de tipus I (Fals Positiu) pot ser un cost assumible per salvar vides.

3. La "Màgia" del Kernel i el Dret a l'Explicació

El *Kernel Trick* representa el zenit de l'eficiència, però també l'abisme de l'explicabilitat. Quan projectem dades a un espai de dimensió infinita (com el Kernel RBF), la decisió final ja no es basa en variables físiques recognoscibles, sinó en una combinació lineal de distàncies en un espai abstracte (*Reproducing Kernel Hilbert Space*).

- **Conseqüència:** Estem generant **"Veredictes sense Argumentació"**. En àmbits com el dret, l'accés a l'habitatge o la medicina, una decisió que no pot ser explicada en termes de l'espai original (x_1: ingressos, x_2: edat, etc.) és, per definició, una decisió arbitrària que vulnera el dret a la defensa de la persona usuària.
- **La paradoxa del Kernel:** Com més "intel·ligent" i flexible fem la nostra frontera mitjançant kernels complexos, més deshumanitzem el procés de decisió.

3.7.2 Preguntes per a l'Aula

1. Enginyeria: L'Asimetria del Risc i el Punt d'Operació

En el disseny d'un detector d'incendis per a una refineria, ens trobem davant d'un problema de classificació amb costos d'error dràsticament asimètrics:

- **Fals Negatiu (FN):** El model prediu "Absència de foc" quan hi ha una ignició. El cost és catastròfic: pèrdua de vides, destrucció d'actius i desastre ambiental.
- **Fals Positiu (FP):** El model prediu "Foc" per soroll en els sensors. El cost és econòmic: aturada d'emergència de la producció i hores de revisió tècnica.

Pregunta per al debat:

- Si cal triar un punt sobre la corba ROC, sacrificaríeu l'especificitat (acceptant molts Falsos Positius) per garantir una sensibilitat (*Recall*) del 100%?
- Un model amb una **AUC** inferior, però que manté una corba molt "vertical" prop de l'origen (garantint detecció primerenca amb menys soroll), podria ser més segur a la pràctica que un model amb més AUC que només assoleix el $TPR = 1$ a canvi d'un FPR inacceptable.

2. Ètica: La Fragilitat del "Hard Margin" davant l'Atipicitat

L'algorisme de marge dur (*Hard Margin*) busca una perfecció geomètrica que ignora la naturalesa probabilística del món real.

- **Escenari:** Un sistema de triatge de beques d'investigació d'alt rendiment avalua trajectòries acadèmiques en un espai $\mathbb{R}^d$ (notes, publicacions, estades).
- **El Cas Atípic:** Una candidata amb un potencial excepcional presenta una caiguda dràstica de rendiment en un semestre degut a un embaràs. En un model de marge dur, aquest punt esdevé un **vector de suport** que "arrastra" la frontera de decisió, decidint excloure-la per mantenir la separabilitat perfecta.

Pregunta per al debat:

- És acceptable que el disseny d'un algorisme sigui tan rígid que un sol esdeveniment vital desfavorable dictamini el futur d'una persona?
- **Reflexió:** L'ús de *soft margin* no és només una solució tècnica per a dades sorolloses; permet que el model no reorganitzi tota la frontera per satisfer perfectament casos atípics, i força a decidir quin grau d'incompliment del marge és acceptable.

3. Societat: El paràmetre C com a regulador de la severitat

En sistemes de justícia predictiva (com la predicció de reincidència), el paràmetre de regularització C deixa de ser una variable matemàtica per convertir-se en política:

- **C elevat (Rigidesa):** El model penalitza severament qualsevol error d'entrenament. Si les dades històriques indiquen una alta taxa de criminalitat en un barri, la SVM de C alt traçarà un hiperplà inflexible. L'algorisme "creurà" cegament en la dada passada, condemnant el futur.
- **C baix (Permissivitat):** El model prioritza la simplicitat (un marge ample) i accepta que alguns punts caiguin a la banda "equivocada". Això equival a una presumpció d'innocència estadística: preferim un model més simple que s'equivoqui en alguns casos d'entrenament abans que un model hiper-ajustat que penalitzi sistemàticament determinats perfils.

Pregunta per al debat:

- Si un C alt maximitza l'encert històric però perpetua la discriminació, i un C baix és més "just" però menys precís, quina és la configuració "correcta"?
- *Conclusió:* La sintonització d'hiperparàmetres (*Fine-tuning*) és, en realitat, una forma oculta de legislació algorísmica.

4. Visió Crítica: La Dimensionalitat del Kernel i la Pèrdua de la Causa

El *Kernel Trick* ens permet operar en espais de dimensió infinita, però ens allunya de la intuïció humana d'una manera perillosa. Així, mentre que en un model lineal (Regressió Logística), podem extreure els pesos $\mathbf{w}$ i dir: "Aquesta persona ha estat denegada perquè el seu nivell d'endeutament és massa alt", en un Kernel RBF la decisió és una combinació ponderada de distàncies en un espai abstracte no visualitzable.

Pregunta per al debat:

- Estem disposats a cedir el control de decisions vitals (com la llibertat condicional o el diagnòstic mèdic) a un algorisme que no ens pot explicar els seus motius?
- Com podem auditar un model que opera en un espai d'$\mathbb{R}^\infty$ si detectem que està prenent decisions injustes?

Tancament docent

Problemes de consolidació.

1. Donada una sortida lineal $z = \theta^T x$, calcular la probabilitat amb la funció sigmoide, aplicar diferents llindars de decisió i explicar com canvia l'etiqueta final $\hat{y}$.
2. A partir d'una matriu de confusió, calcular accuracy, precision, recall i F1. Després, justificar quina mètrica seria prioritària en un sistema de diagnòstic mèdic, en un filtre de correu brossa i en un detector de frau, i indicar quan convindria mirar també la corba Precision-Recall o la PR-AUC.
3. Comparar dues fronteres de decisió: una regressió logística i una SVM. Explicar què aporta la probabilitat en el primer cas i què aporta el marge geomètric en el segon.
4. Donat un dataset amb variables numèriques i categòriques, proposar un pipeline correcte que inclogui separació train/test, encoding, escalat, selecció d'atributs o PCA, i model final. Identificar en quin punt apareixeria *data leakage* si el pipeline es construís malament.

Tancament docent *(continuació)*

5. Davant d'un problema no lineal, decidir si és més raonable provar una transformació explícita de les dades, una SVM amb kernel o una reducció de dimensionalitat, justificant la decisió segons geometria, cost computacional i interpretabilitat.

Criteris breus de resolució. Una bona resolució hauria de separar clarament tres nivells: puntuació, probabilitat i decisió. En problemes de mètriques, cal identificar primer què representen VP, VN, FP i FN, i després interpretar el cost de cada error en el context del problema. En problemes de SVM, cal explicar el marge i el paper dels vectors de suport, no només dibuixar una recta separadora. En problemes de preprocessament, cal mantenir sempre la disciplina de pipeline: qualsevol transformació que aprengui de les dades s'ha d'ajustar només amb el conjunt d'entrenament.

Connexió amb pràctica/laboratori. Aquest capítol es desplega en una pràctica de classificació i validació. L'alumnat pot construir un pipeline amb regressió logística i SVM, comparar mètriques, variar el llindar de decisió, representar la corba ROC, aplicar *Stratified K-Fold*, provar kernels i comprovar l'efecte de l'escalat, el PCA i la selecció d'atributs sobre el rendiment i la robustesa del model.

Capítol 4

Estructures de Decisió i Raonament Bayesià

L'objectiu d'aquest capítol és dotar l'alumnat de la capacitat de triar entre dos paradigmes fonamentals: la interpretabilitat estructurada dels arbres de decisió i el tractament matemàtic de la incertesa mitjançant el teorema de Bayes. Aprendrem que la intel·ligència no és només classificar, sinó també mesurar quanta "sorpresa" (entropia) conté una dada i com actualitzar les nostres creences davant de noves evidències.

Guia d'activació docent

Repte inicial. Una administració vol automatitzar el triatge de sol·licituds d'ajuda. El sistema ha de decidir, però també ha de poder explicar. La pregunta inicial és: *quan és preferible un model transparent, basat en regles o probabilitats interpretables, encara que no sigui el més sofisticat?*

Checkpoints conceptuals.

1. Explicar com un arbre particiona l'espai i com tria divisions amb entropia i guany d'informació.
2. Aplicar Bayes distingint prior, versemblança, evidència i posterior, i entendre el paper del suavitzat i del log-score.

Errors típics provocats.

1. Fer créixer un arbre fins memoritzar l'entrenament i confondre-ho amb generalització.
2. Aplicar Naive Bayes sense suavitzat i anul·lar una classe per probabilitat zero.

Microcas professional. Filtre de correu brossa: un arbre pot donar regles explícites, mentre que Naive Bayes agrega evidència de moltes paraules per estimar la classe més plausible.

Connexió ètica o d'enginyeria. La interpretabilitat pot ser una condició d'ús, però no garanteix justícia: les regles i probabilitats apreses també poden heretar biaixos de les dades.

4.1 Representació del Coneixement

Fins aquest punt del curs, hem abordat l'aprenentatge automàtic des de dues perspectives: la geomètrica, mesurant distàncies amb k-NN, i la paramètrica, ajustant pesos mitjançant funcions de cost i optimització contínua. En aquest capítol introduïm un tercer paradigma: l'aprenentatge basat en regles i particions, on el model pren decisions mitjançant preguntes successives sobre les dades.

Els Arbres de Decisió (*Decision Trees*) no tracten d'aproximar una funció suau $f(\mathbf{x})$ mitjançant pesos sinàptics o vectors de suport. En el seu lloc, particionen recursivament l'espai de característiques mitjançant una estratègia de *divide et impera*: transformen un problema complex en una jerarquia de preguntes simples sobre les variables.

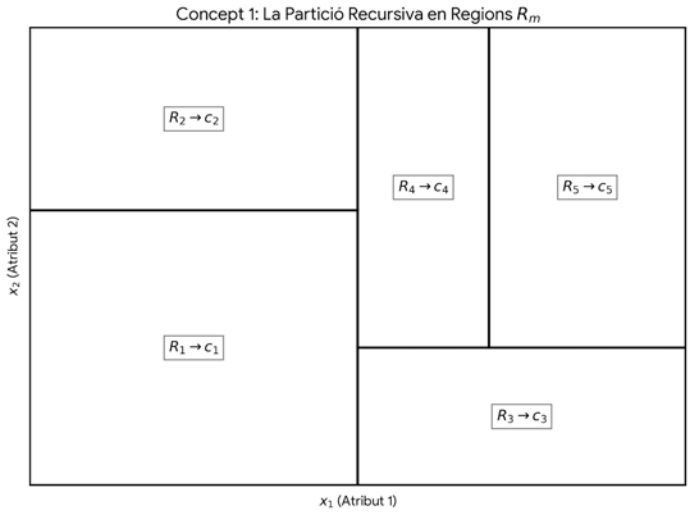


Figura 4.1: La Partició Recursiva de l'Espai. L'algorisme divideix l'espai original en regions rectangulars més petites $(R_1, R_2, \dots)$ mitjançant talls successius. A cada regió final se li assigna un valor constant c_m (predicció).

4.1.1 La Partició Recursiva de l'Espai

Sigui un dataset d'entrenament $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, on $\mathbf{x}_i \in \mathbb{R}^d$ i $y_i \in \{1, \dots, K\}$ (classificació) o $y_i \in \mathbb{R}$ (regressió). Un arbre de decisió construeix una partició de l'espai d'entrada en M regions disjuntes $R_1, R_2, \dots, R_M$. Matemàticament, una manera útil d'entendre l'arbre és veure'l com una funció constant a trossos: cada regió de l'espai rep una predicció associada. En classificació, aquesta predicció pot ser la classe majoritària de la fulla o una distribució empírica de classes.

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} c_m \cdot \mathbb{I}(\mathbf{x} \in R_m) \tag{4.1}$$

On:

- $\mathbb{I}(\cdot)$ és la funció indicadora (retorna 1 si la condició és certa, 0 altrament).
- c_m és la predicció constant per a la regió m.
  - En **regressió**, $c_m = \text{mean}(y_i | \mathbf{x}_i \in R_m)$, la mitjana dels valors de la regió.
  - En **classificació**, $c_m = \text{mode}(y_i | \mathbf{x}_i \in R_m)$, la classe majoritària (votació).

L'objectiu de l'entrenament és trobar la partició òptima que minimitzi una funció de pèrdua (per exemple, $\sum(y_i - \hat{f}(\mathbf{x}_i))^2$). Tanmateix, trobar la partició globalment òptima és un problema **NP-complet**. Per això, en enginyeria utilitzem algorismes *greedy* (voraços) com CART o ID3, que prenen la millor decisió local en cada pas sense garantir l'òptim global.

Explicació Intuïtiva:

Imagineu que teniu un mapa quadrat ple de tresors enterrats i voleu donar instruccions a algú per trobar-los. L'algorisme no pot dibuixar cercles o formes complexes per marcar les zones. L'únic que pot fer és agafar una regla i fer talls rectes de dalt a baix o d'esquerra a dreta.

- Primer fa un tall vertical: "A l'esquerra del riu hi ha pocs tresors, a la dreta n'hi ha molts".

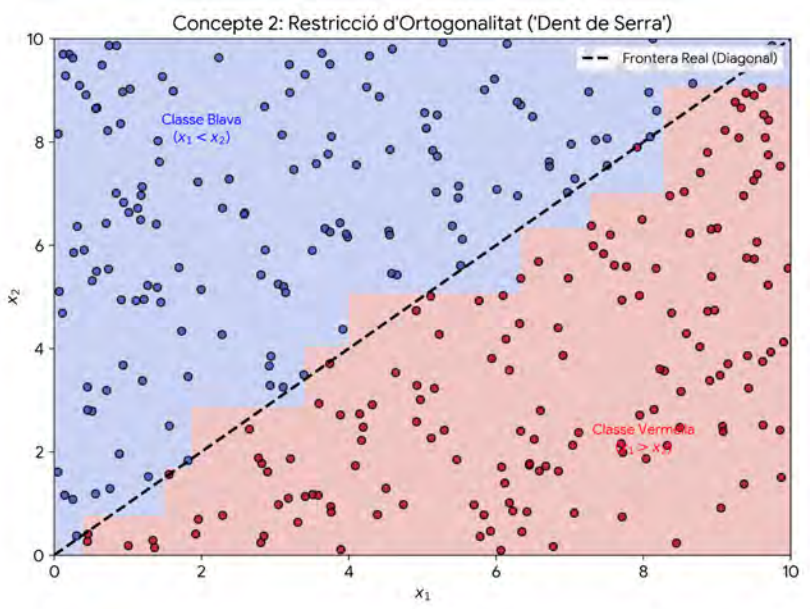


Figura 4.2: L'Efecte "Dent de Serra". Quan la frontera real és diagonal (línia discontínua), l'arbre es veu forçat a crear una estructura en forma d'escala per aproximar-la, ja que només pot fer talls verticals i horitzontals.

Explicació Intuïtiva: *(continuació)*

- Després, agafa el tros de l'esquerra i el torna a tallar horitzontalment: "A la part de dalt no hi ha res, a la de baix sí".

Al final, el mapa queda dividit en "parcel·les" rectangulars (R_1, R_2...). A cada parcel·la li posem una etiqueta (c_m): "Aquí cava" o "Aquí no perdis el temps". Això és exactament el que fa un arbre de decisió amb les vostres dades.

4.1.2 Geometria de la Decisió: La Restricció d'Ortogonalitat

La diferència crítica entre un Arbre de Decisió i models com la Regressió Logística o les SVM rau en la geometria de les seves fronteres.

- Els **Models Lineals** generen hiperplans definits per $\mathbf{w}^T\mathbf{x} + b = 0$. Això permet fronteres obliqües (diagonals) que capturen correlacions entre variables (ex: $x_2 > 2x_1$).
- Els **Arbres de Decisió** clàssics (univariants) generen hiperplans de la forma $x_j \leq t_k$.

Aquesta restricció implica que les regions R_m són sempre **hiper-rectangles** (o hiper-caixes) alineats amb els eixos de coordenades.

L'Efecte "Dent de Serra" (Staircase Effect): Si la frontera de decisió real del problema és diagonal (per exemple, $y = 1 \iff x_1 > x_2$), un arbre de decisió és incapaç de representar-la eficientment. Per aproximar aquesta diagonal, l'arbre haurà de realitzar infinites particions en ziga-zaga:

$$(x_1 > 0.1) \to (x_2 > 0.1) \to (x_1 > 0.2) \to (x_2 > 0.2) \dots$$

Això provoca que l'arbre creixi desmesuradament en profunditat, augmentant la variància i el risc de sobreajust (*overfitting*), tot per intentar dibuixar una simple línia recta de 45 graus.

Explicació Intuïtiva de l'efecte "Dent de Serra"

Si voleu dibuixar una rampa perfecta (una línia diagonal) en una pissarra:

- Un model lineal (com una SVM) agafa el regle, el posa inclinat i traça la línia d'un sol cop. Perfecte i ràpid.
- Un Arbre de Decisió és com aquella joguina clàssica "Etch A Sketch" (la pantalla grisa amb dues rodes). Una roda només mou el llapis amunt/avall i l'altra esquerra/dreta. No pots fer diagonals directament!

Per dibuixar la rampa amb l'"Etch A Sketch", heu de fer: "una mica a la dreta, una mica amunt, una mica a la dreta, una mica amunt...". El resultat no és una línia llisa, sinó una **escala** ("dent de serra"). Això fa que el model sigui molt més complex i "pixelat" del necessari per explicar una cosa tan simple com una línia recta. Per això, si sospiteu que les vostres dades tenen relacions diagonals (ex: pes i alçada estan relacionats), els arbres poden patir si no processeu les dades abans (per exemple, amb PCA).

4.1.3 Estructura de Graf i Interpretabilitat: L'Arbre Invertit

Per entendre un arbre de decisió, no cal pensar en matemàtiques complexes; imagina'l com el joc del "Qui és qui" o un diagrama de flux mèdic. Tot i que tècnicament es defineix com un Graf Acíclic Dirigit (DAG) —que significa que el flux sempre va endavant i mai pots tornar enrere en cercle—, visualment és un arbre capgirat: l'arrel és a dalt i les fulles a baix.

Així, un arbre de decisió és un graf acíclic dirigit organitzat jeràrquicament. L'arrel representa totes les dades abans de la primera partició; els nodes interns són preguntes sobre atributs; i les fulles són les decisions o distribucions de classe finals. Cada predicció correspon al camí que una instància recorre des de l'arrel fins a una fulla. Veiem aquestes tres peces en detall:

1. **Node Arrel (El punt de partida):** Representa tot el grup de dades original abans de fer cap separació. Aquí és on l'algorisme fa la "pregunta reina", la més important de totes per començar a discriminar (per exemple, en un banc: "*La persona clienta té nòmina domiciliada?*").
2. **Nodes de Decisió (Les bifurcacions):** Són els punts intermedis del camí. Cada node és una nova pregunta que s'aplica només al subgrup que ha arribat fins allà (ex: "*Té deutes pendents?*"). Actuen com a vàlvules que dirigeixen el flux de dades cap a una branca o una altra.
3. **Fulles (El veredicte final):** Són els extrems de l'arbre on ja no es fan més preguntes. Si una dada cau en una fulla, l'arbre li assigna una etiqueta definitiva (ex: "*Concedir Crèdit*" o "*Denegar*").

Interpretabilitat: avantatge i límit

La força dels arbres de decisió és que, quan són petits o moderats, permeten resseguir el camí d'una decisió com una cadena de condicions:

$$\text{SI (Nòmina} = \text{Sí) I (Deutes} < 1000) \text{ I (Edat} > 25) \Rightarrow \text{DECISIÓ: Aprovat.}$$

Això facilita l'explicació local d'una predicció. Ara bé, la interpretabilitat no és automàtica: un arbre molt profund, amb centenars de nodes, pot deixar de ser comprensible per a una persona. Per això, en enginyeria, la profunditat, la poda i la claredat de les variables formen part del disseny del model.

El camí des de l'arrel fins a una fulla és simplement una cadena de condicions obligatòries (AND):

$$\text{SI } (\text{Nòmina} = \text{Sí}) \ \mathbf{I} \ (\text{Deutes} < 1000) \ \mathbf{I} \ (\text{Edat} > 25) \implies \text{DECISIÓ: Aprovat}$$

Això és el que anomenem **Interpretabilitat**: la capacitat d'explicar en llenguatge humà per què la màquina ha pres una decisió concreta.

Per entendre la naturalesa "rectangular" de les regions de decisió, visualitzarem la superfície de predicció d'un arbre sobre dades no linealment separables.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.datasets import make_moons

# 1. Generació de dades complexes (No lineals)
# El dataset 'moons' requereix una frontera corba
X, y = make_moons(n_samples=200, noise=0.25, random_state=42)

# 2. Entrenament de l'Arbre
# Limitem la profunditat per evitar un sobreajust total i veure els blocs
clf = DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=42)
clf.fit(X, y)

# 3. Visualització de la Superfície de Decisió (Contour Plot)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))

Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.figure(figsize=(10, 6))
plt.contourf(xx, yy, Z, alpha=0.4, cmap='RdBu') # Regions rectangulars
plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolor='k', cmap='RdBu')
plt.title("Partició de l'Espai per un Arbre (Profunditat=5)")
plt.xlabel("Atribut $x_1$")
plt.ylabel("Atribut $x_2$")
plt.show()
```

Listing 4.1: Anàlisi de l'espai de decisió ortogonal

En `scikit-learn`, el criteri per defecte dels arbres de decisió és `gini`. En aquest capítol utilitzem `criterion='entropy'` per mantenir la coherència amb la discussió sobre entropia i guany d'informació. En la pràctica, ambdós criteris solen produir arbres semblants, però no són idèntics.

Consells de programació: Els arbres no estimen rotacions gratis

Els arbres tallen l'espai amb preguntes alineades amb els eixos: $x_j < t$. Si la frontera real és diagonal, poden necessitar molts talls per aproximar-la. Abans de complicar l'arbre, prova dues coses: visualitza les dades en 2D i valida si una transformació com PCA ajuda dins d'un `Pipeline`. Si PCA millora mètriques però fa perdre interpretabilitat, documenta aquest cost.

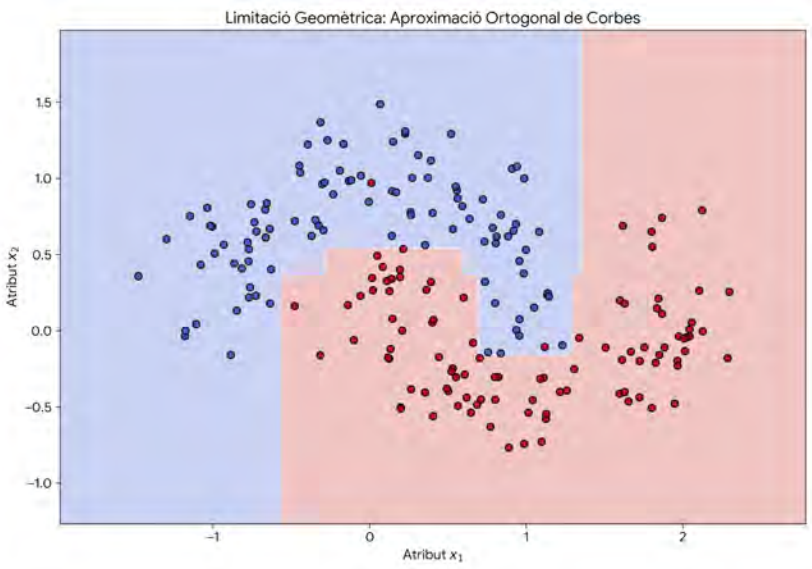


Figura 4.3: Visualització de les regions R_m sobre el dataset 'Moons'. Observeu com l'arbre intenta aproximar la forma de mitja lluna mitjançant una sèrie de rectangles verticals i horitzontals. Aquesta aproximació "pixelada" (efecte aliasing) és característica dels mètodes de partició ortogonal i demostra per què els arbres necessiten molta profunditat per modelar corbes suaus.

Micro-Cas: Risc Financer i Compliance Legal (GDPR)

En el sector bancari, l'algorisme que decideix si es concedeix una hipoteca està sotmès a una regulació estricta. Un model molt complex podria rebutjar un crèdit basant-se en una combinació no lineal de centenars de variables. En un context regulat, això pot ser problemàtic si l'entitat no pot justificar la decisió, auditar les variables utilitzades o garantir que no s'estan emprant proxies de variables protegides. El repte no és només predir bé, sinó poder explicar i defensar el procés de decisió.

El **Reglament General de Protecció de Dades (GDPR)** de la UE estableix el "Dret a l'Explicació". Si un algorisme pren una decisió adversa automàtica, el ciutadà té dret a saber per què. L'Arbre de Decisió és l'eina estàndard en aquests entorns perquè permet generar automàticament una carta de rebuig justificada: *"Benvolgut client, el crèdit ha estat denegat perquè compleix la Regla #42: els seus ingressos anuals són* $< 30.000€$ *I el seu historial d'impagaments en els últims 2 anys és* ≥ 1*."* En enginyeria de sistemes crítics, la **transparència** sovint és una mètrica tan important com l'*accuracy*.

4.2 Teoria de la Informació: L'Entropia

Per construir un arbre de decisió eficient, o qualsevol sistema d'aprenentatge, necessitem una mètrica objectiva per respondre a una pregunta fonamental: *Quant "coneixement" guanyo si descobreixo el valor d'aquest atribut?*

L'any 1948, Claude Shannon, enginyer dels Bell Labs, va establir les bases de l'era digital definint matemàticament el concepte d'informació. La seva idea va ser desvincular el *significat* del missatge de la seva *probabilitat*. Aixi, no importa si un missatge és un poema o una llista de la compra; importa quant "sorprèn".

Per entendre-ho bé: Entropia és desordre abans de preguntar

Si en una capsa hi ha només boles blaves, no hi ha sorpresa: entropia zero. Si hi ha meitat blaves i meitat vermelles, abans de mirar no sabem què sortirà: entropia alta. Un arbre busca preguntes que redueixin aquesta sorpresa. Una bona pregunta deixa els grups fills més "ordenats" que el grup original.

4.2.1 La Quantificació de la Sorpresa

Definim la "sorpresa" o contingut d'informació d'un esdeveniment x com una funció monòtona decreixent de la seva probabilitat $p(x)$. Si un esdeveniment és segur ($p = 1$), la seva ocurrència no aporta cap informació (sorpresa zero). Si és impossible i succeeix, la sorpresa és infinita.

La funció única que satisfà els axiomes d'additivitat per a esdeveniments independents és el logaritme negatiu:

$$I(x) = -\log_b p(x) \tag{4.2}$$

Si usem la base $b = 2$, la unitat resultant és el **bit** (binary digit). L'**Entropia de Shannon** $H(S)$ és l'esperança matemàtica (el valor esperat) d'aquesta sorpresa per a una variable aleatòria discreta S amb K estats possibles:

$$H(S) = \mathbb{E}[I(S)] = -\sum_{i=1}^{K} p_i \log_2 p_i \tag{4.3}$$

On p_i és la probabilitat de pertànyer a la classe i. El resultat representa el nombre mínim mitjà de bits necessaris per codificar el missatge (o classificar la dada).

Per entendre-ho bé: Què fem amb $0 \log 0$?

En la fórmula de l'entropia utilitzem la convenció $0 \log_2(0) = 0$. Això reflecteix que un esdeveniment amb probabilitat zero no contribueix a la incertesa total. En implementació, normalment només sumem sobre les classes que apareixen amb probabilitat positiva.

El Joc del "Qui és Qui"

Imagina que jugues a endevinar un personatge famós.

- Si et dic: "El personatge és humà", t'he donat poca informació. T'ho esperaves (probabilitat alta) i la teva incertesa (Entropia) ha baixat poc.
- Si et dic: "El personatge té tres braços", t'he donat moltíssima informació. Era un fet molt improbable. La teva incertesa s'ha col·lapsat de cop.

L'Entropia mesura quantes preguntes de "Sí/No" (bits) has de fer de mitjana per endevinar la resposta. Si la barreja és 50/50 (màxim desordre), necessites fer moltes preguntes. Si la resposta és òbvia (99/1), amb una pregunta n'hi ha prou. Els arbres de decisió volen trobar la pregunta que redueix més ràpidament aquesta necessitat de seguir preguntant.

4.2.2 Visualització: La Corba de l'Entropia Binària

Per entendre el comportament de l'optimització, analitzem el cas més simple: una moneda trucada (Bernoulli) on la probabilitat de cara és p i la de creu és $1 - p$.

$$H(p) = -[p \log_2 p + (1 - p) \log_2(1 - p)] \tag{4.4}$$

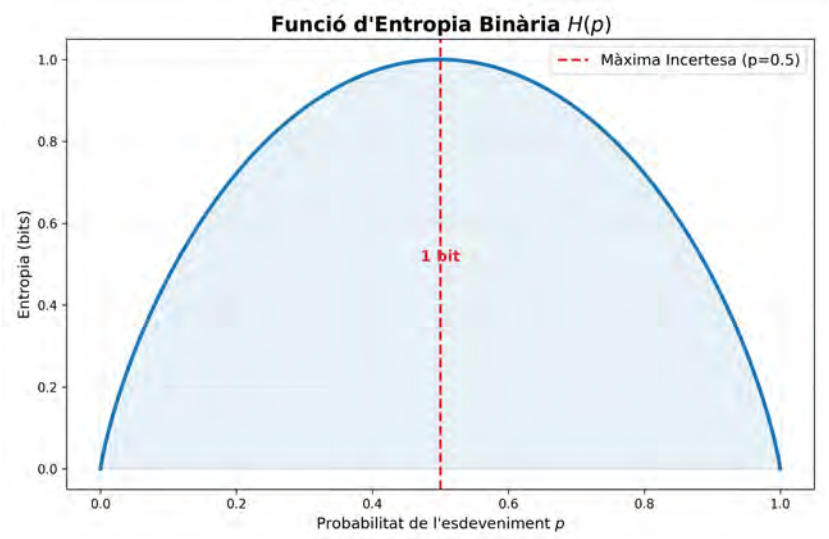


Figura 4.4: La funció d'Entropia Binària. Observeu que la funció és còncava i simètrica. El màxim global s'assoleix quan $p = 0.5$ (màxima incertesa: una moneda justa). Quan el sistema és determinista ($p = 0$ o $p = 1$), l'entropia cau a zero.

```python
import numpy as np
import matplotlib.pyplot as plt

def entropy(y):
    """Calcula l'entropia H(S) d'un vector d'etiquetes."""
    if len(y) == 0:
        return 0.0
    probs = y.value_counts(normalize=True).values
    probs = probs[probs > 0]
    return -np.sum(probs * np.log2(probs))

# Generem l'espai de probabilitats [0, 1]
probs = np.linspace(0.001, 0.999, 100)
ents = [entropy(p) for p in probs]

plt.figure(figsize=(10, 6))
plt.plot(probs, ents, lw=3, color='darkblue')
plt.axvline(x=0.5, color='red', linestyle='--', label='Màxima Incertesa (p=0.5)')
plt.title("Entropia Binària $H(p)$", fontsize=14, fontweight='bold')
plt.xlabel("Probabilitat de l'esdeveniment $p(+)$")
plt.ylabel("Entropia (bits)")
plt.grid(alpha=0.3)
plt.legend()
plt.text(0.5, 0.5, "1 bit", ha='center', va='bottom', fontsize=12, color='red')
plt.savefig('fig_entropy_curve.png', dpi=300)
plt.show()
```

Listing 4.2: Càlcul i Visualització de l'Entropia de Shannon

La corba de la Figura 4.4 explica per què els conjunts de dades desequilibrats (*imbalanced datasets*) són difícils: si tenim un 99% de mostres d'una classe, l'entropia inicial ja és molt baixa ($H \approx 0.08$ bits). El model té "poc a aprendre" perquè endevinar la classe majoritària ja és una estratègia d'alta probabilitat. Per això, una accuracy molt elevada pot amagar que el model pràcticament no detecta la classe minoritària, que sovint és precisament la més important.

Micro-Cas: Enginyeria de Telecomunicacions: El Codi Huffman

L'entropia és un límit físic de l'eficiència. El **Teorema de Codificació de Font** de Shannon demostra que és impossible comprimir una font de dades sense pèrdues per sota de la seva entropia.
Imagina que has de transmetre els resultats d'una biòpsia a un hospital remot amb una connexió satèl·lit molt cara (pagues per bit). Tens dos escenaris possibles:

1. **Escenari A (Incertesa Màxima):** El 50% de les persones pacients tenen la malaltia (+), el 50% no (-).

   $$H(A) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1 \text{ bit per persona pacient.}$$

   Necessites 1 bit per cada diagnòstic (ex: 1=Positiu, 0=Negatiu). No pots comprimir res.

2. **Escenari B (Realitat):** La malaltia és rara. Només el 10% és (+), el 90% és (-).

   $$H(B) = -(0.1 \log_2 0.1 + 0.9 \log_2 0.9) \approx 0.469 \text{ bits per persona pacient.}$$

Conseqüència d'Enginyeria: Encara que tinguis milions de persones pacients, en l'Escenari B pots comprimir les dades més del 50% respecte a l'Escenari A sense perdre informació. Algorismes com **Huffman** o **Lempel-Ziv (ZIP)** aprofiten aquesta baixa entropia per assignar codis curts ("0") als casos freqüents i codis llargs ("1101") als casos rars.

4.2.3 Connexió amb l'Aprenentatge: Del Caos a l'Ordre

En el context dels arbres de decisió, l'entrenament és un procés de reducció d'entropia.

1. Comencem al node arrel amb tot el dataset S. L'entropia $H(S)$ és alta (hi ha una barreja de classes).
2. Busquem una partició (una pregunta) que divideixi S en subconjunts S_1, S_2 tals que la seva entropia mitjana ponderada sigui menor.
3. La diferència entre l'entropia original i la nova s'anomena **Guany d'Informació**:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v) \tag{4.5}$$

L'algorisme està, literalment, "extraient informació" de les dades per reduir la incertesa del sistema fins a arribar a fulles pures ($H = 0$).

La Divergència KL i l'Entropia Creuada

Sovint, l'alumnat es pregunta: *"Per què en Deep Learning usem la funció de pèrdua Cross-Entropy i no l'Accuracy o l'MSE?"* La resposta rau en la **Divergència de Kullback-Leibler (D_{KL})**, que mesura la "distància" entre dues distribucions de probabilitat: la realitat P (les etiquetes reals) i la predicció del model Q.

$$D_{KL}(P||Q) = \sum P(x) \log \frac{P(x)}{Q(x)} = \underbrace{\sum P(x) \log P(x)}_{-H(P)} - \underbrace{\sum P(x) \log Q(x)}_{-\text{CrossEntropy}(P,Q)}$$

La Divergència KL i l'Entropia Creuada *(continuació)*

Com que l'entropia de les dades reals $H(P)$ és una constant fixa (la realitat és la que és), **minimitzar l'Entropia Creuada és matemàticament equivalent a minimitzar la distància KL** entre el que el model creu i la realitat. Per això l'entropia és la reina de les funcions de cost en classificació.

4.3 Construcció de l'Arbre: Guany d'Informació

Un cop entesa l'Entropia com a mesura del caos, *Com ordenem les preguntes per minimitzar aquest caos el més ràpidament possible?* No podem provar tots els arbres possibles; el nombre d'arbres que es poden generar amb d atributs és super-exponencial. Es tracta d'un problema **NP-Complet**. Per fer-ho tractable, Ross Quinlan va introduir l'algorisme **ID3** (Iterative Dichotomiser 3), que utilitza una estratègia *greedy* (voraç/goluda): prendre la millor decisió local en cada node sense preocupar-se per l'optimització global futura.

4.3.1 El Criteri de Selecció: Guany d'Informació (IG)

El nucli de l'ID3 és el **Guany d'Informació** (*Information Gain*). Aquesta mètrica quantifica la reducció esperada de l'entropia després de dividir el conjunt de dades S utilitzant un atribut A. Matemàticament, definim el Guany d'Informació $IG(S, A)$ com la diferència entre l'entropia original i l'entropia condicional posterior a la divisió:

$$IG(S, A) = H(S) - H(S|A) \tag{4.6}$$

On $H(S|A)$ és la suma ponderada de les entropies de cada subconjunt resultant de la divisió. Si l'atribut A té v valors possibles $\{a_1, a_2, \ldots, a_v\}$, particionarem S en subconjunts $S_1, \ldots, S_v$. Llavors:

$$H(S|A) = \sum_{j=1}^{v} \frac{|S_j|}{|S|} H(S_j) \tag{4.7}$$

Substituint, obtenim la fórmula canònica de l'optimització de l'arbre:

$$A^* = \arg\max_{A \in \mathcal{F}} \left(H(S) - \sum_{j=1}^{v} \frac{|S_j|}{|S|} \left(- \sum_{k=1}^{c} p_{jk} \log_2 p_{jk} \right) \right) \tag{4.8}$$

L'atribut A^* serà escollit com el node de decisió actual ("Arrel" del sub-arbre).

El Filtre de Cafè

Imagina que tens una barreja d'aigua bruta (dades amb classes barrejades, alta entropia). Vull passar aquesta aigua per un filtre (atribut) per netejar-la.

- L'**Entropia** $H(S)$ és com de bruta està l'aigua al principi.
- L'**Entropia Condicional** $H(S|A)$ és com de bruta està l'aigua que surt després de passar pel filtre.
- El **Guany d'Informació** IG és la brutícia que s'ha quedat atrapada al filtre.

L'algorisme simplement prova tots els filtres disponibles (colador, filtre de tela, filtre de paper) i tria aquell que atrapa més brutícia en aquell moment.

Per entendre com l'algorisme "tria", implementarem el càlcul manual de l'IG per a un cas de triatge mèdic.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 1. Dataset: Triatge d'Urgències (Simplificat)
data = {
    'Dolor_Toracic': [1, 0, 1, 1, 0, 0, 1, 0],
    'Edat_Major_60': [1, 1, 0, 1, 0, 1, 0, 0],
    'Fumador':       [1, 1, 0, 0, 1, 0, 1, 0],
    'Infart_Real':   [1, 0, 1, 1, 0, 0, 1, 0] # Target (1=Si, 0=No)
}
df = pd.DataFrame(data)

def entropy(y):
    """Calcula l'entropia H(S) d'un vector d'etiquetes."""
    if len(y) == 0:
        return 0.0
    probs = y.value_counts(normalize=True).values
    return -np.sum(probs * np.log2(probs))

def information_gain(df, attribute, target_name):
    # 1. Entropia Pare
    H_parent = entropy(df[target_name])

    # 2. Entropia Fills (Ponderada)
    values = df[attribute].unique()
    H_children = 0
    for v in values:
        subset = df[df[attribute] == v][target_name]
        weight = len(subset) / len(df)
        H_children += weight * entropy(subset)

    # 3. Guany
    return H_parent - H_children

metrics = {}          # Calculem IG per a cada candidat
target = 'Infart_Real'
candidates = ['Dolor_Toracic', 'Edat_Major_60', 'Fumador']

for col in candidates:
    metrics[col] = information_gain(df, col, target)

plt.figure(figsize=(10, 6))      # Visualització
bars = plt.bar(metrics.keys(), metrics.values(), color=['#d62728', '#1f77b4', '#1f77b4'])
plt.title("Rànquing de Preguntes per Guany d'Informació", fontsize=14, fontweight='bold')
plt.ylabel("Guany d'Informació (Bits)")
plt.ylim(0, 1)
plt.axhline(y=0, color='black', linewidth=1)

for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height + 0.02,
             f'{height:.3f} bits', ha='center', va='bottom', fontsize=12)

plt.savefig('fig_info_gain.png', dpi=300)
plt.show()
```

Listing 4.3: Càlcul del Guany d'Informació per a la Selecció d'Arrel

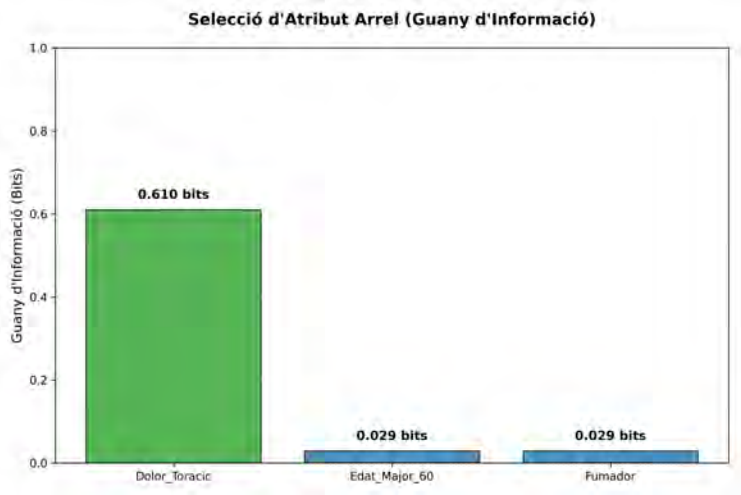


Figura 4.5: Comparació del guany d'informació dels atributs candidats en un dataset simplificat de triatge. En aquest exemple, *Dolor Toràcic* és la variable que més redueix la incertesa sobre la classe objectiu, sent doncs bona candidata a node arrel.

4.3.2 El Problema de l'ID3 i La Solució C4.5

L'algorisme ID3 original té un defecte estructural greu: **premia els atributs amb molts valors únics (alta cardinalitat)**. Imaginem que incloem l'atribut "DNI" o "ID_Pacient" al dataset. Com que cada persona pacient té un DNI únic, si dividim per DNI, obtindrem N subconjunts amb 1 sola persona pacient cadascun.

- Un subconjunt amb 1 element és pur per definició ($H(S_j) = 0$).
- Per tant, $H(S|DNI) = 0$.
- El Guany d'Informació serà màxim: $IG(S, DNI) = H(S)$.

L'arbre triarà el DNI com a arrel. Això genera un arbre inútil que ha memoritzat les dades però no pot generalitzar per a una persona pacient nova (que tindrà un DNI que l'arbre no ha vist mai).

Per solucionar això, l'evolució de l'ID3, anomenada **C4.5**, introdueix una penalització anomenada *Split Information* o **Informació Intrínseca**. Aquesta mesura quantifica l'entropia de la pròpia partició (no de la classe objectiu):

$$SplitInfo_A(S) = -\sum_{j=1}^{v} \frac{|S_j|}{|S|} \log_2 \left(\frac{|S_j|}{|S|} \right) \tag{4.9}$$

Si un atribut té molts valors (com el DNI), dividirà les dades en molts trossets petits, disparant el valor de $SplitInfo$.

Definim la **Ràtio de Guany (Gain Ratio)** com:

$$GainRatio(S, A) = \frac{IG(S, A)}{SplitInfo_A(S)} \tag{4.10}$$

Aquesta mètrica normalitzada evita que l'arbre s'obsessioni amb identificadors únics i afavoreix particions que, a més de pures, siguin estructuralment simples.

La *Gain Ratio* redueix el biaix del guany d'informació cap a atributs d'alta cardinalitat, però no s'ha d'aplicar de manera mecànica. Si la partició generada per un atribut té una *Split*

Information molt petita o nul·la, la ràtio pot esdevenir inestable o no informativa. Per això, implementacions com C4.5 combinen la ràtio amb altres salvaguardes i criteris de selecció.

4.3.3 Tractament de Variables Contínues

Fins ara hem assumit atributs categòrics (Sí/No, Vermell/Verd). Però, com gestiona un arbre variables com "Temperatura = 37.5"? Com que no podem crear una branca per a cada valor de temperatura possible (serien infinits), l'estratègia consisteix a binaritzar la variable contínua cercant el **llindar òptim** t.

1. **Ordenació:** Ordenar les mostres segons el valor de l'atribut continu: $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$.
2. **Candidats:** Considerar els punts mitjos com a possibles llindars de tall: $t_i = \frac{x_{(i)}+x_{(i+1)}}{2}$.
3. **Avaluació:** Per a cada t_i, calcular l'IG de dividir S en $\{x \leq t_i\}$ i $\{x > t_i\}$.
4. **Selecció:** Triar el t^* que maximitza el guany.

Aquest procés té una complexitat de $\mathcal{O}(N \log N)$ degut a l'ordenació, la qual cosa el fa computacionalment més car que els atributs categòrics.

Consells de programació: Els llindars també s'aprenen

Quan un arbre tria el millor llindar per a una variable contínua, està aprenent una decisió a partir de les dades. Per tant, aquest procés s'ha de fer només amb el conjunt d'entrenament dins de cada fold de validació. Si triem llindars utilitzant també les dades de test, introduïm *data leakage* i obtindrem una estimació massa optimista del rendiment.

Discretització i Rendiment

En competicions com Kaggle o en sistemes de trading d'alta freqüència, l'ordenació contínua a cada node és massa lenta. Es recomana pre-processar les dades contínues utilitzant **discretització basada en quantils** (ex: assignar els valors a 10 "cubells" o *bins*). Així, per exemple, convertir "Salari" en "Decil Salarial" (categòric ordinal) permet a l'arbre entrenar-se 10-50 vegades més ràpid amb una pèrdua mínima de precisió.

4.3.4 Quan aturar el creixement?

L'algorisme recursiu necessita condicions de parada (*Base Cases*) per no entrar en bucle infinit o sobreajustar:

1. **Puresa:** Totes les mostres del node pertanyen a la mateixa classe ($H(S) = 0$).
2. **Atributs Esgotats:** No queden atributs per dividir (però encara hi ha barreja de classes). S'assigna la classe majoritària.
3. **Mida Mínima:** El node té menys de k mostres (ex: min_samples_split=20). Això és una forma de pre-poda per evitar regles massa específiques.

Consells de programació: No confiïs en un arbre que acaba en fulles d'una mostra

Abans de mirar l'accuracy d'un arbre, mira quantes mostres hi ha a cada fulla. Si moltes fulles tenen 1 o 2 exemples, probablement estàs memoritzant. En producció, revisa sempre `max_depth`, `min_samples_leaf` i `min_samples_split`. Un arbre més petit i estable acostuma a ser millor que un arbre perfecte en train.

Micro-Cas: Diagnòstic Mèdic: Triatge Intel·ligent

Considerem un arbre de decisió per classificar ictus.

- **Pas 1:** L'algorisme avalua 50 variables. Troba que "Debilitat facial" té l'IG més alt.
- **Pas 2:** Divideix les persones pacients en SI/NO.
- **Pas 3:** Al grup "SI", la variable "Temps des de l'inici" (contínua) és crítica. L'algorisme troba el tall òptim a $t = 4.5$ hores (la finestra per al tractament trombolític).
- **Resultat:** L'arbre no només classifica, sinó que descobreix coneixement mèdic: estableix protocol·làriament que el límit crític són 4.5 hores, una dada que el personal mèdic pot usar en guies clíniques.

4.4 Sobreajust en Arbres i Poda

Els Arbres de Decisió són algorismes no paramètrics amb una capacitat d'aprenentatge teòricament infinita. A diferència d'una regressió lineal, que està restringida a dibuixar una línia recta, un arbre pot continuar dividint l'espai recursivament fins a aïllar cada mostra individual del conjunt d'entrenament en la seva pròpia fulla.

Aquest comportament condueix a un error d'entrenament $\mathcal{E}_{train} = 0$, però provoca un error de generalització $\mathcal{E}_{test}$ catastròfic. Aquest fenomen és el **Sobreajust (Overfitting)**: l'arbre no ha après el patró subjacent, sinó que ha "memoritzat el soroll" estocàstic de les dades.

Analogia: L'Alumnat que Memoritza

Imagineu l'alumnat que es prepara per a un examen d'història.

- **Underfitting (Arbre curt):** només llegeix els títols dels capítols. "Si va passar al segle XX, va ser una guerra mundial". Suspendrà per falta de coneixement.
- **Overfitting (Arbre profund):** memoritza el llibre sencer, incloent les errates i les notes a peu de pàgina. Si l'examen pregunta "Quines causes van portar a la guerra?", la persona estudiant respon recitant la pàgina 45 paraula per paraula. No ha entès la història, només l'ha memoritzat.
- **Arbre Òptim:** entén els conceptes clau i descarta detalls irrellevants (soroll), permetent-li respondre a preguntes que no ha vist mai.

4.4.1 Fonaments Matemàtics: El Compromís Biaix-Variança

L'error esperat d'un model $\hat{f}(x)$ es pot descompondre en tres termes:

$$E[(y - \hat{f}(x))^2] = \text{Biaix}^2 + \text{Variança} + \sigma^2 \text{ (Soroll Irreduïble)} \tag{4.11}$$

En el context dels arbres:

- Arbres Petits (Pocs nodes) tenen **Alt Biaix** (Underfitting): estructura massa simple per la complexitat de les dades, amb fronteres de decisió massa rígides.
- Arbres Grans (Molts nodes) tenen **Alta Variança** (Overfitting): estructura canvia radicalment si modifiquem lleugerament el dataset d'entrenament. El model captura fluctuacions aleatòries.

L'objectiu és trobar la complexitat òptima de l'arbre C^* que minimitza l'error total, aplicant **Pre-poda** (aturar el creixement) i **Post-poda** (retallar l'arbre a posteriori).

4.4.2 Estratègia 1: Pre-Poda (Early Stopping)

La pre-poda consisteix a imposar restriccions geomètriques durant la fase de construcció de l'arbre. Aquests són els hiperparàmetres crítics que s'han de calibrar:

1. Profunditat Màxima (`max_depth`)

Limita el nombre màxim de nivells des de l'arrel fins a la fulla més llunyana.

$$\text{Profunditat}(T) \leq K$$

Restringir la profunditat evita que l'arbre faci preguntes massa específiques, però és una mesura grollera que tracta totes les branques per igual.

2. Mínim de Mostres per Fulla (`min_samples_leaf`)

Aquest és, segons el consens del criteri expert, l'hiperparàmetre més robust. Exigeix que qualsevol node terminal contingui almenys k mostres.

$$|R_m| \geq k \tag{4.12}$$

Geomètricament, això actua com un **suavitzat** de la frontera de decisió. Impedeix que l'arbre aïlli *outliers* (punts anòmals solitaris), forçant-lo a prendre decisions basades en la majoria local d'un grup representatiu.

El Secret del 1%: L'Efecte Horitzó

La pre-poda pateix l'**Efecte Horitzó**: l'algorisme *greedy* pot aturar-se en un node perquè la divisió actual no sembla prometedora (baix Guany d'Informació), sense saber que aquesta divisió hauria permès una divisió excel·lent en el següent nivell. *Consell d'experiència:* Si teniu recursos computacionals, preferiu sempre la Post-poda a la Pre-poda agressiva, ja que la Post-poda avalua l'estructura completa.

4.4.3 Estratègia 2: Post-Poda (Cost-Complexity Pruning)

La Post-Poda permet que l'arbre creixi completament (T_{max}) i després elimina les branques que no justifiquen la seva existència en termes de rendiment vs. complexitat. L'algorisme estàndard en la indústria (implementat en CART i scikit-learn) és el **Minimal Cost-Complexity Pruning**.

Definim la funció de cost d'un arbre T com:

$$R_\alpha(T) = R(T) + \alpha|T| \tag{4.13}$$

On:

- $R(T)$ és l'error total d'entrenament (taxa de mala classificació o MSE).
- $|T|$ és el nombre de nodes terminals (fulles), que representa la complexitat.
- $\alpha \geq 0$ és el paràmetre de complexitat (similar al λ en regularització Lasso).

L'Algorisme de la Baula Més Feble (Weakest Link)

Per a cada valor d'α, existeix un sub-arbre òptim $T_\alpha \subseteq T_{max}$. Per trobar-lo: 1. Comencem amb l'arbre complet T_{max}. 2. Per a cada node intern t, calculem l'**efectivitat del tall**:

$$g(t) = \frac{R(t) - R(T_t)}{|T_t| - 1} \tag{4.14}$$

On $R(t)$ és l'error si podem el node (el convertim en fulla) i $R(T_t)$ és l'error si mantenim la branca. 3. Podem el node amb el $g(t)$ més baix (la baula més feble). 4. Repetim fins a quedar-nos només amb l'arrel.

Això genera una seqüència d'arbres niats. L'equip d'enginyeria utilitza **Validació Creuada** per seleccionar l'α òptim que minimitza l'error de validació.

Interacció d'Hiperparàmetres

Un error comú és ajustar `max_depth` i ignorar la resta. Però els hiperparàmetres interactuen:en lloc de limitar dràsticament la profunditat, sovint és millor deixar l'arbre profund però augmentar `min_samples_leaf` (a 10 o 20). Això permet a l'arbre modelar relacions complexes però assegura que cada decisió estigui recolzada per una evidència estadística significativa (mínim 20 mostres), suavitzant les fronteres i reduint la variança sense perdre tanta flexibilitat.

4.4.4 Corbes de Validació i el Punt d'Inflexió

Per il·lustrar l'overfitting i l'efecte de la profunditat, generarem corbes d'aprenentatge.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import validation_curve
from sklearn.datasets import make_classification

# 1. Generació de dades sintètiques complexes i sorolloses
# 1000 mostres, 20 característiques, flip_y introdueix soroll d'etiqueta (vital per veure overfitting)
X, y = make_classification(n_samples=1000, n_features=20, n_informative=10,
                           n_classes=2, flip_y=0.2, random_state=42)

# 2. Definir el rang de l'hiperparàmetre (Profunditat de l'arbre)
param_range = np.arange(1, 21)

# 3. Calcular Corbes de Validació (5-Fold Cross-Validation)
train_scores, test_scores = validation_curve(
    DecisionTreeClassifier(random_state=42),
    X, y, param_name="max_depth", param_range=param_range,
    cv=5, scoring="accuracy", n_jobs=-1
)

# Càlcul de mitjanes i desviacions estàndard
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# 4. Plotting Professional
plt.figure(figsize=(10, 6))
```

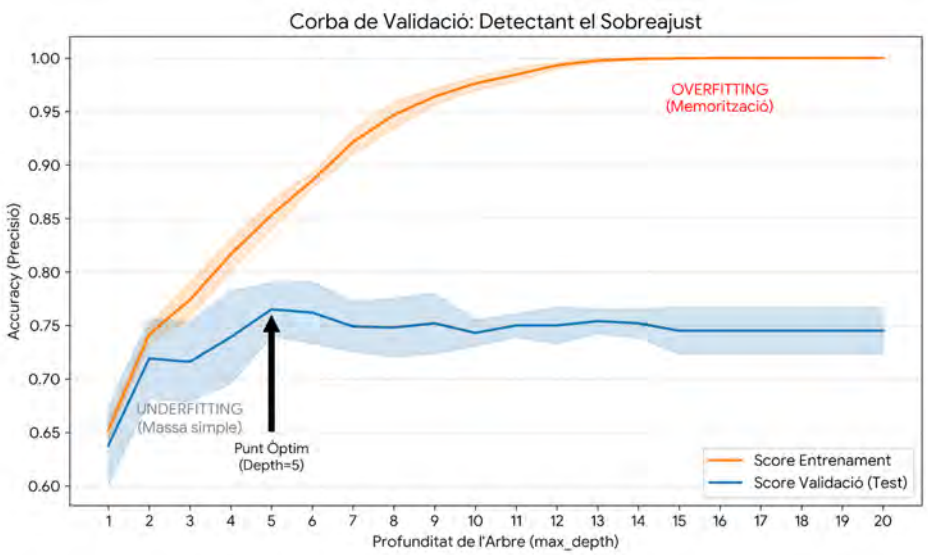


Figura 4.6: Corba de Validació. Observeu com, a partir de profunditat 6 (Punt Òptim), el rendiment en entrenament (taronja) segueix millorant cap a l'1.0, mentre que el rendiment en validació (blau) s'estanca i comença a degradar-se. La divergència entre les dues corbes defineix la **Variança** (Overfitting).

```
plt.plot(param_range, train_mean, label="Score Entrenament", color="darkorange", lw=2)
plt.fill_between(param_range, train_mean - train_std, train_mean + train_std, alpha=0.2,
    color="darkorange")

plt.plot(param_range, test_mean, label="Score Validació Creuada", color="navy", lw=2)
plt.fill_between(param_range, test_mean - test_std, test_mean + test_std, alpha=0.2, color="
    navy")

# Decoració i Anotacions
plt.title("Corba de Validació: L'Equilibri Biaix-Variança", fontsize=14, fontweight='bold')
plt.xlabel("Profunditat Màxima de l'Arbre", fontsize=12)
plt.ylabel("Accuracy", fontsize=12)
plt.legend(loc="best")
plt.grid(ls='--', alpha=0.7)

# Fletxa indicant el punt òptim
opt_depth = param_range[np.argmax(test_mean)]
plt.annotate(f'Punt Òptim (Depth={opt_depth})',
             xy=(opt_depth, test_mean[opt_depth-1]),
             xytext=(opt_depth+2, test_mean[opt_depth-1]-0.05),
             arrowprops=dict(facecolor='black', shrink=0.05))

plt.text(15, 0.95, "Zona d'Overfitting\n(Train puja, Test baixa)", color='red', ha='center')

plt.tight_layout()
plt.savefig('fig_validation_curve.png', dpi=300)
plt.show()
```

Listing 4.4: Generació de Corbes de Validació per detectar Overfitting

Micro-Cas: Motors Elèctrics i Soroll de Sensors

En una planta de producció, es monitoritzen motors mitjançant sensors de vibració. Les dades contenen pics de soroll elèctric que no indiquen fallada. Un arbre sense poda va generar la regla: *"SI Vibració > 50Hz I Hora == 14:03:01 LLAVORS ALARMA"*. Aquesta regla va capturar un pic de soroll específic que va passar un dia concret. És una regla inútil ("memoritza soroll").
L'equip d'enginyeria va aplicar **Post-Poda amb** $\alpha = 0.01$. La branca de l'hora va ser tallada perquè augmentava la complexitat sense reduir l'error global significativament. L'arbre podat va resultar en: *"SI Vibració > 52Hz LLAVORS ALARMA"*. Aquesta regla és lleugerament menys precisa en el dataset d'entrenament (es perd el cas de les 14:03), però és **robusta i generalitzable** a tots els motors i horaris.

4.5 El Teorema de Bayes

Fins ara, hem treballat sota un paradigma **lògic i determinista**. En un Arbre de Decisió, si una mostra compleix $x_{edat} > 18$ i $x_{ingressos} > 20k$, cau inequívocament en una fulla específica. L'algorisme no dubta.

Tanmateix, l'enginyeria del món real és inherentment sorollosa. Els sensors fallen, les dades mèdiques són incompletes i les persones usuàries es comporten de manera impredictible. Quan la certesa absoluta és impossible, l'equip d'enginyeria ha de modelar la **incertesa**. És a dir, en lloc de preguntar "quina regla s'aplica?", preguntarem "com canvia la probabilitat d'una hipòtesi quan observem una nova evidència?".

Aquest és el pas natural cap al raonament bayesià; aquí és on passem de la lògica booleana (0/1) a la lògica Bayesiana ($p \in [0, 1]$). El Teorema de Bayes no és només una fórmula; és un mecanisme per **actualitzar les nostres creences** a mesura que rebem noves dades (evidències). Si l'Arbre de Decisió és un jutge que dicta sentència, el Teorema de Bayes és un detectiu que ajusta les seves sospites amb cada nova pista.

4.5.1 Teoria Matemàtica: De la Freqüència a la Creença

Per entendre Bayes, primer hem de revisar dos conceptes fonamentals que sovint es confonen: la probabilitat conjunta i la condicional.

1. Probabilitat Condicionada

La probabilitat de que passi l'esdeveniment A donat que sabem que ha passat B es defineix com:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{4.15}$$

Això es pot interpretar geomètricament: en saber que ha passat B, el nostre "univers de possibilitats" es redueix de Ω a l'àrea de B. Ara només ens importa quina part de A cau dins de B.

2. La Regla del Producte i la Derivació de Bayes

De la definició anterior, podem aïllar la probabilitat conjunta $P(A \cap B)$:

$$P(A \cap B) = P(A|B)P(B) \tag{4.16}$$

Per simetria, també és cert que:

$$P(A \cap B) = P(B|A)P(A) \tag{4.17}$$

Igualant les dues expressions, obtenim la pedra angular de la inferència estadística:

$$P(A|B)P(B) = P(B|A)P(A) \implies P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4.18}$$

4.5.2 Prior, Likelihood i Posterior

En Aprenentatge Automàtic, reescrivim l'equació anterior substituint A per la **Hipòtesi** (H, el que volem saber, ex: "És un atac?") i B per l'**Evidència** (E, les dades que tenim, ex: "Hi ha molts paquets IP").

$$\underbrace{P(H|E)}_{\text{Posterior}} = \frac{\overbrace{P(E|H)}^{\text{Likelihood}} \cdot \overbrace{P(H)}^{\text{Prior}}}{\underbrace{P(E)}_{\text{Evidence}}} \tag{4.19}$$

Analitzem cada terme amb mentalitat d'enginyeria:

- **Prior $P(H)$ (Coneixement previ):** Quina probabilitat assignem a la hipòtesi *abans* de veure cap dada? (Ex: "Els atacs són rars, així que la prior és baixa, 0.01"). Ignorar la Prior és la font principal d'errors en sistemes d'IA.
- **Likelihood $P(E|H)$ (Versemblança):** Si la hipòtesi fos certa, com de probable seria veure aquestes dades? (Ex: "Si és un atac, és molt probable (0.99) veure aquest trànsit"). Aquest terme surt de l'entrenament del model.
- **Posterior $P(H|E)$ (Creença actualitzada):** La nostra nova probabilitat després de processar la dada. Aquesta serà la sortida del nostre classificador.
- **Evidence $P(E)$ (Normalització):** La probabilitat de veure les dades sota *qualsevol* hipòtesi. Garanteix que les probabilitats sumin 1.

El teorema de Bayes ens dona una probabilitat posterior, però la decisió final depèn del cost de cada error. En un diagnòstic mèdic, una probabilitat moderada pot justificar una prova addicional; en un filtre de correu, potser cal un llindar molt alt abans de bloquejar un missatge legítim. Per tant, Bayes actualitza creences, però l'enginyeria ha de definir la regla de decisió.

La bellesa de Bayes és que permet l'aprenentatge seqüencial. La **Posterior** d'avui es converteix en la **Prior** de demà. Si tenim una seqüència de dades $d_1, d_2, \ldots, d_n$, no cal recalcular-ho tot des de zero:

$$P(H|d_1, \ldots, d_n) \propto P(d_n|H) \cdot \underbrace{P(H|d_1, \ldots, d_{n-1})}_{\text{Prior actualitzada}} \tag{4.20}$$

Per entendre-ho bé: Bayes és actualitzar una creença

El *prior* és el que creiem abans de veure la nova evidència. La *likelihood* diu com de compatible és aquesta evidència amb cada hipòtesi. El *posterior* és la creença actualitzada després de veure l'evidència. Bayes no inventa informació: recombina el que crèiem abans amb el que acabem d'observar.

El cas del paraigua

Imagineu que viviu en un búnquer sense finestres i voleu saber si plou (H).

- **Prior** $P(H)$**:** Sóc a Londres o a Almeria? Si sóc a Londres, la meva creença inicial de pluja és alta (0.5). Si sóc a Almeria, és baixa (0.05). Això és la Prior.
- **Evidència** E**:** Veig entrar al cap un company amb un paraigua moll.
- **Likelihood** $P(E|H)$**:** Si plou, és molt probable que porti paraigua (0.9).
- **Posterior** $P(H|E)$**:** Com que la probabilitat de portar paraigua si fa sol és molt baixa, la visió del paraigua (Evidència) dispara la meva creença que plou, modificant la meva Prior inicial.

Bayes és simplement la matemàtica de "canviar d'opinió quan veus coses noves".

El terme més delicat de l'equació de Bayes és el denominador $P(E)$, anomenat *evidència* o *versemblança marginal*. Aquest terme normalitza la probabilitat posterior i mesura com de probable és observar l'evidència E tenint en compte totes les hipòtesis possibles:

$$P(E) = \sum_i P(E \mid H_i)P(H_i). \tag{4.21}$$

Així, la posterior no depèn només de com de compatible és l'evidència amb una hipòtesi, sinó també de la probabilitat inicial d'aquesta hipòtesi:

$$P(H_k \mid E) = \frac{P(E \mid H_k)P(H_k)}{\sum_i P(E \mid H_i)P(H_i)}. \tag{4.22}$$

En problemes simples, com "atac" vs. "no atac", el denominador és una suma de dos termes. En models amb moltes hipòtesis o paràmetres continus, però, aquesta suma esdevé una integral d'alta dimensió i el càlcul exacte pot ser intractable.

Ignorar el Denominador

En la pràctica, no ens importa el valor absolut de la probabilitat, sinó saber quina hipòtesi és la més probable. Com que $P(E)$ és el mateix per a totes les hipòtesis (és una constant de normalització), s'ignora durant l'optimització:

$$\text{ArgMax}_H P(H|E) \propto \text{ArgMax}_H [P(E|H) \cdot P(H)]$$

Transformem el problema en: **Posterior** $\propto$ **Likelihood** $\times$ **Prior**. Ens estalviem un càlcul costós sense afectar la decisió final.

El següent codi mostra la Probabilitat Posterior en funció de la Prior i la Likelihood (Sensibilitat del test).

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def bayes_theorem(prior, sensitivity, specificity=0.95):
    """
    Calcula P(H|E) donats P(H), P(E|H) i P(no E| no H).
    sensitivity = P(E|H)  (True Positive Rate)
    specificity = P(no E| no H) (True Negative Rate)
    """
    # P(E|no H) = 1 - specificity
```

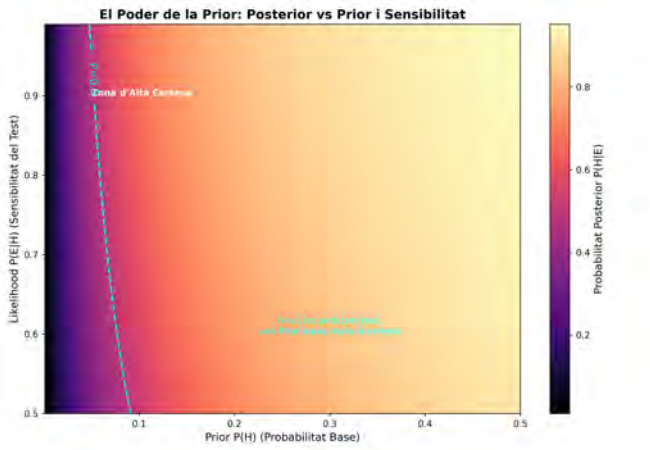


Figura 4.7: Mapa de calor de la Probabilitat Posterior. A la zona inferior esquerra, fins i tot amb un test bo (Sensibilitat 0.8), si l'esdeveniment és molt rar (Prior < 0.05), la probabilitat posterior no supera el 50%. Això demostra que **per afirmar coses extraordinàries (rares), calen evidències extraordinàries (tests perfectes)**.

```
    false_positive_rate = 1 - specificity

    # Likelihood * Prior
    numerator = sensitivity * prior

    # Evidence P(E) = P(E|H)P(H) + P(E|no H)P(no H)
    denominator = (sensitivity * prior) + (false_positive_rate * (1 - prior))

    return numerator / denominator

# Creem un espai de valors
priors = np.linspace(0.001, 0.50, 100) # De 0.1% a 50%
sensitivities = np.linspace(0.5, 0.99, 100) # Test dolent a perfecte

# Matriu de resultats
posterior_map = np.zeros((100, 100))
for i, sens in enumerate(sensitivities):
    for j, pri in enumerate(priors):
        posterior_map[i, j] = bayes_theorem(pri, sens)

# Plotting
plt.figure(figsize=(10, 8))
# Usem extent per mapar els indexs als valors reals als eixos
plt.imshow(posterior_map, extent=[0.001, 0.5, 0.5, 0.99], origin='lower',
           aspect='auto', cmap='viridis')
plt.colorbar(label='Probabilitat Posterior $P(H|E)$')
plt.title("Impacte de la Prior vs. Qualitat del Test (Likelihood)", fontsize=14, fontweight='
    bold')
plt.xlabel("Prior $P(H)$ (Probabilitat Base de l'esdeveniment)", fontsize=12)
plt.ylabel("Likelihood $P(E|H)$ (Sensibilitat del Test)", fontsize=12)

# Línia de tall del 50% de confiança
plt.contour(priors, sensitivities, posterior_map, levels=[0.5], colors='white', linestyles='--
    ')
plt.text(0.1, 0.6, 'Zona de Dubte ($P < 0.5$)', color='white', fontsize=12, fontweight='bold')

plt.text(0.3, 0.9, 'Zona de Certesa ($P > 0.5$)', color='white', fontsize=12, fontweight='
```

```
    bold')

plt.tight_layout()
plt.savefig('fig_bayes_heatmap.png', dpi=300)
plt.show()
```

Listing 4.5: Codi per analitzar l'impacte de la Prior en la Posterior

Micro-Cas: El Detector d'Intrusions "Perfecte"

Escenari: Una empresa processa 1 milió de connexions de xarxa al dia. Els atacs reals són extremadament rars: només 10 al dia ($P(H) = 10^{-5}$). L'empresa compra un software AI amb una precisió del 99%:

- Si és un atac, el detecta el 99% dels cops ($P(E|H) = 0.99$).
- Si és trànsit legal, només s'equivoca l'1% ($P(E|\neg H) = 0.01$).

Quina és la probabilitat que sigui un atac real?:

$$\begin{aligned} P(H|E) &= \frac{0.99 \cdot 0.00001}{(0.99 \cdot 0.00001) + (0.01 \cdot 0.99999)} \\ &\approx \frac{0.0000099}{0.0000099 + 0.0099999} \approx \mathbf{0.00098} \end{aligned} \tag{4.23}$$

Resultat: Fins i tot amb l'alarma sonant, la probabilitat que sigui un atac real és **menor al 0.1%**. Però l'1% d'error sobre el milió de connexions legals genera 10.000 falsos positius, que ofeguen completament els 10 atacs reals.

4.6 El Classificador Naive Bayes: Independència

Fins ara hem vist models que intenten trobar la frontera de decisió directament (com les SVM o els Arbres). Naive Bayes representa un canvi de paradigma cap als **Models Generatius**. En lloc de preguntar-se "Quina forma té la frontera?", es pregunta: "Com són les dades que genera cada classe?".

Aquest enfocament és especialment poderós en dominis d'alta dimensionalitat, com el Processament del Llenguatge Natural (NLP), on el nombre de característiques d (paraules al vocabulari) pot superar fàcilment les 100.000. En aquests espais, la "maledicció de la dimensionalitat" fa que mètodes com k-NN siguin inútils (tots els punts estan lluny). Naive Bayes, mitjançant una simplificació radical, aconsegueix no només sobreviure sinó excel·lir.

L'objectiu de Naive Bayes és calcular la probabilitat posterior $P(y|\mathbf{x})$ per triar la classe més probable (MAP - Maximum A Posteriori):

$$\hat{y} = \arg \max_{y \in \{1,\dots,K\}} P(y|x_1, \dots, x_d) \tag{4.24}$$

Aplicant el Teorema de Bayes:

$$P(y|\mathbf{x}) = \frac{P(y)P(x_1, \dots, x_d|y)}{P(\mathbf{x})} \tag{4.25}$$

El terme problemàtic és la **Versemblança Conjunta** $P(x_1, \dots, x_d|y)$. Per estimar aquesta distribució sense cap assumpció, necessitaríem un nombre de mostres exponencial respecte a d. Si cada x_i és binari, hi ha 2^d combinacions possibles. Amb $d = 1000$, 2^{1000} és superior al nombre d'àtoms de l'univers visible. No tenim prou dades ni memòria per modelar això.

L'algorisme Naive Bayes assumeix que, **condicionat a la classe** y, els atributs són independents entre si. Per exemple, en un classificador de correus, saber que apareix la paraula "promoció" no ens dona informació addicional sobre si apareixerà la paraula "descompte", *un cop ja sabem que el correu és publicitari*. Matemàticament, la distribució conjunta es descompon:

$$P(x_1, \dots, x_d|y) \approx \prod_{i=1}^{d} P(x_i|y) \tag{4.26}$$

Aquesta simplificació redueix el nombre de paràmetres a estimar de 2^d a $d \times K$. Hem passat d'un problema impossible a un de lineal. Segons la naturalesa de les dades $\mathbf{x}$, triem diferents distribucions de probabilitat per modelar la versemblança:

1. **Gaussian Naive Bayes:** Per a dades contínues (ex: temperatura, alçada). Assumim que cada característica segueix una distribució Normal:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_{y,i}^2}} \exp\left(-\frac{(x_i - \mu_{y,i})^2}{2\sigma_{y,i}^2}\right) \tag{4.27}$$

2. **Multinomial Naive Bayes:** L'estàndard en classificació de text. Els atributs x_i representen **recomptes** (quantes vegades apareix la paraula w_i al document).

3. **Bernoulli Naive Bayes:** Per a dades binàries. Només ens importa si la paraula apareix (1) o no (0), ignorant la freqüència.

Per entendre-ho bé: Per què és "naive"?

Naive Bayes fa veure que les característiques són independents dins de cada classe. En text, això voldria dir tractar paraules com si aportessin evidència per separat. Aquesta hipòtesi d'independència de Naive Bayes no afirma que les variables siguin realment independents al món real. És una simplificació de model que fa el càlcul tractable i sovint funciona sorprenentment bé, especialment en text i alta dimensió.

El Retrat Robot

Imagina't fent un retrat robot. Tens testimonis que han vist el sospitós.

- **Enfocament Realista (Complex):** Intentaries dibuixar la cara sencera de cop. "Tenia el nas gran i els ulls petits, però potser si el nas és gran els ulls semblen més petits...". Les combinacions són infinites.
- **Enfocament Naive Bayes:** Dibuixes el nas, els ulls i la boca per separat. Assumeixes que tenir el "Nas Gran" no té res a veure amb tenir "Orelles Petites".

És cert això? No. A la genètica real, els trets estan correlacionats. El retrat robot resultant potser no és 100% realista (pot semblar un Frankenstein), però és **suficientment bo** per distingir si el sospitós és l'acusat A o l'acusat B. L'enginyeria accepta la imperfecció a canvi de l'eficiència.

4.6.1 Quan utilitzar Naive Bayes?

Tot i ser "ingenu", aquest model segueix sent l'estat de l'art en situacions específiques:

- **Cold Start:** Quan tenim moltíssimes característiques ($d > 10.000$) però molt poques dades d'entrenament ($N < 1000$). Naive Bayes té una variança molt baixa (difícil de sobreajustar) comparat amb una Xarxa Neuronal.

- **Latència Ultra-Baixa:** En sistemes de licitació d'anuncis en temps real (RTB), on cal classificar en microsegons. La suma de logaritmes és imbatible en velocitat de CPU.
- **Baseline:** Sempre hauria de ser el primer model que proveu. Si una xarxa neuronal complexa no supera significativament al Naive Bayes, probablement no val la pena el cost computacional extra.

Per entendre què ha après realment el model, visualitzarem la matriu de versemblances $P(x_i|y)$ per a un problema de classificació de text simple.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Definició del Vocabulari i Classes
vocab = ["pilota", "govern", "gol", "llei", "partit", "eleccions", "jugador", "vot"]
classes = ["Esports", "Política"]

# 2. Matriu de Likelihoods P(Paraula | Classe)
# Generem dades sintètiques que representen el coneixement après
# Fila 0: Esports, Fila 1: Política
# Noteu la paraula "partit": és polisèmica (apareix en ambdós contextos)
raw_counts = np.array([
    [15,  1, 20,  0, 10,  1, 25,  1], # Esports
    [ 1, 18,  1, 15, 12, 20,  1, 18]  # Política
])

# Apliquem Laplace Smoothing (alpha=1)
alpha = 1
vocab_size = len(vocab)
class_totals = raw_counts.sum(axis=1, keepdims=True)

# Fórmula de Laplace
probs = (raw_counts + alpha) / (class_totals + alpha * vocab_size)

# 3. Visualització
df_lik = pd.DataFrame(probs, columns=vocab, index=classes)

plt.figure(figsize=(10, 5))
sns.heatmap(df_lik, annot=True, cmap="Blues", fmt=".3f",
            cbar_kws={'label': r'$P(x_i | y)$'})
plt.title("Matriu de Versemblances amb Suavitzat de Laplace", fontsize=14, fontweight='bold')
plt.xlabel("Atributs (Token)", fontsize=12)
plt.ylabel("Classe", fontsize=12)
plt.tight_layout()

plt.savefig('fig_naive_bayes_heatmap.png', dpi=300)
plt.show()
```

Listing 4.6: Visualització de les Probabilitats Condicionals en NLP

Convertir Productes en Sumes

Un error comú és implementar la fórmula del producte directament:

$$P(y|\mathbf{x}) \propto P(y) \cdot P(x_1|y) \cdot P(x_2|y) \dots$$

Si tenim un document amb 2000 paraules, estem multiplicant 2000 nombres petits (ex:

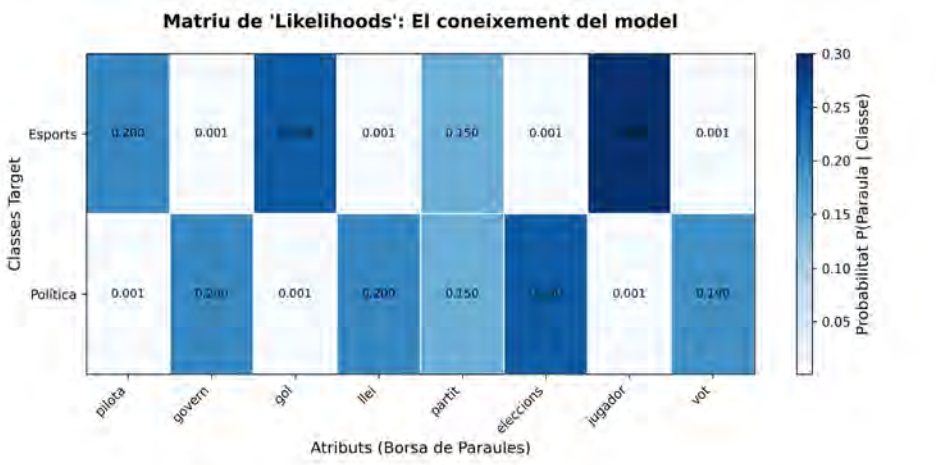


Figura 4.8: Mapa de calor de les probabilitats condicionals apreses ($P(x_i|y)$). Observeu com el terme "partit" pot tenir probabilitat significativa en més d'una classe. En aquests casos, Naive Bayes no decideix per una sola paraula: agrega l'evidència de totes les paraules del document. La combinació de termes com "partit" i "gol" pot augmentar la puntuació de la classe Esports, sempre que les probabilitats apreses i el prior ho afavoreixin.

Convertir Productes en Sumes *(continuació)*

$0.001^{2000} \approx 10^{-6000}$). Aquest nombre és molt més petit que el mínim representable per un flotant de 64 bits (10^{-308}). El resultat pot provocar *underflow aritmètic*: l'ordinador pot arrodonir el producte a zero i generar valors `-inf`.
Maximitzar una probabilitat és equivalent a maximitzar el seu logarisme (ja que $\log(x)$ és monòtona creixent), així que podem transformar un producte inestable en una suma estable:

$$\log P(y|\mathbf{x}) \propto \log P(y) + \sum_{i=1}^{d} \log P(x_i|y) \tag{4.28}$$

Això té una conseqüència fascinant: en l'espai logarítmic, **Naive Bayes es pot escriure com un classificador lineal en l'espai logarítmic**.

$$\text{Score} = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

On el "pes" β_i de cada paraula és simplement $\log P(x_i|y)$. Aquesta és la raó per la qual Naive Bayes és tan extremadament ràpid en temps d'inferència.

4.6.2 Del Text al Vector (Bag of Words)

Els algorismes de Machine Learning són funcions matemàtiques; no poden "llegir" Shakespeare ni correus brossa. Només entenen vectors numèrics $\mathbf{x} \in \mathbb{R}^d$. El procés de convertir text en números s'anomena **Vectorització**.

Una representació clàssica per aplicar Naive Bayes a text és el *Bag of Words* (BoW). La seva simplificació és forta: representa un document per les paraules que conté i les seves freqüències, ignorant l'ordre i la gramàtica. Això no captura tot el significat del text, però sovint és suficient per a tasques de classificació simples com spam, tema o sentiment bàsic.

Formalització Matemàtica: Definim un **Vocabulari** V com el conjunt de totes les paraules úniques presents en el corpus d'entrenament, ordenades lexicogràficament:

$$V = \{w_1, w_2, \ldots, w_{|V|}\}$$

Un document D es transforma en un vector $\mathbf{x}$ de dimensió $d = |V|$, on cada component x_j representa la freqüència del terme w_j en el document:

$$\mathbf{x} = [count(w_1, D), count(w_2, D), \ldots, count(w_{|V|}, D)]^T \tag{4.29}$$

Si utilitzem *Bernoulli Naive Bayes*, el vector és binari ($x_j \in \{0, 1\}$). Si usem *Multinomial Naive Bayes*, el vector conté enters positius.

Finalment, per a una representació BoW o binària adequada, el *score* logarítmic de Naive Bayes es pot escriure com una suma de contribucions per característica:

$$\text{Score}_c(\mathbf{x}) = \log P(y = c) + \sum_{i=1}^{d} x_i \log P(w_i \mid y = c). \tag{4.30}$$

Aquesta forma és lineal respecte a les característiques del document per a cada classe c, i explica per què Naive Bayes és tan ràpid en inferència. Ara bé, aquesta linealitat del *score* no implica que les probabilitats estiguin ben calibrades.

La Sopa de Lletres

Imagina que retalles totes les paraules d'una carta d'amor i les fiques en una bossa (Bag). Sacseges la bossa.

- Si treus les paraules i llegeixes: "estimar", "vida", "sempre", "tu"... encara que hagis perdut l'ordre de la frase, saps que és una carta romàntica.
- Si trobes: "factura", "impagament", "recàrrec", "banc"... saps que és un avís de cobrament.

Naive Bayes és com un cec que fica la mà a la bossa, treu un grapat de paraules i endevina el tema sense llegir la frase sencera.

El Repte de l'Esparsitat (Sparsity): En un problema real, el vocabulari V pot tenir 100.000 paraules. Tanmateix, un correu electrònic concret potser només en té 50. Això significa que el vector $\mathbf{x}$ tindrà 50 valors diferents de zero i 99.950 zeros. Això genera una **Matriu Esparsa**.

- **Problema:** Si guardem tots els zeros en memòria RAM (usant arrays estàndard de 64 bits), un dataset petit de 10.000 correus ocuparia: $10.000 \times 100.000 \times 8$ bytes ≈ 8 GB.
- **Solució d'Enginyeria:** Utilitzar estructures de dades especials (com `scipy .sparse.csr_matrix`) que només guarden els valors no nuls i els seus índexs. Això redueix l'ús de memòria de 8 GB a uns pocs MB.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer

corpus = [        # Corpus de joguina
    "El gos mossega la persona",        # Doc 1
    "La persona mossega el gos",        # Doc 2 (Idèntic en BoW)
    "El gat menja peix",           # Doc 3
```

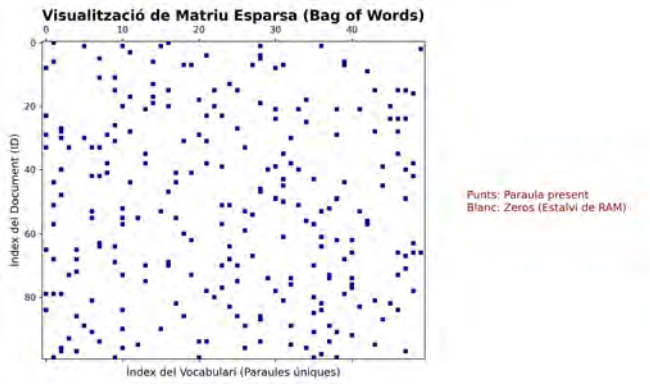


Figura 4.9: Spy Plot de la matriu document-terme. Els punts negres indiquen la presència d'una paraula (valors no nuls), mentre que l'espai blanc són zeros que, en aplicacions reals com NLP, ocupa el 99% del gràfic.

```
    "La persona pesca el peix al riu"  # Doc 4
]

# 1. Instanciar Vectoritzador (BoW)
vec = CountVectorizer()
X = vec.fit_transform(corpus) # Retorna matriu esparsa

# 2. Convertir a dens per visualitzar (Només per educació!)
df_bow = pd.DataFrame(X.toarray(), columns=vec.get_feature_names_out())
df_bow.index = [f"Doc {i+1}" for i in range(len(corpus))]

print("Representació Vectorial (BoW):")
print(df_bow)

# 3. Visualització de l'Esparsitat
plt.figure(figsize=(8, 4))
plt.spy(X, markersize=20, aspect='auto')
plt.title("Visualització de l'Esparsitat (Spy Plot)", fontsize=14, fontweight='bold')
plt.xlabel("Index del Vocabulari (Paraules)")
plt.ylabel("Index del Document")
plt.savefig('fig_sparsity.png', dpi=300)
plt.show()
```

Listing 4.7: Vectorització BoW i Visualització de l'Esparsitat

Consells de programació: El vocabulari també és part del model

En classificació de text, construir el vocabulari amb tots els documents abans del split és una forma de *data leakage*. El model ja ha vist quines paraules existeixen al test. El vocabulari, el `CountVectorizer` o el `TfidfVectorizer` s'han d'ajustar només amb train i aplicar després a validació i test.

Malgrat tot, el principal defecte del BoW és la pèrdua de context. Per mitigar-ho, s'utilitzen els **N-Grams**: en lloc de comptar paraules individuals (Unigrams), comptem seqüències de N paraules consecutives.

- **Unigrams (1-gram):** ["no", "és", "un", "bon", "model"]
- **Bigrams (2-gram):** ["no és", "és un", "un bon", "bon model"]
- **Trigrams (3-gram):** ["no és un", "és un bon", "un bon model"]

Escollim Arbres de Decisió quan...	**Escollim Naive Bayes quan...**
Necessitem **explicabilitat** legal ("Per què m'ha denegat el crèdit?").	Tenim **alta dimensionalitat** ($d > 10.000$), com en text o genòmica.
Hi ha relacions **no lineals** complexes i interaccions entre variables.	Necessitem entrenament i inferència en **temps real** (baixa latència).
Les dades són una barreja de tipus (numèriques, categòriques, ordinals).	Tenim poques dades (**Small Data**) i volem evitar l'overfitting.
No necessitem una probabilitat explícita, sinó regles locals fàcilment auditables.	Volem un *score* probabilístic ràpid i interpretable, sabent que pot requerir calibratge si s'ha d'usar com a probabilitat fiable.

Taula 4.1: Matriu de decisió estratègica per a l'enginyeria del Machine Learning.

L'Explosió Combinatòria

Compte amb els N-Grams! El vocabulari creix exponencialment. Si $|V|_{unigrams} = 10.000$, el nombre possible de bigrams és $10.000^2 = 100$ Milions. **Consell d'Experiència:** Mai utilitzis N-Grams sense aplicar una forta selecció de característiques (com χ^2 o freqüència mínima `min_df`). La majoria de combinacions de paraules ("de la", "que el") són soroll inútil. Filtra agressivament per quedar-te només amb les parelles amb significat real (ex: "Targeta Crèdit").

Micro-Cas: El cas de la "Falsa Positiva" de Harvard

Als inicis dels filtres d'spam (anys 2000), la Universitat de Harvard va patir un incident on molts correus d'admissió eren bloquejats automàticament pels servidors de correu. El motiu era que contenien frases com "Financial Aid", "Special Offer", "Opportunity" i "Deposit". El model Naive Bayes (unigrams), entrenat amb milions de correus brossa, havia après que la probabilitat individual d'aquestes paraules era altíssima en Spam:

$$P(\text{"Offer"}|\text{Spam}) \gg P(\text{"Offer"}|\text{Legit})$$

La solució va venir d'aplicar dues estratègies:
1. **Whitelisting:** Forçar probabilitat 1 (Prior absoluta) per a dominis '.edu'.
2. **Bigrams:** Introduir parelles de paraules al vocabulari. Mentre que "Offer" és sospitós, el bigrama "Admission Offer" té una probabilitat molt alta de ser legítim. Això trenca lleugerament l'assumpció d'independència per guanyar context semàntic crític.

4.7 Conclusions i Debat: La Justícia Algorítmica

En aquest capítol hem explorat dos paradigmes oposats per modelar el coneixement:

- **Arbres de Decisió (Lògica):** Fragmenten el món en regles rígides i ortogonals. Són interpretables, no paramètrics i imiten el raonament deductiu humà.
- **Naive Bayes (Probabilitat):** Modela la incertesa mitjançant la gestió d'evidències independents. Són robustos, eficients en alta dimensió i imiten la intuïció inductiva.

L'elecció entre un i l'altre no és només una qüestió de mètriques d'encert (Accuracy); és una decisió de disseny que es resumeix en la taula 4.1.

El "Benchmark" Obligatori

En qualsevol projecte cal entrenar primer un baseline simple i fort. En text, Naive Bayes o regressió logística poden ser punts de partida excel·lents; en dades tabulars, un arbre, un Random Forest o un model lineal regularitzat poden donar una referència sòlida. Si un model molt més complex només millora marginalment el baseline, cal justificar si el guany compensa el cost computacional, la pèrdua d'interpretabilitat i la complexitat de manteniment.

Per visualitzar la diferència aquests dos mètodes, hem generat una comparativa sobre el mateix dataset.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import make_classification

# 1. Dades sintètiques
X, y = make_classification(n_samples=300, n_features=2, n_informative=2,
                    n_redundant=0, random_state=42, n_clusters_per_class=1)

# 2. Entrenament
tree = DecisionTreeClassifier(max_depth=5)
bayes = GaussianNB()
tree.fit(X, y)
bayes.fit(X, y)

# 3. Plotting
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

# Funció auxiliar de plotting
def plot_boundary(clf, ax, title):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                         np.arange(y_min, y_max, 0.02))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.3, cmap='coolwarm')
    ax.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', cmap='coolwarm')
    ax.set_title(title, fontsize=14, fontweight='bold')

plot_boundary(tree, ax[0], "Arbre de Decisió (Ortogonal)")
plot_boundary(bayes, ax[1], "Naive Bayes (Gaussià)")

plt.tight_layout()
plt.savefig('fig_comparison_final.png', dpi=300)
plt.show()
```

Listing 4.8: Comparativa de Fronteres: Arbre vs. Gaussian Naive Bayes

4.7.1 Debat Ètic: Justícia Algorítmica i el "Dret al Procés"

La diferència tècnica entre els Arbres de Decisió (lògica) i Naive Bayes (estadística) transcendeix el codi i impacta directament en els drets civils. Quan aquests algorismes s'utilitzen en tribunals per a sentències automàtiques, en triatge mèdic o en la concessió d'hipoteques, l'enginyeria deixa de ser una tècnologia per convertir-se en una legislació *de facto*.

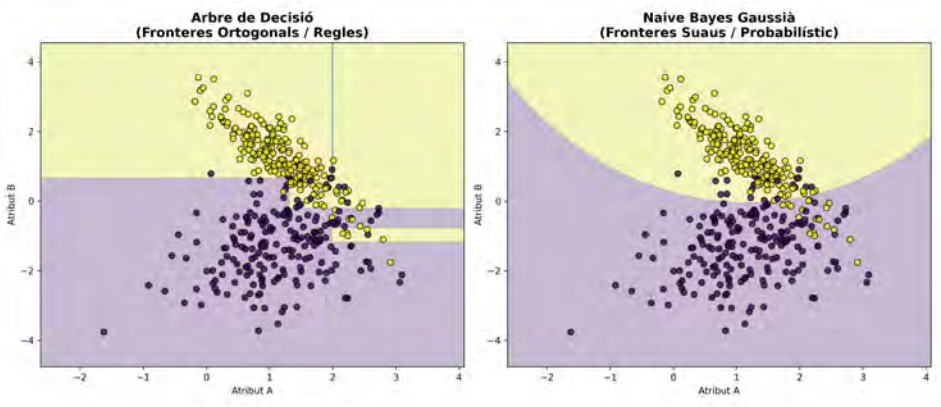


Figura 4.10: Comparativa geomètrica. L'**Arbre** (esquerra) crea fronteres rectes i tallants, aïllant regions específiques (potential overfitting). El **Naive Bayes** (dreta) genera una frontera corba i suau, resultant de la intersecció de les distribucions gaussianes, oferint una generalització més orgànica però menys flexible als detalls.

El dilema central en l'ètica de la IA és la tensió entre l'**Eficiència Utilitarista** (maximitzar el benestar total, l'especialitat de Bayes) i l'**Equitat Deontològica** (tractar a tothom segons regles universals, l'especialitat dels Arbres). A continuació, analitzem tres casos reals on l'elecció de l'algorisme determina la vida de les persones.

Cas 1: Justícia Penal i el Teorema d'Impossibilitat (El Cas COMPAS)

L'algorisme COMPAS s'utilitza àmpliament als Estats Units per predir el risc de reincidència d'un detingut. Aquesta predicció condiciona la llibertat sota fiança.

- **El Model Bayesià (La realitat):** El sistema calcula $P(\text{Reincidència}|\text{Perfil})$. Utilitza dades històriques massives. Com que històricament, degut a biaixos sistèmics i policials, certes minories han estat detingudes més sovint, la *Prior* $P(\text{Crim}|\text{Minoria})$ és més alta en les dades d'entrenament.
- **El Resultat:** Una investigació de ProPublica va demostrar que el model tenia una taxa de **Falsos Positius** el doble d'alta per a afroamericans que per a blancs. És a dir, s'etiquetava erròniament com a perillosos a negres innocents el doble de vegades que a blancs innocents.

Anàlisi d'Enginyeria: El Teorema d'Impossibilitat de Kleinberg Moltes persones de la comunitat d'enginyeria van intentar "arreglar" l'algorisme, però matemàticament és impossible satisfer totes les definicions de justícia simultàniament. Kleinberg, Mullainathan i Raghavan (2016) van demostrar que si les taxes base de crim (la prevalença real en les dades) són diferents entre dos grups, un algorisme no pot tenir alhora:

1. **Calibració:** (Si el model diu 70% risc, el 70% de la gent d'ambdós grups reincideix).
2. **Paritat de Falsos Positius:** (Equivocar-se igual en contra dels innocents d'ambdós grups).
3. **Paritat de Falsos Negatius:** (Equivocar-se igual a favor dels culpables d'ambdós grups).

Si escollim Bayes, optimitzem la Calibració (precisió global) però sacrifiquem la Paritat (justícia individual). Si escollim un Arbre amb regles que prohibeixin explícitament l'ús de variables proxy (com el codi postal), perdem precisió global però guanyem equitat processal.

En aquest cas, l'ús d'un Arbre de Decisió interpretable és superior èticament, encara que sigui menys precís. Un acusat té dret a saber: *"Se li denega la fiança perquè té 2 antecedents penals"*, i no *"Se li denega perquè s'assembla estadísticament a la gent que reincideix"*.

Cas 2: Triatge Mèdic en Pandèmia (Utilitarisme vs. Equitat)

Durant la crisi de la COVID-19, els hospitals van haver d'automatitzar la decisió de qui accedia a una UCI quan els recursos eren escassos.

- **Enfocament Bayesià (Maximitza Vides):** L'algorisme calcula la probabilitat de supervivència $P(\text{Viure}|\text{Edat, Comorbiditats})$. Assigna el respirador a la persona pacient amb la probabilitat més alta (sovint els més joves i sans).
- **Enfocament d'Arbre/Regles (Igualtat d'Oportunitats):** Una regla ètica podria ser "Primer en arribar, primer en ser atès" (FIFO) o "Loteria", ignorant l'edat o l'estat de salut previ per evitar discriminar discapacitats o gent gran.

El model Bayesià és "eficient" (salva més anys de vida totals), però pot ser discriminatori contra col·lectius vulnerables. Un Arbre de Decisió permet codificar regles de discriminació positiva (ex: *IF personal_sanitari THEN prioritat_maxima*), que en Bayes és difícil d'introduir.

Cas 3: "Redlining" Digital i Assegurances

El *redlining* era la pràctica històrica de marcar en vermell els barris pobres per denegar hipoteques. Avui dia és il·legal utilitzar la raça o la religió per calcular primes d'assegurança. Tanmateix, un model **Naive Bayes** entrenat amb milers de variables de comportament digital (historial de navegació, tipus de mòbil, hora de connexió) pot reconstruir aquestes variables.

Així, si el model detecta que "connectar-se sovint a la nit" correlaciona amb "major risc d'accident" —una correlació espúria o simplement indirecta—, podria penalitzar aquestes persones usuàries sense que hi hagi una relació causal clara. El resultat és que pagarien més sense entendre quin factor ha influït en la decisió. Aquí, en un Arbre de Decisió podem podar explícitament les branques que utilitzen variables sensibles. En Bayes, la informació està "difosa" en milers de petites probabilitats marginals, fent impossible extirpar el biaix.

4.7.2 Preguntes per a l'Aula

Qüestió 1: La Mentida Útil de la Independència

Pregunta: En un diagnòstic mèdic, sabem que "Febre" i "Tos" NO són independents donada la "Grip". Si tens grip i febre, és molt probable que tinguis tos. Per què, aleshores, Naive Bayes funciona tan bé en diagnòstic mèdic si la base teòrica és falsa?

Resposta d'Enginyeria: El secret rau en el fet que la classificació només requereix encertar el **signe** de la decisió, no la magnitud exacta. Considerem l'equació en l'espai logarítmic (Log-Odds):

$$\text{Score} = \log P(\text{Grip}) + \log \frac{P(\text{Febre}|\text{Grip})}{P(\text{Febre}|\neg\text{Grip})} + \log \frac{P(\text{Tos}|\text{Grip})}{P(\text{Tos}|\neg\text{Grip})} \tag{4.31}$$

Si "Febre" i "Tos" estan correlacionades positivament, el model comet **Double Counting** (compta l'evidència dues vegades).

- Evidència real: $+5$ punts.
- Naive Bayes calcula: $+5 + 5 = +10$ punts.

Això fa que la probabilitat final estigui mal calibrada (serà 99.9% en lloc de 85%), però la decisió final (Score > 0) segueix sent correcta! El model està "massa segur", però té raó. En classificació, l'ordre importa més que el valor exacte.

Qüestió 2: L'Atribut Fantasma en Producció

Pregunta: Heu entrenat un sistema de classificació de correus per a un banc. El sistema funciona perfectament (99% accuracy). Un mes després, el banc llança un nou producte anomenat "Cripto-Estalvi". De cop, tots els correus legítims que parlen d'aquest producte van a la carpeta de Spam. Per què?

Resposta d'Enginyeria: Aquest és un cas de **canvi de distribució** en producció. La paraula "cripto" gairebé no apareixia en correus legítims durant l'entrenament, però podia aparèixer associada a spam o frau. Quan el banc llança un producte legítim amb aquest terme, el patró après deixa de ser vàlid.

En un classificador Naive Bayes, això és especialment crític:

- $P(\text{"cripto"} \mid \text{Legítim})$ és nul·la o molt baixa.
- $P(\text{"cripto"} \mid \text{Spam})$ és més alta per les dades històriques.

Així, el model interpreta com a sospitós un terme que ara és legítim. El suavitzat de Laplace pot evitar probabilitats exactament zero, però no resol el problema de fons. En producció cal monitorar errors, detectar canvis de vocabulari, revisar casos recents i reentrenar el model amb dades actualitzades.

Qüestió 3: El Llindar de Decisió i la Matriu de Costos

Pregunta: En l'explicació teòrica hem dit que classifiquem com a Positiu si $P(H|E) > 0.5$. És així en una central nuclear? I en un filtre de correu brossa?

Resposta d'Enginyeria: Absolutament no. El llindar 0.5 assumeix que el cost d'un Fals Positiu (FP) és idèntic al d'un Fals Negatiu (FN).

- **Filtre d'Spam:**
  - FP (Correu important a la brossa): Catastròfic (perds oferta de feina).
  - FN (Spam a la bústia): Molèstia lleu.
  - *Enginyeria:* Movem el llindar a $P(\text{Spam}|E) > 0.99$. Preferim deixar passar brossa que perdre correus bons.
- **Detecció de Càncer (Screening):**
  - FP (Sostreball sa): Ensurt i més proves.
  - FN (Malalt a casa): Mort.
  - *Enginyeria:* Movem el llindar a $P(\text{Càncer}|E) > 0.05$. Volem detectar qualsevol sospita mínima (Alta Sensibilitat).

Tancament docent

Problemes de consolidació.

1. Donat un conjunt petit d'exemples amb dues o tres característiques categòriques, calcular l'entropia inicial, el guany d'informació de cada possible atribut i decidir quin hauria de ser el primer node d'un arbre de decisió.

Tancament docent *(continuació)*

2. A partir d'un arbre ja construït, classificar diverses instàncies noves, explicar el camí de decisió seguit i comparar aquesta explicació amb la que donaria un Naive Bayes per al mateix problema. Discutir quin model seria preferible si es prioritza interpretabilitat local, text, poques dades o capacitat d'explicar la decisió.
3. Donada una taula de freqüències per a dues classes, aplicar teorema de Bayes per calcular la probabilitat posterior d'una classe donada una observació. I si hi han valors no observats, incorporar suavitzat de Laplace.
4. Classificar un missatge curt com a *spam* o *no spam* amb Naive Bayes utilitzant un model Bag of Words. Calcular el resultat en espai logarítmic i justificar per què aquesta forma és més estable que multiplicar moltes probabilitats petites.
5. El classificador Bayesià ens dóna un número continu. La decisió binària final depèn de la **Matriu de Costos** del negoci. Això es coneix com a **Risc Bayesià**.
6. La clau no és només preguntar-se "quan encertarà el model?", sinó també "quan falli, quin tipus d'error és acceptable i quin no ho és?". Cal dissenyar el sistema perquè, en cas d'incertesa o error, tendeixi cap a una fallada segura (*fail-safe*).

Criteris breus de resolució. Una bona resolució hauria de mostrar el càlcul ordenat de les probabilitats o de les mesures d'impuresa, no només el resultat final. En els problemes d'arbres, cal justificar per què una partició redueix la incertesa i cal detectar quan un arbre està memoritzant soroll. En els problemes bayesians, cal separar correctament probabilitat prèvia, versemblança i posterior, i evitar el col·lapse de probabilitats mitjançant suavitzat. En els problemes de text, cal entendre que la hipòtesi d'independència de Naive Bayes és una simplificació forta, però sovint útil en enginyeria. De fet, quan s'interpreten sortides probabilístiques de Naive Bayes, cal recordar que poden servir com a puntuacions comparatives, però no necessàriament com a probabilitats calibrades sense validació addicional.

Connexió amb pràctica/laboratori. Aquest capítol es desplega en una pràctica centrada en models interpretables i classificació probabilística. L'alumnat pot construir un arbre de decisió, visualitzar-ne l'estructura, modificar paràmetres de profunditat i poda, comparar-lo amb un Naive Bayes sobre dades textuals i analitzar com canvia la decisió quan apareixen paraules rares, variables no observades o classes descompensades.

Capítol 5

Aprenentatge No Supervisat i Variables Latents

Fins ara, hem operat en un entorn confortable: teníem un "mestre" que ens donava les respostes correctes (les etiquetes y). Però, en molts contextos reals, la major part de les dades disponibles no arriben amb una etiqueta fiable. Etiquetar dades és car, lent i sovint requereix expertesa humana. En aquest capítol, entrem en l'aprenentatge no supervisat, on l'objectiu no és predir una resposta coneguda, sinó proposar estructures possibles a partir de les dades: agrupaments, components latents o patrons de densitat.

Guia d'activació docent

Repte inicial. Una plataforma vol segmentar persones usuàries sense etiquetes prèvies. La pregunta inicial és: com es poden proposar agrupaments útils quan ningú ha donat les classes correctes, i com sabem si aquests agrupaments tenen sentit?

Checkpoints conceptuals.

1. Distingir classificació supervisada i clustering no supervisat.
2. Explicar la diferència entre assignació dura en K-Means i responsabilitat probabilística en GMM.

Errors típics provocats.

1. Tractar els clústers com a classes naturals i objectives.
2. Triar K només pel gràfic del colze, sense criteri de model ni utilitat.

Microcas professional. Segmentació de clients: K-Means dona grups ràpids i interpretables, però pot fallar amb formes allargades o densitats diferents; GMM permet grups el·líptics i pertinences probabilístiques.

Connexió ètica o d'enginyeria. L'aprenentatge no supervisat no descobreix automàticament la veritat: proposa una estructura segons representació, mètrica i model.

5.1 El Problema de l'Agrupament (Clustering)

El *Clustering* és la tasca d'agrupar un conjunt d'objectes de manera que els objectes del mateix grup (clúster) siguin més similars entre si que als d'altres grups. Però què vol dir exactament "similar"? A més, en l'aprenentatge no supervisat, no disposem d'una etiqueta externa única contra la qual comparar el resultat. Per això, la qualitat d'un agrupament depèn de la representació, la mètrica, el nombre de grups i la utilitat del resultat per al problema.

Per entendre-ho bé: Un clúster no és una classe

Una classe és una etiqueta donada: "malalt", "sa", "spam", "no spam". Un clúster és una agrupació que el model proposa perquè alguns punts s'assemblen segons una mètrica. El clúster no diu què són els punts; diu que, amb aquesta representació, han quedat junts.

5.1.1 Formalització Matemàtica: La Partició de l'Espai

Sigui un dataset $D = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$, amb $\mathbf{x}^{(i)} \in \mathbb{R}^d$. A diferència de la classificació, no disposem d'etiquetes $y^{(i)}$. L'objectiu del clustering és assignar cada exemple a un dels K grups mitjançant una funció d'assignació:

$$c : \{1, \dots, N\} \to \{1, \dots, K\}, \tag{5.1}$$

on $c(i) = k$ indica que l'exemple i pertany al clúster k. A partir d'aquesta assignació, definim els clústers com a conjunts d'índexs:

$$C_k = \{i \in \{1, \dots, N\} : c(i) = k\}. \tag{5.2}$$

Aquests conjunts formen una partició dels exemples:

$$\bigcup_{k=1}^{K} C_k = \{1, \dots, N\}, \qquad C_i \cap C_j = \emptyset \text{ si } i \neq j. \tag{5.3}$$

Per entendre-ho bé: Particionar punts no sempre és partir l'espai

Quan fem clustering, el primer que fem és assignar exemples a grups. Alguns mètodes, com K-Means, també indueixen regions geomètriques de l'espai, com les cel·les de Voronoi. Altres models, com els GMM, no dibuixen fronteres rígides: assignen graus de pertinença probabilística.

Per avaluar la qualitat d'aquesta partició, definim dues mètriques antagòniques que volem optimitzar simultàniament:

1. Cohesió Intra-Clúster (Minimitzar) Volem que els punts dins d'un grup estiguin "atapeïts", és a dir, que s'assemblin molt entre ells. Mesurem la dispersió interna (Sum of Squared Errors - SSE):

$$J_{intra} = \sum_{k=1}^{K} \sum_{\mathbf{x} \in \mathcal{C}_k} ||\mathbf{x} - \mu_k||^2 \tag{5.4}$$

On μ_k és el centroide (mitjana geomètrica) del clúster k. Si $J_{intra} = 0$, significaria que tots els punts del clúster són idèntics al seu centroide (cohesió perfecta).

2. Separació Inter-Clúster (Maximitzar) Volem que els grups estiguin el més lluny possible els uns dels altres per garantir que són distints.

$$J_{inter} = \sum_{k=1}^{K} N_k ||\mu_k - \mu_{global}||^2 \tag{5.5}$$

On μ_{global} és la mitjana de totes les dades.

El Teorema de la Descomposició de la Variància: Curiosament, hi ha una relació fonamental en estadística que lliga aquests dos conceptes. La inèrcia total (variància) de les dades és constant, independentment de com les agrupem:

$$J_{total} = J_{intra} + J_{inter} = \text{Constant} \tag{5.6}$$

Com que la suma és constant, **minimitzar la cohesió interna (J_{intra}) equival matemàticament a maximitzar la separació externa (J_{inter})**. Això simplifica enormement el disseny d'algorismes (com K-Means): només ens cal preocupar-nos de fer els grups compactes, i la separació vindrà "de regal".

Organitzar la Biblioteca

Imagina que tens 1.000 llibres tirats a terra en una habitació i has de posar-los en prestatgeries. No tens cap llista de temes ni instruccions.

- **Intuïció:** Agafes un llibre de "Cuina". Busques altres que tinguin fotos de menjar i els poses junts. Després trobes un de "Viatges" i crees un nou piló.
- **Cohesió (J_{intra}):** Si en el piló de "Cuina" hi ha un llibre de "Mecànica quàntica", la cohesió és dolenta (hi ha molta "distància" temàtica). Vols que tots els llibres del prestatge siguin molt similars.
- **Separació (J_{inter}):** Vols que el prestatge de "Cuina" estigui ben diferenciat del de "Mecànica". Si barreges "Cuina Vegana" i "Cuina Tradicional", potser la separació és baixa i haurien d'anar junts.

El clustering és l'art de decidir on acaba un tema i on en comença un altre sense que ningú t'hagi dit quins temes existeixen.

5.1.2 La Subjectivitat per la Manca de "Ground Truth"

A diferència de l'aprenentatge supervisat, on podem calcular l'error de test (Accuracy) comparant la predicció $\hat{y}$ amb la realitat y, en clustering **no existeix la veritat absoluta (Ground Truth)**. Estem en un territori no supervisat; no hi ha cap mestre que ens corregeixi.

Que no tinguem etiquetes no vol dir que qualsevol agrupament sigui igual de bo. Podem avaluar un clustering amb criteris interns (cohesió, separació, silueta), amb estabilitat davant de re-mostreig, amb criteris d'informació o amb utilitat per al problema d'enginyeria. El que no tenim és una resposta única i universal abans de definir la representació, la mètrica i l'objectiu pràctic. Això implica que un mateix conjunt de dades pot tenir múltiples agrupacions vàlides, i l'elecció depèn exclusivament de la **mètrica de distància** i de l'enginyeria de característiques seleccionada.

Considerem un dataset format per: {Poma Vermella, Poma Verda, Pilota Vermella}.

- **Clustering per Color:** Si l'eix principal és l'espectre visible (x_{color}):

$$\mathcal{C}_1 = \{\text{Poma V., Pilota V.}\}, \quad \mathcal{C}_2 = \{\text{Poma Verda}\}$$

- **Clustering per Forma:** Si l'eix principal és la rodonesa o l'origen ($x_{biologia}$):

$$\mathcal{C}_1 = \{\text{Poma V., Poma Verda}\}, \quad \mathcal{C}_2 = \{\text{Pilota Vermella}\}$$

Cap de les dues solucions és matemàticament "incorrecta". Ambdues optimitzen la cohesió interna segons les variables que hem escollit. Per tant, l'avaluació d'un model de clustering sempre requereix **validació de negoci**: el clúster és útil per a l'objectiu de l'empresa?

El Teorema d'Impossibilitat de Kleinberg Per donar rigor matemàtic a aquesta subjectivitat, l'any 2002 Jon Kleinberg va demostrar un teorema que va sacsejar els fonaments del clustering. Va definir tres axiomes intuïtius que tot "bon" algorisme de clustering hauria de complir:

1. **Invariància d'Escala:** Si multipliquem totes les distàncies per una constant α, el resultat no hauria de canviar.
2. **Riquesa (Richness):** L'algorisme hauria de ser capaç de generar qualsevol partició possible de les dades.
3. **Consistència:** Si reduïm la distància entre punts del mateix clúster o augmentem la distància entre punts de clústers diferents, el resultat no hauria de canviar.

Kleinberg va demostrar que **és impossible** satisfer els tres axiomes simultàniament. Tot algorisme de clustering en viola almenys un. Això confirma que el clustering és inherentment subjectiu i depèn dels compromisos de disseny de l'enginyer.

Per entendre com detectem patrons, generarem un dataset sintètic amb estructures clares ("blobs") i visualitzarem com es distribueixen a l'espai.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

# 1. Generació de Dades Sintètiques
# Creem 4 grups amb diferents dispersions (desviació estàndard)
# random_state=42 assegura reproductibilitat
X, y_true = make_blobs(n_samples=500, centers=4, cluster_std=0.60, random_state=0)

plt.figure(figsize=(10, 6))        # 2. Visualització

# Mostrem les dades 'crues' (sense color) per simular el que veu l'algorisme
plt.scatter(X[:, 0], X[:, 1], s=50, color='gray', alpha=0.5, edgecolor='k', label='Dades No Etiquetades')

plt.title("El Repte del Clustering: Trobar Estructura en el Núvol", fontsize=16, fontweight='bold')
plt.xlabel("Característica 1 ($x_1$)")
plt.ylabel("Característica 2 ($x_2$)")
plt.grid(True, linestyle='--', alpha=0.3)

# Afegim anotacions per guiar l'ull
plt.annotate('Grup dens', xy=(-1.5, 7.5), xytext=(-4, 9),
             arrowprops=dict(facecolor='black', shrink=0.05), fontsize=12)
plt.annotate('Possibles outliers', xy=(2, 1), xytext=(4, -1),
             arrowprops=dict(facecolor='black', shrink=0.05), fontsize=12)

plt.tight_layout()
plt.savefig('fig_clustering_intro.png', dpi=300)
plt.show()
```

Listing 5.1: Generació i Visualització de Blobs per Clustering

La Maledicció de l'Escalat

Un error de principiant catastròfic en clustering és oblidar-se d'escalar les dades. Recordeu que el clustering es basa quasi sempre en la **Distància Euclidiana**.

- Variable A: "Sou anual" (rang 20.000 - 100.000).
- Variable B: "Edat" (rang 18 - 80).

Si calculeu la distància $\sqrt{(S_1 - S_2)^2 + (E_1 - E_2)^2}$, la diferència de sou (ex: 10.000) dominarà totalment sobre la diferència d'edat (ex: 10). L'algorisme agruparà **només per sou**, ignorant completament l'edat, perquè matemàticament l'edat és "soroll

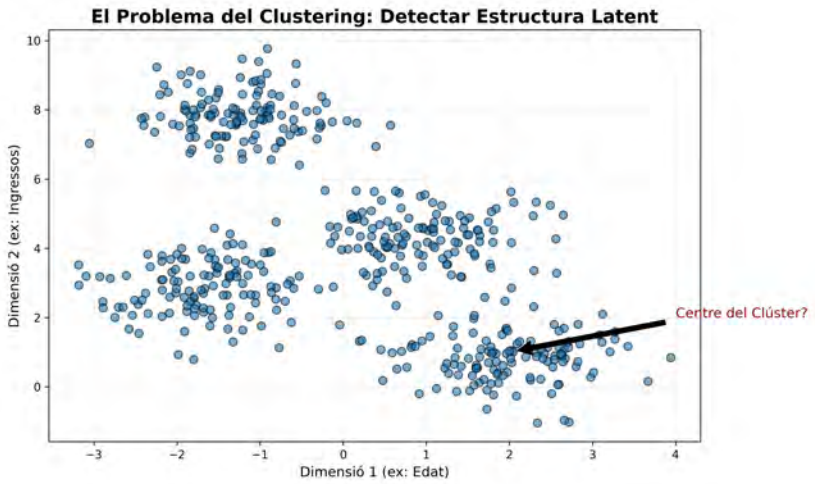


Figura 5.1: Visualització d'un dataset no etiquetat. L'ull humà detecta ràpidament 4 zones d'alta densitat gràcies a la nostra capacitat innata de reconeixement de patrons (Gestalt). L'objectiu dels algorismes de clustering és automatitzar aquesta intuïció visual utilitzant criteris matemàtics de distància.

La Maledicció de l'Escalat *(continuació)*

menyspreable" comparat amb la magnitud del sou. Sempre, sense excepció, apliqueu `StandardScaler` (normalització z-score) a totes les columnes abans de fer qualsevol tipus de clustering basat en distàncies.

Consells de programació: Abans d'escalar, mira què significa cada columna

En clustering, l'escalat canvia el resultat. Abans d'aplicar `StandardScaler`, pregunta't: aquesta variable és numèrica contínua? és categòrica? és un comptatge? té outliers forts? Escalar bé no és prémer un botó: és decidir quina geometria vols imposar al problema.

Micro-Cas: E-Commerce de Moda: Segmentació de la clientela (Model RFM)

En màrqueting, tractar a tota la clientela igual és ineficient. El clustering permet descobrir els **"Arquetips de Compradors"**. Una botiga online té 1 milió de persones usuàries. No saben qui són, però tenen dades de navegació:

- x_1: Preu mitjà dels productes visitats.
- x_2: Freqüència de compra.
- x_3: Dies des de l'última visita (Recency).

Aplicant clustering, l'algorisme descobreix 3 grups naturals:

1. **Grup A ("Caçadors d'Ofertes"):** Preu baix, Freqüència alta. → Acció: Enviar cupons de descompte.
2. **Grup B ("VIPs"):** Preu alt, Freqüència alta. → Acció: Oferir servei de Personal Shopper i accés exclusiu.
3. **Grup C ("Finestres"):** Visiten molt però compren poc. → Acció: Retargeting agressiu.

Aquests grups van emergir de les dades (Data-Driven Decision Making).

5.2 L'Algorisme K-Means: Geometria de Voronoi

En aprenentatge no supervisat, el model no rep una resposta correcta externa, sinó que imposa una estructura segons una mètrica, una representació i un nombre de grups. L'algorisme **K-Means** (o K-mitjanes), proposat originalment per Stuart Lloyd als Bell Labs el 1957 (per a la quantització de senyals), agrupa per **proximitat espacial**. La seva premissa és que un clúster es pot representar perfectament per un únic punt: el seu centre de gravetat o **centroide**.

Sigui un conjunt de dades $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ en un espai euclidià de d dimensions. El nostre objectiu és particionar aquestes dades en K clústers, cadascun definit per un prototip o centroide μ_k. Per formalitzar l'assignació, introduïm una variable binària $r_{nk} \in \{0, 1\}$ (anomenada "responsabilitat"), que val 1 si la dada $\mathbf{x}_n$ pertany al clúster k, i 0 altrament.

L'objectiu de K-Means és minimitzar la suma de distàncies quadràtiques de cada punt al seu centroide assignat. Aquesta funció de cost s'anomena **Distorsió** o **Inèrcia**:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \mu_k||^2 \tag{5.7}$$

On:

- N: Nombre total de mostres.
- K: Nombre de clústers (hiperparàmetre fixat a priori).
- $|| \cdot ||^2$: Distància Euclidiana al quadrat (norma L_2).

Minimitzar J trobant l'òptim global per a r_{nk} i μ_k simultàniament és un problema**NP-Hard**. Tanmateix, podem trobar un òptim local molt bo utilitzant un enfocament iteratiu conegut com a **Coordinate Descent** (Descens per Coordenades).

5.2.1 L'Algorisme de Lloyd: Iteració de Dos Passos

L'objectiu de K-Means és minimitzar la suma de distàncies quadràtiques de cada punt al centroide del seu clúster. L'algorisme de Lloyd redueix aquest cost a cada iteració i convergeix, però pot quedar atrapat en un mínim local. Per això, la inicialització dels centroides i l'execució de diverses inicialitzacions són decisions importants d'enginyeria. Aquest algorisme alterna entre optimitzar les assignacions r_{nk} (mantenint μ_k fixos) i optimitzar els centroides μ_k (mantenint r_{nk} fixos).

Pas 1: Assignació (The Assignment Step): Fixem els centroides μ_k. Volem minimitzar J respecte a r_{nk}. Com que la suma és independent per a cada n, minimitzem terme a terme assignant cada punt al centroide més proper:

$$r_{nk} = \begin{cases} 1 & \text{si } k = \arg\min_j ||\mathbf{x}_n - \mu_j||^2 \\ 0 & \text{altrament} \end{cases} \tag{5.8}$$

Geomètricament, això divideix l'espai en regions poligonals separades per hiperplans. Aquesta partició s'anomena **Tessel·lació de Voronoi**. Qualsevol punt dins d'una cel·la de Voronoi està més a prop del nucli d'aquesta cel·la que de qualsevol altre.

Pas 2: Actualització (The Update Step): Fixem les assignacions r_{nk}. Volem minimitzar J respecte a μ_k. Com que J és una funció quadràtica convexa respecte a μ, podem trobar el mínim igualant la derivada a zero:

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{n=1}^{N} r_{nk}(\mathbf{x}_n - \mu_k) = 0 \tag{5.9}$$

Aïllant μ_k:

$$\sum_{n=1}^{N} r_{nk}\mathbf{x}_n = \sum_{n=1}^{N} r_{nk}\mu_k \implies \mu_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}} \tag{5.10}$$

Aquest resultat és fonamental: el nou centroide òptim és simplement la **mitjana aritmètica** dels punts assignats al clúster. D'aquí ve el nom "K-Means".

La Franquícia de Pizzeries

Imagina que ets el cap de "Domino's Pizza" i vols obrir 3 pizzeries ($K = 3$) en una ciutat per servir a la clientela el més ràpid possible.

1. **Inicialització:** Poses les 3 botigues en llocs aleatoris del mapa.
2. **Pas 1 (Assignació):** La clientela truca a la pizzeria que té més a prop. Això crea 3 "zones de repartiment" (regions de Voronoi).
3. **Pas 2 (Actualització):** T'adones que una botiga està al límit de la seva zona i els repartidors fan viatges massa llargs. Decideixes moure la botiga exactament al centre geogràfic de la clientela que ha trucat avui.
4. **Repetició:** L'endemà, com que has mogut les botigues, alguna clientela que abans trucava a la botiga A ara té la B més a prop. Les zones canvien. Repeteixes el procés fins que les botigues es mouen més.

Ara visualitzarem com les fronteres de decisió (Voronoi) evolucionen fins a l'estabilitat.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from scipy.spatial import Voronoi, voronoi_plot_2d

# 1. Generació de Dades
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

fig, axs = plt.subplots(1, 3, figsize=(15, 5))   # Configuració de la figura
titles = ["Inicialització (Aleatòria)", "Iteració 3 (Ajust)", "Convergència (Final)"]

states = [              # Estats de l'algorisme a simular
    KMeans(n_clusters=4, init='random', n_init=1, max_iter=1, random_state=42),
    KMeans(n_clusters=4, init='random', n_init=1, max_iter=3, random_state=42),
    KMeans(n_clusters=4, init='random', n_init=1, max_iter=300, random_state=42)
]

def plot_voronoi(kmeans, ax, title):     # Funció auxiliar per dibuixar Voronoi
    kmeans.fit(X)
    centers = kmeans.cluster_centers_
    labels = kmeans.labels_

    # Malla de fons per acolorir regions (Voronoi manual)
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.05),
                         np.arange(y_min, y_max, 0.05))
    Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
```

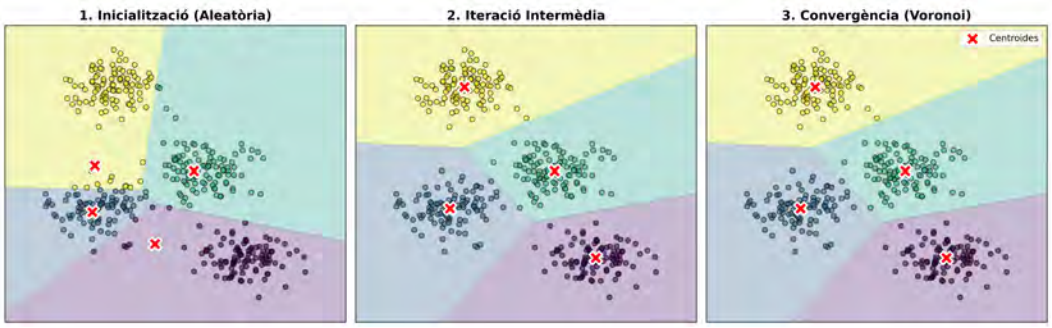


Figura 5.2: Evolució de l'algorisme K-Means. (Esquerra) Els centroides comencen en posicions aleatòries. (Centre) Després de poques iteracions, els centroides "migren" cap a les zones d'alta densitat. (Dreta) A la convergència, els centroides se situen al centre de massa dels clústers naturals, i les cel·les de Voronoi defineixen fronteres òptimes de separació.

```
    Z = Z.reshape(xx.shape)

    ax.contourf(xx, yy, Z, alpha=0.2, cmap='viridis')    # Plot regions i punts
    ax.scatter(X[:, 0], X[:, 1], c=labels, s=30, cmap='viridis', edgecolor='k')
    ax.scatter(centers[:, 0], centers[:, 1], c='red', s=200, marker='X', edgecolor='white',
    label='Centroides')
    ax.set_title(title, fontsize=14, fontweight='bold')
    ax.legend()

for i, model in enumerate(states):     # Bucle de renderitzat
    plot_voronoi(model, axs[i], titles[i])

plt.tight_layout()
plt.savefig('fig_kmeans_evolution.png', dpi=300)
plt.show()
```

Listing 5.2: Visualització de l'evolució de K-Means i Voronoi

K-Means és un Model Gaussià Disfressat

Part de l'alumnat veu K-Means com un mètode purament geomètric. La persona experta sap que K-Means és, en realitat, un cas particular dels **Models de Mixtura Gaussiana (GMM)**. Si assumeixes que:
1. Tots els clústers tenen la mateixa probabilitat a priori ($\pi_k = 1/K$).
2. Tots els clústers són esfèrics amb la mateixa variància ($\Sigma_k = \sigma^2 I$).
3. La variància σ^2 tendeix a zero ($\sigma^2 \to 0$, "hard assignment").

Llavors, la maximització de la versemblança en GMM esdevé matemàticament idèntica a minimitzar la distorsió Euclidiana en K-Means. Això explica per què K-Means falla quan els clústers són el·líptics o tenen mides molt diferents (està assumint esferes rígides).

5.2.2 Convergència i Complexitat Computacional

Una pregunta clàssica és: "K-Means sempre acaba o pot entrar en un bucle infinit?" La resposta matemàtica és que **sempre convergeix**:

- El nombre de maneres possibles d'assignar N punts a K grups és finit (encara que gegant, K^N).
- A cada pas (assignació i actualització), la funció de cost J disminueix estrictament o es manté igual. Mai augmenta.

- Per tant, l'algorisme ha d'arribar a un punt on J ja no pugui baixar més (un mínim local) en un temps finit.

La convergència vol dir que l'algorisme ha arribat a un punt on les assignacions i els centroides ja no canvien. Aquest punt pot ser un mínim local, i per això la inicialització és important.

Per altra banda, l'algorisme és molt eficient, amb complexitat $\mathcal{O}(t \cdot K \cdot N \cdot d)$, on t és el nombre d'iteracions. Com que normalment $t, K, d \ll N$, es considera linear respecte al nombre de dades, fent-lo eficient i escalable. Però pot resultar costós en datasets molt grans o en alta dimensió.

Micro-Cas: Compressió d'imatges

K-Means és la base de molts algorismes de compressió amb pèrdua.
El Problema: Una imatge Full HD (1920×1080) té 2 milions de píxels. Cada píxel té 3 canals de color (R, G, B) de 8 bits (0-255). Total colors possibles: $256^3 \approx 16.7$ milions. Mida en memòria: $2.000.000 \times 3$ bytes ≈ 6 MB.
La Solució K-Means: Volem representar aquesta imatge utilitzant només $K = 16$ colors per estalviar espai, però que es vegi "igual" a l'ull humà.

1. **Transformació:** Convertim la imatge en una matriu de dades X de mida $(N_{pixels}, 3)$. Cada píxel és un punt en l'espai 3D de colors.
2. **Clustering:** Executem K-Means amb $K = 16$.
3. **Codebook:** Els 16 centroides resultants $\mu_1, \ldots, \mu_{16}$ formen la "Paleta Òptima" per a aquesta imatge específica, els colors més representatius.
4. **Compressió:** Substituïm cada píxel original per l'índex del seu centroide més proper ($k \in \{0..15\}$).

Resultat d'Enginyeria:

- *Abans:* Cada píxel ocupava 24 bits (3 bytes).
- *Després:* Cada píxel ocupa 4 bits (per indexar 16 colors) + una petita taula de colors (el codebook).
- **Compressió:** Reducció de mida del **83%** (de 6 MB a ≈ 1 MB) amb una pèrdua visual mínima.

5.3 Limitacions de K-Means i Inicialització

Tot i que K-Means és l'algorisme de clustering més popular, la simplicitat de l'algorisme (basar-se exclusivament en la distància Euclidiana al quadrat) comporta un conjunt d'assumpcions geomètriques molt fortes sobre la forma de les dades. Quan les assumpcions no es compleixen, K-Means dóna resultats matemàticament òptims per a la seva funció de cost, però totalment inútils per al problema.

5.3.1 L'Assumpció de Variància Isotrópica (Esfericitat)

La funció de cost J minimitza $||\mathbf{x} - \mu||^2$. Geomètricament, això implica que l'algorisme assumeix que tots els clústers són **esfèrics (o hiperesfèrics en** $\mathbb{R}^d$ i tenen aproximadament la mateixa mida (radi).

Des d'una perspectiva probabilística, K-Means és un cas particular d'estimació de màxima versemblança on assumim que les dades han estat generades per K distribucions Gaussianes amb matrius de covariància proporcionals a la identitat:

$$\Sigma_k = \sigma^2 \mathbf{I} \tag{5.11}$$

És a dir, la variància és la mateixa en totes les direccions de l'espai (**Isotropia**). Però si els clústers tenen formes geomètriques complexes (com dues mitges llunes entrellaçades o anells concèntrics), la distància Euclidiana al centroide no és una bona mètrica de pertinença: en una mitja lluna, el "centre de gravetat" (mitjana) cau al buit, fora del conjunt de punts. K-Means intentarà tallar les llunes amb una línia recta (un hiperplà de Voronoi), barrejant punts de les dues llunes.

El Problema de la Densitat i la Mida: Si tenim dos clústers, un molt dens i petit (A) i un altre molt dispers i gran (B), K-Means tendirà a trencar el clúster gran o "robar" punts del petit per equilibrar la inèrcia J. L'algorisme prefereix clústers d'igual volum espacial, ignorant la densitat real.

Explicació per a Dummies: La Festa a la Piscina

Imagina una festa amb dos grups de gent:

- **Grup A (La Conga):** Una fila llarga i corba de persones ballant agafades per la cintura.
- **Grup B (La Rotllana):** Un grup compacte al voltant de les begudes.

K-Means és com un encarregat de seguretat molt quadriculat que vol separar la gent en dues zones rodones.

- Plantarà una bandera al mig de la rotllana (correcte).
- Però per a la Conga, plantarà la bandera al mig de la sala. Com que la Conga és allargada, la meitat de la cua està "més a prop" de la zona de les begudes que del seu propi centre.
- **Resultat:** L'encarregat tallarà la Conga pel mig i dirà que la meitat de la gent ballant pertany al grup de les begudes. Matemàticament és correcte (distància), però socialment és absurd (connectivitat).

5.3.2 El Problema de la Inicialització: K-Means++

La funció de cost J no és convexa. Això significa que té múltiples valls (mínims locals). L'algorisme K-Means és un mètode de descens de gradient; és com deixar caure una pilota per una muntanya. La pilota acabarà al fons de la vall més propera, però no necessàriament a la vall més profunda de tot el mapa (l'òptim global).

El lloc on acaba la pilota depèn exclusivament d'on la deixem caure (la **inicialització** dels centroides). Si inicialitzem dos centroides dins del mateix clúster real i cap en un altre, l'algorisme pot quedar atrapat en una configuració subòptima on un clúster es divideix en dos i altres dos es fusionen erròniament.

Consells de programació: No confiïs en una sola execució de K-Means

A `scikit-learn`, per defecte, l'algorisme s'executa 10 vegades (`n_init=10`) amb diferents llavors aleatòries, i es queda amb la millor solució (menor inèrcia J).
En problemes d'alta dimensionalitat o amb molts clústers, 10 és insuficient. Així, si $K > 50$, augmenteu `n_init` a 50 o 100 per evitar caure en mínims locals.
A més, K-Means pot acabar en mínims locals diferents segons la inicialització. Es recomana fixar `random_state` per reproduïbilitat i utilitzar diverses inicialitzacions amb `n_init`. Així es pot comparar la inèrcia i les mides dels clústers. Si cada execució dona una història diferent, el clustering no és estable.

L'any 2007, David Arthur i Sergei Vassilvitskii van proposar un mètode d'inicialització que garanteix teòricament que la solució final no estarà massa lluny de l'òptim ($\mathcal{O}(\log K)$). K-Means++ no elimina el problema dels mínims locals, però redueix el risc de començar amb centroides molt mal situats. La idea és senzilla: escollir centroides inicials separats entre si, de manera que l'algorisme comenci cobrint millor l'espai de dades:

1. Triar el primer centre μ_1 uniformement a l'atzar entre les dades.
2. Per a cada punt de dades $\mathbf{x}_n$, calcular la distància al centre més proper

$$D(\mathbf{x}_n) = \min_{j \in \{1..k_{actual}\}} ||\mathbf{x}_n - \mu_j|| \tag{5.12}$$

3. Triar el següent centre μ_{new} seleccionant un punt $\mathbf{x}_n$ amb una probabilitat proporcional al quadrat de la seva distància:

$$P(\text{triar } \mathbf{x}_n) = \frac{D(\mathbf{x}_n)^2}{\sum_{m=1}^{N} D(\mathbf{x}_m)^2} \tag{5.13}$$

4. Repetir els passos 2 i 3 fins a tenir K centres.
5. Executar l'algorisme K-Means estàndard.

Aquest mecanisme "empeny" els centres a dispersar-se inicialment, evitant que dos centroides comencin dins del mateix grup dens. De fet, K-Means++ no elimina el problema dels mínims locals, però proporciona una inicialització molt més informada que triar centroides completament a l'atzar. La idea és escollir centroides inicials separats entre si, de manera que l'algorisme comenci cobrint millor l'espai de dades. Això sol millorar la qualitat final i reduir la variabilitat entre execucions.

A continuació, generem visualitzacions que demostren com K-Means falla en dades no esfèriques i com K-Means++ ajuda.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_moons, make_blobs

# 1. Generació de Datasets problemàtics
n_samples = 1500
random_state = 170

# Cas A: Make Moons (No convex)
X_moons, y_moons = make_moons(n_samples=n_samples, noise=0.05, random_state=random_state)

# Cas B: Blobs Anisotròpics (Allargats)
X_aniso, y_aniso = make_blobs(n_samples=n_samples, random_state=random_state)
transformation = [[0.6, -0.6], [-0.4, 0.8]] # Matriu de transformació lineal
X_aniso = np.dot(X_aniso, transformation)

# Cas C: Variància Desigual (Blobs de diferent mida)
X_varied, y_varied = make_blobs(n_samples=n_samples, cluster_std=[1.0, 2.5, 0.5],
    random_state=random_state)

# 2. Aplicació de K-Means
# Noteu que sabem el nombre correcte de clústers (k=2 o k=3), però l'algorisme fallarà
    igualment
y_pred_moons = KMeans(n_clusters=2, random_state=random_state).fit_predict(X_moons)
y_pred_aniso = KMeans(n_clusters=3, random_state=random_state).fit_predict(X_aniso)
```

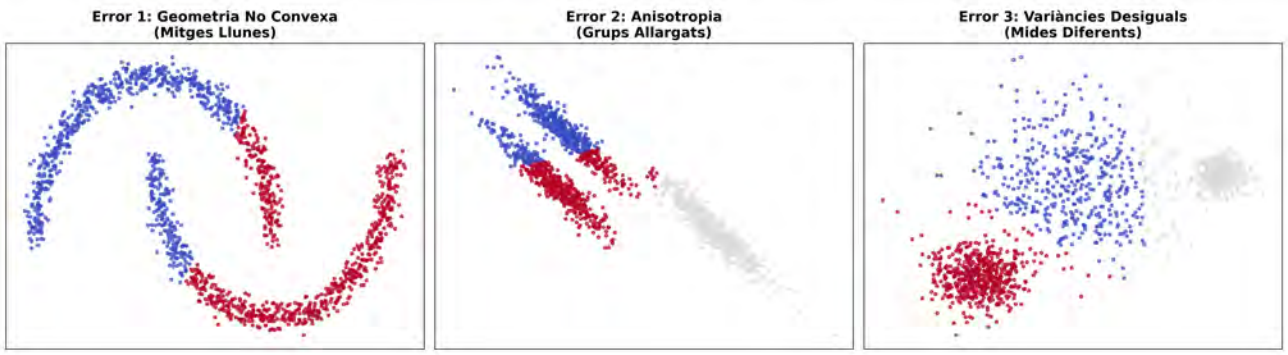


Figura 5.3: Errors de K-Means. (Esquerra) En les dades tipus "Moons", la frontera de decisió lineal talla les formes naturals. (Centre) En dades allargades, K-Means fragmenta els grups perquè assumeix esfericitat. (Dreta) En grups de densitat variable, els punts perifèrics del grup gran són "absorvits" pel grup petit i dens.

```
y_pred_varied = KMeans(n_clusters=3, random_state=random_state).fit_predict(X_varied)

# 3. Visualització
fig, axs = plt.subplots(1, 3, figsize=(18, 5))

# Plot Moons
axs[0].scatter(X_moons[:, 0], X_moons[:, 1], c=y_pred_moons, cmap='viridis', s=10)
axs[0].set_title("Problema 1: No Convexitat (Moons)\nK-Means talla recte", fontsize=12,
    fontweight='bold')

# Plot Anisotropic
axs[1].scatter(X_aniso[:, 0], X_aniso[:, 1], c=y_pred_aniso, cmap='viridis', s=10)
axs[1].set_title("Problema 2: Anisotropia (Cigars)\nK-Means assumeix cercles", fontsize=12,
    fontweight='bold')

# Plot Varied
axs[2].scatter(X_varied[:, 0], X_varied[:, 1], c=y_pred_varied, cmap='viridis', s=10)
axs[2].set_title("Problema 3: Mides Diferents\nK-Means penalitza la dispersió", fontsize=12,
    fontweight='bold')

plt.tight_layout()
plt.savefig('fig_kmeans_failures.png', dpi=300)
plt.show()
```

Listing 5.3: Limitacions de K-Means en dades complexes

Sobressegmentació i Fusió

Si sabeu que els vostres clústers no són esfèrics, una estratègia "pro" és utilitzar un K molt més gran del necessari (ex: $K = 50$ per a un problema de 5 grups reals). Això es diu **Sobressegmentació (Over-segmentation)**. K-Means cobrirà la forma complexa (ex: la mitja lluna) amb moltes petites esferes ("Voronoi cells"). **Pas posterior:** Després, un segon algorisme (o regla de negoci) fusiona aquests micro-clústers basant-se en connectivitat o densitat. És millor tenir 10 peces de puzzle correctes que 1 peça gran i errònia.

Micro-Cas: L'Eixam Sísmic i el Terratrèmol Principal

La localització d'epicentres és un exemple crític on les assumpcions de K-Means poden portar a errors científics greus. **Escenari:** Un sismògraf registra milers d'esdeveniments en una zona de falla activa.

- **Estructura A:** Un gran terratrèmol principal amb milers de rèpliques disperses en una àrea de 100 km (Alta dispersió, σ_A alt).
- **Estructura B:** Una falla secundària propera amb micro-terratrèmols molt localitzats en 5 km (Baixa dispersió, σ_B baix).

Si apliquem K-Means estàndard per separar les dues fonts sísmiques: **L'Error:** Com que K-Means intenta minimitzar la variància intra-clúster ($\sum ||x - \mu||^2$), penalitza molt els punts llunyans del clúster A. Per reduir el "cost", l'algorisme assignarà les rèpliques llunyanes del terratrèmol A al clúster B, simplement perquè estan geomètricament més a prop del centre de B, tot i que geològicament pertanyen a la falla A.
En aquests casos, K-Means és inservible. Cal utilitzar algorismes basats en densitat com **DBSCAN** o models probabilístics com **GMM**.

5.4 Modelització per Mixtura de Gaussianes (GMM)

Fins ara, hem aplicat una visió rígida del món: un punt pertany a un grup o no hi pertany. K-Means és un algorisme de **Hard Clustering** (assignació dura). Si un punt cau a 0.001 mm de la frontera de Voronoi, se li assigna la classe A amb un 100% de certesa, ignorant que està gairebé a la classe B.

A continuació farem el salt cap al **Soft Clustering** (assignació tova) i l'aprenentatge generatiu amb els **Gaussian Mixture Models (GMM)**, un marc probabilístic que assumeix que les dades són generades per una combinació de distribucions normals multivariants.

5.4.1 La Distribució Normal Multivariant

Per entendre GMM, primer hem de dominar el seu bloc constructiu: la campana de Gauss en d dimensions. La funció de densitat de probabilitat (PDF) d'una variable aleatòria $\mathbf{x} \in \mathbb{R}^d$ que segueix una distribució normal multivariant ve donada per la següent equació:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right) \tag{5.14}$$

Analitzem els components des d'una perspectiva d'àlgebra lineal avançada:

- $\mu \in \mathbb{R}^d$: El vector de mitjanes. Determina el **centre** de l'el·lipse.
- $\Sigma \in \mathbb{R}^{d \times d}$: La matriu de covariància. Determina la **forma, mida i orientació** de l'el·lipse. Ha de ser simètrica i definida positiva.
- $|\Sigma|$: El determinant de la matriu de covariància. Actua com un factor de normalització de volum.
- $(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)$: Aquesta forma quadràtica és la **Distància de Mahalanobis** al quadrat (Δ^2).

Per entendre com Σ controla la forma del clúster, podem aplicar la descomposició en valors propis (eigendecomposition):

$$\Sigma = U \Lambda U^T = \sum_{i=1}^{d} \lambda_i \mathbf{u}_i \mathbf{u}_i^T \tag{5.15}$$

On:

- $\mathbf{u}_i$ (vectors propis) defineixen els **eixos principals de rotació** de l'el·lipse.
- $\sqrt{\lambda_i}$ (arrels dels valors propis) defineixen la **longitud** dels eixos (l'estirament en cada direcció).

Això és el que dóna a GMM el seu aventatge comparat amb K-Means: mentre K-Means està atrapat en cercles (on $\lambda_1 = \lambda_2 = \cdots = \sigma^2$), GMM pot modelar el·lipses allargades i rotades ($\lambda_1 \gg \lambda_2$), adaptant-se a correlacions entre variables.

5.4.2 El Model de Mixtura (The Mixture Model)

Un GMM és una suma ponderada de K components gaussianes. La probabilitat d'observar un punt $\mathbf{x}$ és:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \tag{5.16}$$

On:

- K: Nombre de components (clústers).
- π_k: Coeficient de barreja (*mixing coefficient*). Representa la probabilitat a priori de pertànyer al grup k.
- Restriccions: $0 \leq \pi_k \leq 1$ i $\sum_{k=1}^{K} \pi_k = 1$.

Aquesta equació descriu un **Model Generatiu**. Ens diu com generar noves dades sintètiques: 1. Triem un clúster k amb probabilitat π_k (llancem un dau de K cares trucat). 2. Mostregem un punt $\mathbf{x}$ de la distribució normal $\mathcal{N}(\mu_k, \Sigma_k)$.

5.4.3 Probabilitat Tova i Responsabilitat

En lloc de dir "el punt n és del clúster k", ara calculem la probabilitat posterior (usant Bayes):

$$\gamma(z_{nk}) \equiv P(z_k = 1|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \Sigma_j)} \tag{5.17}$$

El valor $\gamma(z_{nk})$ s'anomena **responsabilitat**. Si per a un punt tenim $\gamma = [0.1, 0.4, 0.5]$, vol dir que el model "dubta": creu que és del grup 3 amb un 50% de certesa, però admet que podria ser del grup 2.

Si volem trobar els paràmetres òptims $\theta = \{\pi, \mu, \Sigma\}$ que maximitzin la versemblança (Log-Likelihood) de les dades:

$$\ln p(\mathcal{D}|\theta) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \right) \tag{5.18}$$

Aquí tenim una **suma dins d'un logaritme**. La derivada del logaritme d'una suma no es simplifica ($\ln(A+B) \neq \ln A + \ln B$). Això fa que igualar la derivada a zero per trobar l'òptim analíticament sigui impossible. Per resoldre això, necessitem un algorisme iteratiu més potent que el descens de gradient: l'algorisme **Expectation-Maximization (EM)**, que veurem a la següent secció.

Tipus	Descripció Geomètrica	Paràmetres
`spherical`	Els clústers han de ser esfèrics ($\Sigma_k = \sigma_k^2 I$). Similar a K-Means però amb mides variables.	K
`diag`	Els eixos de l'el·lipse estan alineats amb els eixos cartesians. Pot estirar-se però no rotar. (Σ_k és diagonal).	$K \times d$
`tied`	Tots els clústers comparteixen la mateixa forma i orientació ($\Sigma_k = \Sigma$), però tenen diferent centre.	$d(d+1)/2$
`full`	Llibertat total. Cada clúster pot tenir qualsevol forma, mida i rotació. És el més potent però el més lent i propens a l'overfitting.	$K \times \frac{d(d+1)}{2}$

Taula 5.1: Tipus de restriccions de covariància en GMM.

Singularitats i Regularització

Un problema comú en GMM és el "col·lapse de la covariància". Si un clúster té un sol punt (o punts molt propers), la seva variància σ_k^2 tendeix a zero.

$$\mathcal{N}(x|\mu, \sigma \to 0) \to \infty$$

La versemblança explota cap a l'infinit (Singularitat), i el codi falla (errors numèrics o *NaNs*). **Truc d'Experiència:** A `scikit-learn`, utilitzeu sempre el paràmetre `reg_covar` (per defecte $1e-6$). Això afegeix un petit valor ϵ a la diagonal de la matriu de covariància ($\Sigma + \epsilon I$), impedint que sigui singular. És com un "suavitzat" matemàtic que garanteix l'estabilitat numèrica.

Per entendre-ho bé: Responsabilitat vol dir pertinença parcial

En un GMM, una mostra no ha de pertànyer al 100% a un únic grup. Pot ser un 70% compatible amb un component i un 30% compatible amb un altre. Aquesta proporció és la responsabilitat. No és una causa ni una certesa: és el grau amb què cada component explica aquella mostra segons el model.

Un dels paràmetres més importants a ajustar en llibreries com `scikit-learn` és el `covariance_type`. Això determina quants graus de llibertat té el model per adaptar-se. A continuació, generarem dades anisotròpiques (allargades) per demostrar la superioritat de GMM sobre K-Means.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# 1. Generació de Dades "Estirades" (Anisotròpiques)
X, y_true = make_blobs(n_samples=500, centers=2, cluster_std=0.60, random_state=0)
# Apliquem una transformació lineal per estirar els blobs
rng = np.random.RandomState(13)
X_stretched = np.dot(X, rng.randn(2, 2))

# 2. Entrenament dels Models
# K-Means (Assumeix cercles)
kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
```

```
kmeans.fit(X_stretched)
y_kmeans = kmeans.predict(X_stretched)

# GMM (Assumeix el·lipses completes)
gmm = GaussianMixture(n_components=2, covariance_type='full', random_state=42)
gmm.fit(X_stretched)
y_gmm = gmm.predict(X_stretched)

# 3. Funcions de Visualització (Auxiliars)
def draw_ellipse(position, covariance, ax=None, **kwargs):
    ax = ax or plt.gca()
    # Descomposició espectral per trobar angle i amplada
    if covariance.shape == (2, 2):
        U, s, Vt = np.linalg.svd(covariance)
        angle = np.degrees(np.arctan2(U[1, 0], U[0, 0]))
        width, height = 2 * np.sqrt(s)
    else:
        angle = 0
        width, height = 2 * np.sqrt(covariance)

    # Dibuixem múltiples el·lipses per simular nivells de confiança (1, 2, 3 sigma)
    for nsig in range(1, 4):
        ax.add_patch(Ellipse(position, nsig * width, nsig * height,
                             angle, **kwargs, alpha=0.2/nsig))

# 4. Plotting
fig, ax = plt.subplots(1, 2, figsize=(14, 6))

# Plot K-Means
ax[0].scatter(X_stretched[:, 0], X_stretched[:, 1], c=y_kmeans, s=30, cmap='viridis', zorder
    =2)
ax[0].set_title("K-Means: Fronteres Rígides (Cercles)", fontsize=14, fontweight='bold')
# K-means no té covariància real, dibuixem cercles basats en la distància màxima
centers = kmeans.cluster_centers_
ax[0].scatter(centers[:, 0], centers[:, 1], c='red', s=200, marker='X', zorder=3)

# Plot GMM
ax[1].scatter(X_stretched[:, 0], X_stretched[:, 1], c=y_gmm, s=30, cmap='viridis', zorder=2)
ax[1].set_title("GMM: Fronteres Adaptatives (El·lipses)", fontsize=14, fontweight='bold')
for pos, covar, w in zip(gmm.means_, gmm.covariances_, gmm.weights_):
    draw_ellipse(pos, covar, ax=ax[1], color='red')

plt.tight_layout()
plt.savefig('fig_gmm_vs_kmeans.png', dpi=300)
plt.show()
```

Listing 5.4: Comparativa K-Means vs GMM amb Contorns de Probabilitat

Per entendre-ho bé: GMM és més flexible, però també assumeix una forma

Un GMM és més flexible que K-Means perquè permet grups el·líptics, amb mides i orientacions diferents. Però no és lliure de supòsits: assumeix que les dades es poden explicar com una combinació de núvols gaussianes. Si els grups tenen formes molt irregulars, un GMM també pot ser inadequat.

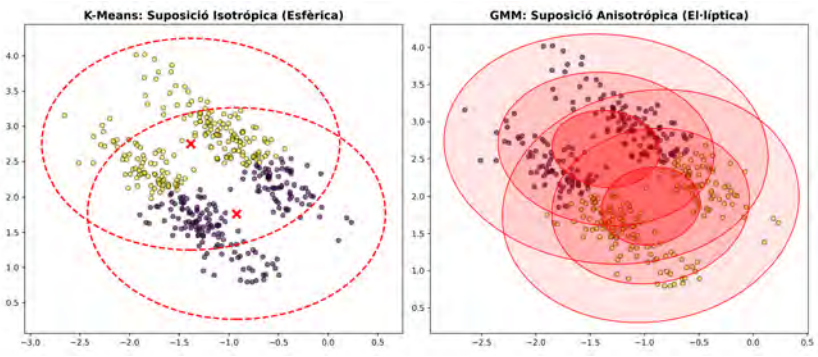


Figura 5.4: Comparativa geomètrica entre K-Means i GMM. A l'esquerra, K-Means imposa fronteres de Voronoi i pot fallar quan les dades tenen una orientació diagonal o formes no esfèriques. A la dreta, un GMM amb covariància completa pot ajustar el·lipses orientades, representant millor grups amb diferents formes, mides i correlacions.

Micro-Cas: Detectant el "Bear Market"

Una aplicació clàssica dels GMM és detectar règims latents en dades temporals. Per exemple, els rendiments diaris d'un índex financer poden alternar períodes de baixa volatilitat i períodes de crisi amb alta volatilitat. Un model de mixtura de dues Gaussianes podria representar:

1. Un règim de mercat estable, amb volatilitat baixa.
2. Un règim de mercat tens, amb volatilitat alta i més probabilitat d'extrems.

Quan arriba una nova observació, el model pot calcular la responsabilitat posterior de cada règim. Aquesta informació no hauria de traduir-se mecànicament en una acció automàtica, però pot servir com a senyal de risc per a una revisió humana o per a un sistema de control més ampli. K-Means difícilment captura aquest patró si la diferència principal és la variància i no la mitjana.

5.5 L'Algorisme Expectation-Maximization (EM)

Arribem al mecanisme central que permet entrenar models amb variables latents. Mentre que K-Means assigna cada punt a un únic clúster, l'algorisme Expectation-Maximization (EM) permet treballar amb assignacions incertes: primer estima graus de pertinença i després actualitza els paràmetres del model. Publicat en la seva forma general per Dempster, Laird i Rubin el 1977, l'EM és és un procediment general per ajustar models amb variables latents o dades incompletes.

El problema que volem resoldre és l'ajust dels paràmetres d'un model de mixtura de Gaussianes (GMM): $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$. El nostre objectiu és maximitzar la Log-Versemblança (Log-Likelihood) de les dades observades $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$:

$$\ln p(\mathcal{D}|\theta) = \sum_{n=1}^{N} \ln \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \right) \tag{5.19}$$

Si intentem derivar aquesta expressió respecte a μ_k i igualar a zero (Maximum Likelihood Estimation - MLE), ens trobem amb un bloqueig algebraic insuperable. El logaritme $\ln(\cdot)$ no pot "entrar" dins del sumatori $\sum_k$. Això provoca que els paràmetres de tots els clústers quedin "acoblats" en una equació no lineal complexa. No existeix una solució tancada (closed-form solution).

La genialitat de l'EM rau en fer veure que el problema és "fàcil" si coneguéssim una informació que ens falta. Imaginem que, per a cada dada $\mathbf{x}_n$, coneguéssim realment a quin clúster pertany. Definim una variable latent binària $\mathbf{z}_n \in \{0,1\}^K$ (un vector one-hot), on $z_{nk} = 1$ si $\mathbf{x}_n$ va ser generat pel clúster k. Si coneguéssim $\mathbf{z}$, la **Complete-Data Log-Likelihood** seria:

$$\ln p(\mathcal{D}, \mathcal{Z}|\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \left[\ln \pi_k + \ln \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k)\right] \tag{5.20}$$

Observeu el miracle: ara el logaritme actua directament sobre la Gaussiana (cancel·lant l'exponencial) i el sumatori $\sum_k$ està fora. Això és trivial de maximitzar! El problema, és clar, és que no coneixem z_{nk}.

L'EM resol això iterant dos passos:

1. **Pas E (Expectation):** Estimem els valors esperats de z_{nk} (les responsabilitats) usant els paràmetres actuals.
2. **Pas M (Maximization):** Maximitzem la versemblança completa assumint que les responsabilitats són fixes, actualitzant els paràmetres.

El Pas E: L'Avaluació de la Responsabilitat: Calculem la probabilitat posterior de que el punt n pertanyi al clúster k, donats els paràmetres actuals $\theta^{(t)}$. Anomenem a aquest valor **responsabilitat** $\gamma(z_{nk})$:

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \frac{\pi_k^{(t)} \mathcal{N}(\mathbf{x}_n|\mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_{j=1}^{K} \pi_j^{(t)} \mathcal{N}(\mathbf{x}_n|\mu_j^{(t)}, \Sigma_j^{(t)})} \tag{5.21}$$

Això és simplement aplicar el Teorema de Bayes (vegeu Capítol 4).

El Pas M: L'Actualització dels Paràmetres: Ara tractem $\gamma(z_{nk})$ com a pesos fixos i maximitzem respecte a θ. Els resultats són intuïtius i elegants:

1. **Nous Centroides (μ_k^{new}):** És la mitjana ponderada de tots els punts, on el pes és la responsabilitat.

$$\mu_k^{(t+1)} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \quad \text{on} \quad N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{5.22}$$

2. **Noves Covariàncies (Σ_k^{new}):** És la variància ponderada respecte al nou centre.

$$\Sigma_k^{(t+1)} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k^{new})(\mathbf{x}_n - \mu_k^{new})^T \tag{5.23}$$

3. **Noves Priors (π_k^{new}):** És la fracció "efectiva" de punts assignats al clúster k.

$$\pi_k^{(t+1)} = \frac{N_k}{N} \tag{5.24}$$

Una propietat matemàtica de l'EM és que **garanteix** que la Log-Versemblança augmenti (o es mantingui igual) a cada iteració.

$$\ln p(\mathcal{D}|\theta^{(t+1)}) \geq \ln p(\mathcal{D}|\theta^{(t)})$$

Tanmateix, com K-Means, l'EM només garanteix trobar un **òptim local**, no el global. La superfície de la funció de versemblança sol ser complexa i plena de "turons".

Per entendre-ho bé: EM és repetir: endevina i reajusta

L'EM alterna dos passos. Primer diu: "amb els paràmetres actuals, de quin component sembla venir cada punt?" Aquest és el pas E. Després diu: "si aquestes pertinences fossin certes, com hauria d'actualitzar els paràmetres?" Aquest és el pas M. Repetint aquests dos passos, el model millora la versemblança.

El Sopar a les Fosques

Imagina un restaurant on han marxat els llums. Hi ha 3 taules (clústers) i molts plats bruts (dades) escampats pel terra. No saps de quina taula ha caigut cada plat. L'algorisme EM funciona així:

- **Inicialització:** Suposes on són les taules més o menys.
- **Pas E (Detectiu):** Agafes cada plat i dius: "Per on ha caigut, hi ha un 80% de possibilitats que vingui de la Taula A i un 20% de la Taula B". No t'arrisques a dir "és de la A", mantens la incertesa.
- **Pas M (Arquitecte):** Ara recalculas la posició de les taules. "Si aquests plats venien (probablement) d'aquí, la Taula A ha d'estar realment una mica més a la dreta".
- **Repetir:** Amb la nova posició de les taules, tornes a avaluar els plats.

A cada pas, l'ajust és més precís fins que taules i plats encaixen.

És fonamental entendre que K-Means no és un algorisme diferent, sinó un **cas límit** de l'EM per a GMM. Si prenem un GMM i fem que la variància de tots els clústers tendeixi a zero ($\epsilon \to 0$):

$$\Sigma_k = \epsilon \mathbf{I} \tag{5.25}$$

La responsabilitat $\gamma(z_{nk})$ esdevé binària: 1 per al clúster més proper i 0 per a la resta (perquè l'exponencial de la Gaussiana cau a zero infinitament ràpid).

- El Pas E es converteix en l'Assignació de K-Means.
- El Pas M es converteix en l'Actualització de K-Means.

Això demostra que K-Means és només un GMM "congelat" i simplificat.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal
from sklearn.datasets import make_blobs

# 1. Generar dades simples 1D per visualitzar la superfície
X, _ = make_blobs(n_samples=300, centers=[2, 7], cluster_std=0.8, random_state=42)
X = X[:, 0].reshape(-1, 1) # Ens quedem amb 1 dimensió

# Funció Log-Likelihood per a 2 Gaussianes (Fixant sigmes i pesos per simplicitat visual)
# Variem només les mitjanes mu1 i mu2
def log_likelihood(mu1, mu2, data):
    # Model simplificat: pi=0.5, sigma=1
    p1 = multivariate_normal.pdf(data, mean=mu1, cov=1)
    p2 = multivariate_normal.pdf(data, mean=mu2, cov=1)
    return np.sum(np.log(0.5 * p1 + 0.5 * p2))

mu_range = np.linspace(-2, 12, 100)      # 2. Calcular la superfície de versemblança
M1, M2 = np.meshgrid(mu_range, mu_range)
LL = np.zeros_like(M1)
```

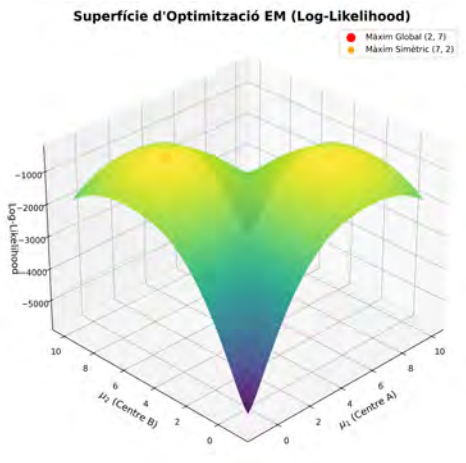


Figura 5.5: Superfície de Log-Versemblança en funció dels paràmetres μ_1 i μ_2. L'algorisme EM és un escalador que puja per aquesta muntanya. A diferència del Gradient Descent, que fa passos segons la pendent local, l'EM fa "salts" analítics cap a màxims locals de la funció auxiliar (Q-function), resultant en una convergència sovint més ràpida i estable prop del cim.

```
21
22 for i in range(len(mu_range)):
23     for j in range(len(mu_range)):
24         LL[i, j] = log_likelihood(M1[i, j], M2[i, j], X)
25
26 fig = plt.figure(figsize=(10, 8))        # 3. Plotting
27 ax = fig.add_subplot(111, projection='3d')
28 surf = ax.plot_surface(M1, M2, LL, cmap='viridis', edgecolor='none', alpha=0.8)
29 ax.set_title('Superfície de Log-Versemblança (Log-Likelihood)', fontsize=14, fontweight='bold
       ')
30 ax.set_xlabel('$\mu_1$ (Centre Clúster 1)')
31 ax.set_ylabel('$\mu_2$ (Centre Clúster 2)')
32 ax.set_zlabel('Log-Likelihood')
33
34 opt_idx = np.unravel_index(np.argmax(LL), LL.shape)      # Anotació de l'òptim
35 ax.scatter(M1[opt_idx], M2[opt_idx], LL[opt_idx], color='red', s=100, label='Mxim Global')
36 ax.legend()
37
38 plt.tight_layout()
39 plt.savefig('fig_em_surface.png', dpi=300)
40 plt.show()
```

Listing 5.5: Visualització de l'Ascens de la Versemblança (EM Algorithm)

L'EM Variacional (Variational Inference)

L'algorisme EM clàssic té un defecte: tendeix a sobreajustar (overfitting) i cal fixar K manualment. L'enginyeria moderna d'IA (ex: Latent Dirichlet Allocation per a Topic Modeling) utilitza una extensió anomenada **Variational Bayes EM**. En lloc de trobar un punt òptim (estimació puntual), s'estima tota una distribució sobre els paràmetres. Això permet que el model **apagui automàticament** els clústers innecessaris (fent $\pi_k \to 0$), descobrint el nombre real de grups per si sol. Si useu `BayesianGaussianMixture` en scikit-learn, esteu usant aquesta versió avançada.

Micro-Cas: Qui està parlant?

Un dels problemes més desafiants en processament de senyal és, donada una gravació d'una reunió, determinar "qui va parlar quan" sense saber a priori quantes persones hi ha ni qui són.

El Repte: Una gravació d'àudio de 10 minuts amb dos interlocutors (Alice i Bob) que s'interrompen.

Enginyeria de Característiques: Es divideix l'àudio en finestres de 20ms i s'extreuen vectors MFCC (Mel-Frequency Cepstral Coefficients). Tenim un núvol de punts en $\mathbb{R}^{13}$.

Aplicació EM:

1. **Inicialització:** S'assumeix $K = 2$.
2. **Pas E:** Per a cada fragment de 20ms, calculem la probabilitat que sigui Alice o Bob. Els fragments de silenci o soroll tindran probabilitats properes al 0.5 (incertesa).
3. **Pas M:** Actualitzem el model acústic (la "veu") d'Alice i Bob usant els fragments on estem segurs.

L'EM permet separar les veus fins i tot quan se solapen lleugerament, ja que modela la veu com una distribució de probabilitat i no com una esfera rígida.

5.6 Selecció de Models: Quants grups hi ha?

En classificació, sabem quantes classes hi ha (les que hi hagi al dataset d'entrenament). En clustering, el nombre de grups K és un **Hiperparàmetre** que hem de fixar a priori (en K-Means i GMM). Però, com sabem si la nostra clientela es divideix en 3 segments o en 5? La resposta intuïtiva ("prova diversos K i tria el que tingui menys error") és una trampa mortal matemàtica.

5.6.1 La Trampa de l'Overfitting en Clustering

Definim l'error d'entrenament (Distorsió o Inèrcia) com $J(K)$. Si augmentem el nombre de clústers K, la distorsió **sempre** disminueix (o es manté igual). És una funció monòtona decreixent.

$$J(K+1) \leq J(K) \tag{5.26}$$

Imaginem que tenim una solució amb K clústers. Si dividim un d'aquests clústers en dos sub-clústers (passant a $K+1$), els nous punts estaran, per definició, més a prop (o igual) dels seus nous centres locals que de l'antic centre global. Si portem aquest raonament al límit, quan $K = N$ (tants clústers com dades), cada dada és el seu propi clúster. La distorsió és exactament zero:

$$J(N) = \sum_{n=1}^{N} ||\mathbf{x}_n - \mathbf{x}_n||^2 = 0$$

Aquesta solució és perfecta matemàticament, però inútil semànticament. Hem memoritzat les dades, no hem après cap estructura. Aquest és el fenomen de l'**Overfitting** aplicat al clustering. Per trobar el K òptim, hem de trobar un equilibri entre **Bondat d'Ajust** (Goodness of Fit) i **Complexitat del Model**.

5.6.2 Mètode 1: La Regla del Colze (The Elbow Method)

Aquesta és l'heurística més utilitzada en enginyeria per la seva simplicitat visual, associada habitualment a K-Means. Consisteix a graficar la corba $J(K)$ i buscar el punt on el guany marginal de afegir un nou clúster cau dràsticament. Aquest punt d'inflexió s'anomena "colze".

Matemàticament, busquem el punt on la segona derivada (discreta) és màxima:

$$K^* = \arg\max_K |J(K-1) - 2J(K) + J(K+1)| \tag{5.27}$$

Tot i ser molt popular, el mètode del colze és **subjectiu**. En dades reals, la corba sol ser suau i no presenta un angle agut, deixant la decisió a una interpretació dubtosa.

Explicació per a Dummies: Talles de Samarretes

Imagina que vols obrir una botiga de roba. Quantes talles de samarretes has de fabricar?

- **Opció A ($K = 1$, Talla Única):** Molt barat de fabricar (complexitat baixa), però molta gent estarà descontenta perquè li anirà gran o petita (error alt).
- **Opció B ($K = N$, Sastrería a Mida):** Fas una samarreta exacta per a cada client. Tothom està perfecte (error zero), però els costos de fabricació són infinits (complexitat màxima).
- **El Colze ($K = 5$, XS-XL):** Passar d'1 talla a 2 millora moltíssim la satisfacció. De 2 a 3 també. Però passar de 100 talles a 101 ja no aporta gairebé res. El "Colze" és el punt on deixes de guanyar prou satisfacció per justificar afegir una nova màquina de tallar tela.

5.6.3 Mètode 2: Criteris d'Informació (AIC i BIC)

Quan treballem amb models probabilístics com GMM, podem ser molt més rigorosos. Utilitzem la teoria de la informació per penalitzar el nombre de paràmetres lliures del model. Volem maximitzar la Log-Versemblança ($\ln \hat{L}$), però restem un terme de penalització:

1. AIC (Akaike Information Criterion): Proposat per Hirotugu Akaike el 1974, busca el model que millor prediu dades futures.

$$AIC = 2k - 2\ln(\hat{L}) \tag{5.28}$$

On k és el nombre de paràmetres estimables del model.

2. BIC (Bayesian Information Criterion): Proposat per Gideon Schwarz el 1978, deriva de principis Bayesians i busca identificar el **model veritable** que ha generat les dades (si existeix).

$$BIC = k\ln(N) - 2\ln(\hat{L}) \tag{5.29}$$

On N és el nombre de mostres (dades).

Comparativa ente BIC i AIC:

- El terme de penalització del BIC és $k\ln(N)$, mentre que el de l'AIC és $2k$.
- Com que normalment $\ln(N) > 2$ (per a $N > 7$), el **BIC penalitza més fortament la complexitat**.
- **Conseqüència pràctica:** L'AIC tendeix a suggerir models més complexos (més clústers, risc lleu d'overfitting). El BIC tendeix a suggerir models més simples (menys clústers, risc d'underfitting). En la indústria, sovint es prefereix el BIC per obtenir models més parsimoniosos i interpretables.

Per aplicar aquestes fórmules, una persona experta en enginyeria ha de saber comptar quants paràmetres té el seu model. Per a un GMM amb K components en d dimensions i covariància *full*:

1. **Mitjanes (μ_k):** $K \times d$ paràmetres.
2. **Pesos (π_k):** $K - 1$ paràmetres (perquè sumen 1).
3. **Covariàncies (Σ_k):** Una matriu simètrica $d \times d$ té $d(d+1)/2$ valors únics. Per tant, $K \times \frac{d(d+1)}{2}$.

$$k_{GMM} = K\left(d + \frac{d(d+1)}{2} + 1\right) - 1 \tag{5.30}$$

Aquest creixement quadràtic amb d explica per què el BIC es dispara ràpidament si afegim clústers en alta dimensió.

Per acabar, cal comentar que hi ha altres mètodes més enllà del BIC. Un d'ells és la **Silhouette Score**: És una mètrica que no depèn del model, sinó de la geometria. Per a cada punt i, calcula:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{5.31}$$

On $a(i)$ és la distància mitjana als punts del seu propi clúster (cohesió) i $b(i)$ és la distància mitjana als punts del clúster veí més proper (separació). El valor va de -1 a +1. Un valor mitjà alt indica clústers ben definits. A diferència de la Inèrcia, el Silhouette no sempre millora en augmentar K, per la qual cosa el seu màxim pot indicar el K òptim.

5.6.4 Implementació de La Recerca del Mínim

A continuació, simularem un dataset amb $K = 4$ clústers reals i veurem si les mètriques (Inèrcia, AIC, BIC) són capaces de recuperar aquest nombre.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture

# 1. Generar Dades (Ground Truth: K=4)
X, y = make_blobs(n_samples=500, centers=4, cluster_std=0.60, random_state=0)

k_range = range(1, 10)    # 2. Rangs de prova
inertias = []
bics = []
aics = []

for k in k_range:         # 3. Bucle d'Entrenament
    # K-Means per a Elbow Method
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X)
    inertias.append(kmeans.inertia_)

    # GMM per a AIC/BIC
    gmm = GaussianMixture(n_components=k, random_state=42)
    gmm.fit(X)
    bics.append(gmm.bic(X))
    aics.append(gmm.aic(X))

# 4. Visualització Dual
fig, ax1 = plt.subplots(figsize=(10, 6))

# Eix Esquerre: Inèrcia (K-Means)
color = 'tab:red'
ax1.set_xlabel('Nombre de Clústers ($K$)', fontsize=12)
```

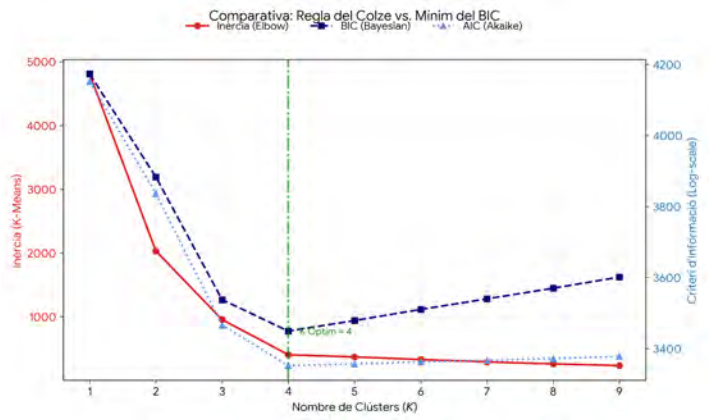


Figura 5.6: Comparativa de mètodes de selecció. (Eix vermell) La Inèrcia mostra un "colze" suau al voltant de $K = 4$, però continua baixant. (Eix blau) El BIC i l'AIC mostren una paràbola clara amb un **mínim absolut** a $K = 4$. Això demostra que els Criteris d'Informació són menys ambigus que el mètode del colze.

```
ax1.set_ylabel('Inèrcia (K-Means)', color=color, fontsize=12)
ax1.plot(k_range, inertias, color=color, marker='o', linestyle='--', label='Inèrcia (Elbow)')
ax1.tick_params(axis='y', labelcolor=color)

# Eix Dret: BIC/AIC (GMM)
ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Score d\'Informació (GMM)', color=color, fontsize=12)
ax2.plot(k_range, bics, color='navy', marker='s', label='BIC (Bayesian)')
ax2.plot(k_range, aics, color='cornflowerblue', marker='^', linestyle=':', label='AIC (Akaike)
    ')
ax2.tick_params(axis='y', labelcolor=color)

# Decoració
plt.title("Selecció de Models: Elbow vs. BIC/AIC", fontsize=14, fontweight='bold')
plt.axvline(x=4, color='green', linestyle='-.', alpha=0.5, label='K Real = 4')
fig.legend(loc="upper right", bbox_to_anchor=(1,1), bbox_transform=ax1.transAxes)
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('fig_model_selection.png', dpi=300)
plt.show()
```

Listing 5.6: Cerca del nombre òptim de clústers (Model Selection)

Consells de programació: No validis un clustering només amb AIC/BIC

AIC i BIC ajuden a penalitzar models massa complexos, però no saben si els grups trobats tenen sentit per al problema real. Després de triar K, comprova sempre quatre coses: estabilitat, mida dels grups, interpretabilitat i utilitat per a la presa de decisions. Un mínim de BIC no és, per si sol, una bona segmentació.

Imprimeix també quants punts hi ha a cada clúster. Si un grup conté el 95% de les dades i la resta són grups minúsculs, probablement K no és una bona elecció. I si petites variacions de les dades o de la llavor aleatòria canvien molt els grups, el resultat és fràgil. No venguis com a segments fiables agrupacions que no són estables.

L'Estadística Gap (Gap Statistic)

El mètode del colze és subjectiu. Els investigadors de Stanford (Tibshirani et al.) van proposar el **Gap Statistic**. La idea és comparar la inèrcia del teu clustering W_k amb la inèrcia esperada d'una distribució aleatòria uniforme (sense estructura) W_k^{ref}.

$$Gap(k) = \mathbb{E}[\ln(W_k^{ref})] - \ln(W_k) \tag{5.32}$$

Busquem el K que maximitza aquest Gap. Aquesta tècnica quantifica formalment "quant millor és el meu clustering comparat amb el pur atzar".

Micro-Cas: Quantes races hi ha? (La resposta matemàtica)

Un dels casos més fascinants de clustering és l'anàlisi de l'estructura genètica de la humanitat. Programari com *STRUCTURE* (basat en models Bayesians tipus GMM) s'utilitza per determinar K poblacions ancestrals. Els biòlegs no pregunten per "races", sinó per grups d'ancestralitat (K). En analitzar l'ADN de milers d'humans globals:

- **Amb $K = 2$:** L'algorisme separa Àfrica vs. La resta del món (la divisió genètica més antiga).
- **Amb $K = 5$:** Els clústers s'alineen aproximadament amb els continents (Àfrica, Euràsia Occidental, Àsia Oriental, Oceania, Amèrica).
- **Amb $K = 7$:** Apareixen sub-poblacions com els Kalash del Pakistan, un grup aïllat genèticament.

Lliçó: En molts problemes reals, **no existeix un únic K correcte**. L'estructura és jeràrquica. El BIC pot donar un mínim, però sovint l'equip d'enginyeria ha d'explorar múltiples nivells de granularitat ($K = 2, 5, 7$) per entendre la història completa de les dades. El "millor" model depèn de la pregunta biològica que vulguem respondre.

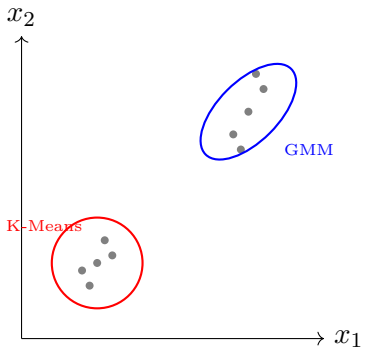
5.7 Conclusions i Debat

En aquest cinquè capítol hem passat de l'aprenentatge supervisat, on disposem d'etiquetes, a l'aprenentatge no supervisat, on hem d'inferir estructures sense una resposta correcta externa. Hem passat de la classificació supervisada a la descoberta de patrons latents:

1. **De la Rigidesa a la Flexibilitat:** Hem vist que K-Means és eficient, però imposa una geometria rígida: assignacions dures i regions de Voronoi basades habitualment en distància euclidiana. Els GMM, en canvi, ens han permès ajustar models de densitat més flexibles, amb components el·líptics que poden tenir orientacions, mides i pesos diferents.

2. **L'algorisme EM com a pont:** Hem vist que l'Expectation-Maximization permet gestionar assignacions incertes mitjançant responsabilitats. Sota condicions específiques —components gaussianes amb covariàncies esfèriques iguals i assignacions cada vegada més dures— K-Means es pot entendre com un cas límit d'aquest marc probabilístic. Aquesta connexió ajuda a veure K-Means i GMM com dos punts d'un mateix continu: de particions rígides a pertinences probabilístiques.

3. **Complexitat vs. informació:** Hem après que, en absència d'etiquetes, cal evitar triar simplement el model que millor s'ajusta a les dades. Criteris com AIC i BIC penalitzen la complexitat i són especialment útils en models probabilístics com els GMM. Tot i així, no són l'únic criteri: l'estabilitat dels grups, la interpretabilitat i la utilitat per al domini també han de formar part de la decisió.



Evolució del Paradigma:
1. No Supervisat: $\mathcal{D} = \{\mathbf{x}_n\}$
2. Distància Euclidiana → Mahalanobis
3. Hard Assignment → Soft Responsibility
4. Optimització EM ($\mathcal{Q}$-function)

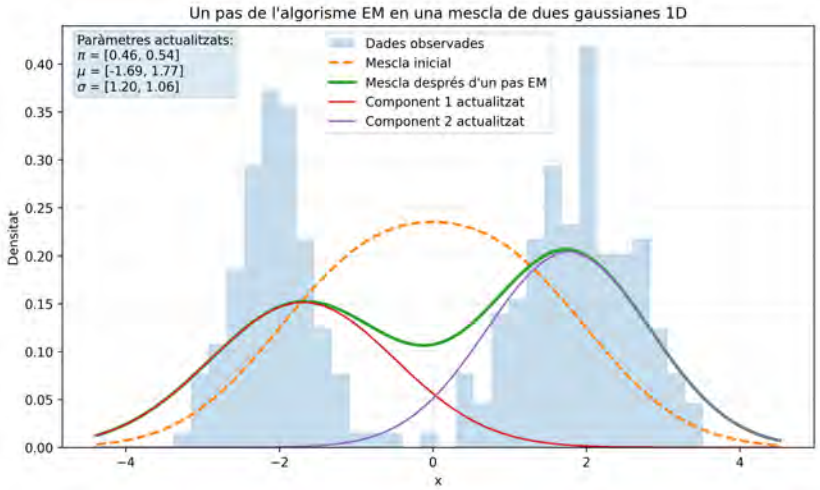
Figura 5.7: El paradigma del Capítol 5: De la partició rígida de l'espai a la modelització de densitats latents.

4. **La Variable Latent:** Hem entès que molts fenòmens no són observables directament, però poden deixar patrons en les dades. Els models amb variables latents intenten estimar una estructura oculta compatible amb aquests patrons i amb les hipòtesis del model.

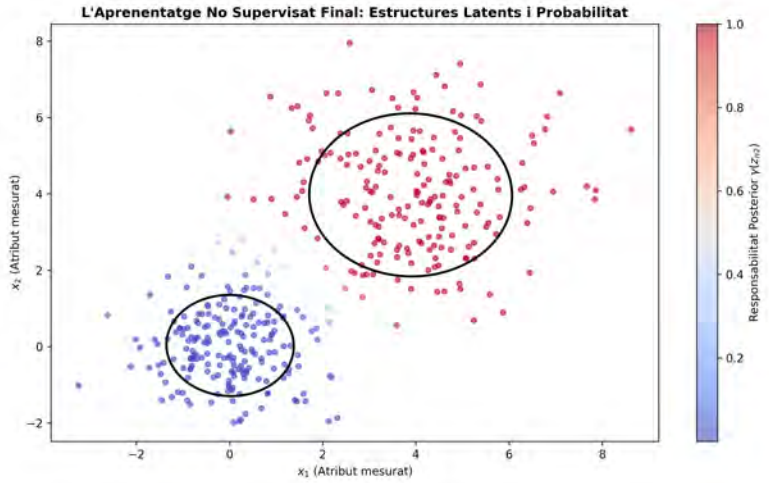
Demostració final: un pas EM en una dimensió

Per tancar el capítol, implementem una versió mínima d'un pas EM en una dimensió. L'objectiu no és substituir les implementacions professionals, sinó fer visible la mecànica del procés: estimem responsabilitats latents (pas E) i després actualitzem els paràmetres del model (pas M).

```python
import numpy as np

def normal_pdf(x, mu, sigma):
    """Densitat d'una normal univariant N(mu, sigma^2)."""
    eps = 1e-12
    sigma = np.maximum(sigma, eps)
    coef = 1.0 / (np.sqrt(2 * np.pi) * sigma)
    return coef * np.exp(-0.5 * ((x - mu) / sigma)**2)

def em_1d_step(x, pi, mu, sigma):
    """Un pas EM per a una mescla de dues gaussianes 1D."""
    eps = 1e-12

    # Pas E: calcular responsabilitats
    r1 = pi[0] * normal_pdf(x, mu[0], sigma[0])
    r2 = pi[1] * normal_pdf(x, mu[1], sigma[1])
    denom = r1 + r2 + eps
    gamma = r1 / denom  # responsabilitat del component 1

    # Pas M: actualitzar paràmetres
    N1 = np.sum(gamma)
    N2 = len(x) - N1

    pi_new = np.array([N1 / len(x), N2 / len(x)])
    mu_new = np.array([
        np.sum(gamma * x) / (N1 + eps),
        np.sum((1 - gamma) * x) / (N2 + eps)
    ])
    sigma_new = np.sqrt(np.array([
        np.sum(gamma * (x - mu_new[0])**2) / (N1 + eps),
        np.sum((1 - gamma) * (x - mu_new[1])**2) / (N2 + eps)
```



(a) Un pas de l'algorisme EM en una mescla de dues gaussianes 1D. L'histograma mostra les dades observades, mentre que les corbes representen la mescla inicial i la mescla actualitzada després d'un pas EM, juntament amb els dos components actualitzats.



(b) Visualització de la convergència d'un model GMM. El gradient de color representa la responsabilitat posterior $\gamma_{n2} = P(z_n = 2 \mid \mathbf{x}_n)$, és a dir, el grau de pertinença probabilística de cada mostra al segon component. Les el·lipses representen les regions de densitat associades a les matrius de covariància apreses.

Figura 5.8: **Procés i resultat final de l'algorisme EM en models de mescla**. A la part superior es mostra un pas de l'actualització EM en una mescla de dues gaussianes 1D. A la part inferior es presenta la síntesi final de la convergència del model GMM, amb les responsabilitats posteriors i les regions de confiança apreses.

```
    ]))

    return pi_new, mu_new, sigma_new, gamma

# Conclusio: L'EM no cerca la veritat, cerca la maxima versemblanca.
```

Listing 5.7: Resum d'Enginyeria: El cicle EM en acció

5.7.1 Debat Ètic: El Perill del Perfilat sense "Etiquetes"

Si en els capítols anteriors el biaix provenia d'etiquetes humanes injustes, en l'aprenentatge no supervisat el biaix és més insidiós: és un **biaix d'estructura**. L'algorisme trobarà grups basant-se en qualsevol correlació, sense importar si aquesta correlació és èticament lícita o un reflex d'una injustícia estructural.

1. El Perfilat Invisible i la Discriminació Indirecta

L'ús de GMM o K-Means per a la segmentació social permet a les empreses i governs crear grups de persones sense utilitzar variables protegides (raça, religió). No obstant això, l'algorisme pot crear un clúster que actuï com a proxy fort, encara que indirecte, d'un grup vulnerable.

Dilema: El Clustering com a eina de segregació

Imagineu un banc que agrupa la clientela segons comportament de despesa i geolocalització. Un clúster pot acabar concentrant persones d'un barri, horari laboral o perfil socioeconòmic concret. Si després el banc restringeix productes financers a aquest clúster, pot estar generant discriminació indirecta sense haver utilitzat explícitament cap variable protegida. El problema no és només el clustering, sinó l'ús que es fa dels segments resultants.

2. La Reificació dels Clústers: Quan el model crea la realitat

Hi ha un risc sociològic anomenat "Reificació": tractar un clúster abstracte com si fos una entitat real i immutable. En màrqueting o política, un cop l'algorisme diu "existeixen 4 tipus de votants", els humans comencen a dissenyar campanyes per a aquests 4 tipus, forçant la realitat a ajustar-se al model.

3. El Biaix Latent en les Dades de Salut

En medicina genòmica, s'usa l'EM per trobar sub-poblacions. Si una base de dades genòmica conté majoritàriament mostres d'un grup poblacional, el model pot aprendre subestructures fines per a aquest grup i agrupar de manera massa grossa o imprecisa la resta de poblacions. Així, les medicines personalitzades es dissenyaran per als clústers detectats (els europeus), deixant a les minories amb tractaments basats en una "mitjana" que no els representa.

5.7.2 Preguntes per a l'Aula

1. Filosofia: Existeix la "Veritat" sense etiquetes?

Hem vist que el BIC i el mètode del colze ens ajuden a triar K.

- **Pregunta:** Si el BIC diu que $K = 3$ és l'òptim, però la persona experta en domini diu que "sap" que n'hi ha 5, qui té raó?
- **Discussió:** En l'aprenentatge no supervisat, el model no és una representació de la veritat, sinó una **eina de compressió**. Un model amb $K = 3$ resumeix més; un amb $K = 5$ detalla més. L'equip d'enginyeria ha de decidir el nivell de granularitat segons l'aplicació, no segons una "veritat" inexistent.

2. Enginyeria: El perill de la Isotropia

Recordeu que K-Means assumeix que els clústers són cercles de la mateixa mida.

- **Escenari:** Esteu agrupant dades de transmissió d'un virus. Hi ha un brot petit molt concentrat en un hospital, i un brot gran molt dispers en una ciutat.
- **Pregunta:** Què farà K-Means?
- **Anàlisi:** El K-Means "robarà" punts de la ciutat i els donarà a l'hospital per intentar que els dos cercles tinguin el mateix radi. Això pot portar a una política de salut pública errònia. Com ho solucionaríeu? (Pista: GMM amb covariància "full").

3. Ètica: L'algorisme "Ceg" davant la Història

Un algorisme de clustering agrupa alumnat segons el seu rendiment i hores d'estudi. Troba un grup de "Baix Rendiment".

- **Fet:** L'algorisme no sap que l'alumnat d'aquest grup han de treballar 40 hores setmanals per pagar-se la carrera.
- **Debat:** És lícit que una universitat retiri beques basant-se en aquest clustering? L'algorisme és "objectiu" perquè no coneix la situació personal, o és "cruel" precisament per la mateixa raó? Com a professionals de l'enginyeria, hauríem d'introduir variables socioeconòmiques a l'espai $\mathbb{R}^d$ per "ajudar" el clustering?

4. Visió Crítica: La tirania del centroide

En agrupar persones, el centroide μ_k es converteix en l'estàndard de la norma.

- **Reflexió:** Tothom qui no està prop del centroide és vist com a "soroll" o "outlier". En un món dominat per l'aprenentatge no supervisat, estem eliminant la diversitat individual en favor de la mitjana del clúster?

Consell Final: La verificació externa

Mai doneu per bo un resultat de clustering basat només en el BIC o l'AIC. Cal **Validació Externa**: guardem una variable que el model **no ha vist** (ex: el gènere o el resultat d'una enquesta) i mirem si els clústers que han "emergit" sols tenen alguna correlació amb aquesta variable. Si el clustering no és capaç de predir cap variable externa real, és que només heu agrupat soroll.

Tancament docent

Problemes de consolidació.

1. Donat un conjunt petit de punts en dues dimensions i dos centroides inicials, executar manualment una iteració de K-Means: assignar cada punt al centroide més proper i recalcular els nous centroides.
2. A partir de tres resultats de K-Means amb valors diferents de K, interpretar la inèrcia, detectar el possible punt de colze i discutir per què aquesta decisió pot no tenir una resposta única.
3. Comparar un cas on K-Means funciona bé amb un cas on falla per geometria: grups allargats, densitats diferents o formes no esfèriques. Explicar per què la distància euclidiana i l'assumpció d'esfericitat poden imposar una estructura artificial.
4. Donat un punt situat entre dos grups, comparar l'assignació dura de K-Means amb la responsabilitat probabilística d'un GMM. Interpretar què vol dir que una mostra tingui, per exemple, responsabilitat 0.65 en un component i 0.35 en un altre.

Tancament docent *(continuació)*

5. A partir d'una taula amb valors d'AIC i BIC per a diferents nombres de components, decidir quin model seria més raonable i justificar la decisió en termes de compromís entre ajust, complexitat i interpretabilitat.
6. Analitzar un cas de segmentació social o comercial i discutir quin risc apareix si els clústers es tracten com a categories naturals, estables i accionables sense revisió humana.

Criteris breus de resolució. Una bona resolució hauria de deixar clar que el clustering no parteix d'etiquetes, sinó d'una hipòtesi geomètrica o probabilística sobre l'estructura de les dades. En K-Means, cal aplicar correctament la distància, l'assignació al centroide i l'actualització de centroides, però també cal saber detectar quan l'assumpció de grups compactes i esfèrics és inadequada. En GMM, cal entendre que les responsabilitats no són etiquetes fixes, sinó graus de pertinença probabilística. En selecció de models, cal evitar escollir sempre el model amb millor ajust i justificar el compromís entre explicació, simplicitat i ús pràctic. En qualsevol resposta sobre clustering, cal evitar parlar dels grups com si fossin classes naturals. Una bona resolució ha d'indicar sempre quina representació, quina mètrica, quin valor de K i quin criteri de validació s'han utilitzat.

Connexió amb pràctica/laboratori. Aquest capítol es desplega en una pràctica d'aprenentatge no supervisat. L'alumnat pot aplicar K-Means i GMM sobre un dataset bidimensional o de segmentació de clients, visualitzar fronteres de Voronoi i el·lipses de covariància, comparar assignacions dures i responsabilitats suaus, provar diferents valors de K, calcular criteris com AIC/BIC i redactar una decisió final sobre quin model utilitzaria i amb quines cauteles.

Capítol 6

Incertesa, Seqüències i Ensembles

Fins ara, hem perseguit la quimera del "model perfecte" i de l'algorisme "mestre". Hem buscat l'hiperplà òptim (SVM) o el clúster ideal (GMM). erò l'enginyeria del món real ens ensenya una lliçó d'humilitat: quan les dades són sorolloses, canviants o incompletes, sovint cal anar més enllà d'un únic model i pensar en robustesa, diversitat i incertesa.

En aquest capítol, aprendrem a combinar molts models imperfectes però diversos per construir predictors més robustos (*ensembles*), a gestionar dades que canvien amb el temps (seqüències) i, el més important, a quantificar el que no sabem (incertesa).

Guia d'activació docent

Repte inicial. Un sistema de diagnòstic rep una imatge de baixa qualitat. Un model dona una predicció segura, però altres models discrepen. La pregunta inicial és: *quan hem de confiar, quan hem d'agregar models i quan és més responsable abstenir-se?*

Checkpoints conceptuals.

1. Explicar com bootstrap, bagging i Random Forest redueixen variància.
2. Distingir incertesa aleatòria, epistèmica i fora de distribució.

Errors típics provocats.

1. Pensar que qualsevol ensemble millora qualsevol model individual.
2. Confondre una probabilitat alta amb confiança calibrada.

Microcas professional. Detecció de frau: Random Forest pot agregar patrons diversos, però casos amb alta discrepància o fora de distribució han d'anar a revisió humana.

Connexió ètica o d'enginyeria. Un sistema responsable no només prediu: també estima quan no sap prou i defineix què fer amb aquesta incertesa.

6.1 La Saviesa dels Grups: Bootstrap i Bagging

El 1906, Francis Galton va observar que la mitjana de moltes estimacions independents del pes d'un bou era sorprenentment propera al valor real. Cap individu encertava perfectament, però l'agregació col·lectiva reduïa l'error. Aquesta intuïció inspira els ensembles.

6.1.1 Per què la mitjana redueix l'error?

Per entendre l'enginyeria darrere dels ensembles, hem de tornar a l'estadística bàsica. Suposem que tenim M models diferents $h_1(\mathbf{x}), \dots, h_M(\mathbf{x})$ per predir una variable y. Suposem que cada model té un error ϵ_i amb variància σ^2 i que els errors tenen mitjana zero ($\mathbb{E}[\epsilon_i] = 0$). Si fem la predicció final fent la mitjana simple:

$$H(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} h_i(\mathbf{x}) \tag{6.1}$$

Quina és la variància (l'error esperat) d'aquest "super-model" H?

Cas 1: Models totalment independents: Si els errors dels models són **independents i idènticament distribuïts (i.i.d.)** (és a dir, $\text{Cov}(\epsilon_i, \epsilon_j) = 0$ per a $i \neq j$), la variància de la mitjana és:

$$\text{Var}(H) = \text{Var}\left(\frac{1}{M}\sum \epsilon_i\right) = \frac{1}{M^2}\sum \text{Var}(\epsilon_i) = \frac{1}{M^2}(M \cdot \sigma^2) = \frac{\sigma^2}{M} \tag{6.2}$$

Amb 100 models realment independents, la variància es podria reduir en un factor de 100. Aquesta és la intuïció estadística central dels ensembles: la mitjana pot cancel·lar part del soroll individual.

Cas 2: Models Correlacionats (La Realitat): En la pràctica, els models s'entrenen amb les mateixes dades, per la qual cosa els seus errors estan correlacionats positivament ($\rho > 0$). La fórmula general de la variància de la suma és:

$$\text{Var}(H) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2 \tag{6.3}$$

Això ens diu dues coses crítiques:

1. Si $M \to \infty$, el segon terme desapareix.
2. L'error residual és $\rho\sigma^2$. **No podem reduir l'error per sota de la correlació entre models.**

Per tant, l'objectiu dels ensembles és construir models que siguin el més diversos (descorrelacionats) possible, mantenint un cert nivell de precisió individual.

Per entendre-ho bé: Molts errors petits es compensen

Si un model s'equivoca una mica cap amunt i un altre una mica cap avall, la mitjana pot cancel·lar part de l'error. Però això només funciona si els models no s'equivoquen tots igual. Per això els ensembles necessiten diversitat: sense diversitat, només repetim el mateix error moltes vegades.
Per això, veurem a continuació com el bagging introdueix diversitat mitjançant mostres bootstrap, i Random Forest afegeix encara més diversitat seleccionant subconjunts aleatoris de característiques en cada divisió.

6.1.2 Bootstrap: Creant diversitat del no-res

Com podem crear M models diversos si només tenim un dataset $\mathcal{D}$? Si entrenem el mateix algorisme M vegades amb les mateixes dades, obtindrem M models idèntics ($\rho = 1$), i l'ensemble no servirà de res.

La solució és el **Bootstrap** (inventat per Bradley Efron el 1979). Consisteix a generar "nous" datasets sintètics $\mathcal{D}_1, \ldots, \mathcal{D}_M$ mitjançant **mostreig amb reemplaçament** del dataset original.

Si tenim un dataset de mida N i generem una mostra Bootstrap de mida N:

- La probabilitat de triar una instància específica x_i és $1/N$.
- La probabilitat de **no** triar x_i en un intent és $1 - 1/N$.
- La probabilitat de **no** triar x_i després de N intents (tot el procés) és:

$$P(\text{no triat}) = \left(1 - \frac{1}{N}\right)^N \tag{6.4}$$

Quan N és gran, aquest límit és una de les constants més famoses en matemàtiques:

$$\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^N=\frac{1}{e}\approx 0.368 \tag{6.5}$$

Cada mostra bootstrap conté aproximadament un 63.2% d'exemples únics del dataset original. El 36.8% restant dels exemples originals, de mitjana, no apareix en aquella mostra bootstrap i queda com a conjunt *out-of-bag* per a aquell model. Això garanteix que cada model h_i veu una visió lleugerament "distorsionada" de la realitat, introduint la diversitat necessària ($\rho < 1$) perquè l'ensemble funcioni.

6.1.3 Bagging (Bootstrap Aggregating)

L'algorisme **Bagging** (acrònim de *Bootstrap Aggregating*, proposat per Leo Breiman el 1996) és l'aplicació directa d'aquesta teoria:

1. **Bootstrap:** Generar M datasets $\mathcal{D}_1,\ldots,\mathcal{D}_M$ amb reemplaçament.
2. **Entrenament:** entrenar un model d'alta variància, habitualment un arbre de decisió profund, en cada D_i. La idea és que cada model sigui sensible a la mostra bootstrap, de manera que l'agregació pugui reduir la variància.
3. **Aggregating:**
   - *Regressió:* Mitjana de les prediccions.
   - *Classificació:* Vot per majoria (*Hard Voting*) o mitjana de probabilitats (*Soft Voting*).

Bagging no millora models dolents

Bagging redueix la **Variança**, no el **Biaix**.

- Si utilitzeu Bagging amb **Regressió Lineal** (que té biaix alt i variança baixa), no guanyareu res. La mitjana de 100 línies rectes similars és... una línia recta similar.
- Bagging necessita models **inestables** (com arbres profunds o xarxes neuronals) per funcionar. Necessita que els "membres del comitè" tinguin opinions diferents. Si tots opinen igual, el comitè és inútil.

Imagineu la pregunta final del concurs "Qui vol ser milionari"

- **Model Únic:** Truques a la teva amistat "experta". Si ella ho sap, guanyes. Si té un mal dia o la pregunta és sobre un tema que ella no domina (biaix), perds. La persona amiga té "alta variança" (encerta molt o falla molt).
- **Ensemble (Bagging):** Demanes el comodí del públic. 100 persones voten.
- Alguns no en tenen ni idea (soroll). Altres saben una mica. Pocs ho saben segur.
- En fer la mitjana (votar), els errors aleatoris es cancel·len. Si el 70% del públic diu "B", és molt probable que sigui "B".

Bagging és, literalment, democratitzar la intel·ligència artificial.

El millor exemple per veure el poder del Bagging és utilitzar Arbres de Decisió, que són inestables (si canvies una sola dada d'entrenament, tot l'arbre pot canviar).

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

```

```python
# 1. Generar dades sorolloses i no lineals
X, y = make_moons(n_samples=300, noise=0.30, random_state=42)

# 2. Entrenar models
# Model A: Un sol arbre molt profund (Overfitting, Alta Variança)
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X, y)

# Model B: Bagging de 100 arbres
bagging = BaggingClassifier(
    estimator=DecisionTreeClassifier(),
    n_estimators=100,
    max_samples=0.6, # Cada arbre veu el 60% de les dades (Bootstrap)
    bootstrap=True,
    random_state=42,
    n_jobs=-1
)
bagging.fit(X, y)

# 3. Plotting Boundaries
def plot_decision_boundary(clf, X, y, ax, title):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                         np.arange(y_min, y_max, 0.02))

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    ax.contourf(xx, yy, Z, alpha=0.3, cmap='coolwarm')
    ax.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', cmap='coolwarm', s=30)
    ax.set_title(title, fontsize=14, fontweight='bold')

fig, axs = plt.subplots(1, 2, figsize=(14, 6))

plot_decision_boundary(tree, X, y, axs[0], "Arbre Unic (Alta Variança)")
plot_decision_boundary(bagging, X, y, axs[1], "Bagging: 100 Arbres (Frontera Suau)")

plt.tight_layout()
plt.savefig('fig_bagging_smoothness.png', dpi=300)
plt.show()
```

Listing 6.1: Visualització de l'efecte suavitzador del Bagging

Micro-Cas: El Sistema de Predicció per Conjunts (EPS)

Si mireu l'app del temps i diu "30% de probabilitat de pluja a les 17h", d'on surt aquest número? No surt d'un sol model. La física de l'atmosfera és caòtica (Efecte Papallona): petits errors en la mesura inicial de temperatura creixen exponencialment.

El Centre Europeu de Prediccions Meteorològiques (ECMWF) utilitza Bagging físic:

1. **Pertorbació Inicial (Bootstrap):** No executen el simulador una vegada. L'executen 51 vegades.
2. En la simulació 1, la temperatura actual a Barcelona és 15.0°C.
3. En la simulació 2, introdueixen un petit error aleatori: 15.1°C.
4. En la simulació 3, 14.9°C, etc.
5. **Simulació:** Deixen córrer els 51 models físics durant 10 dies.

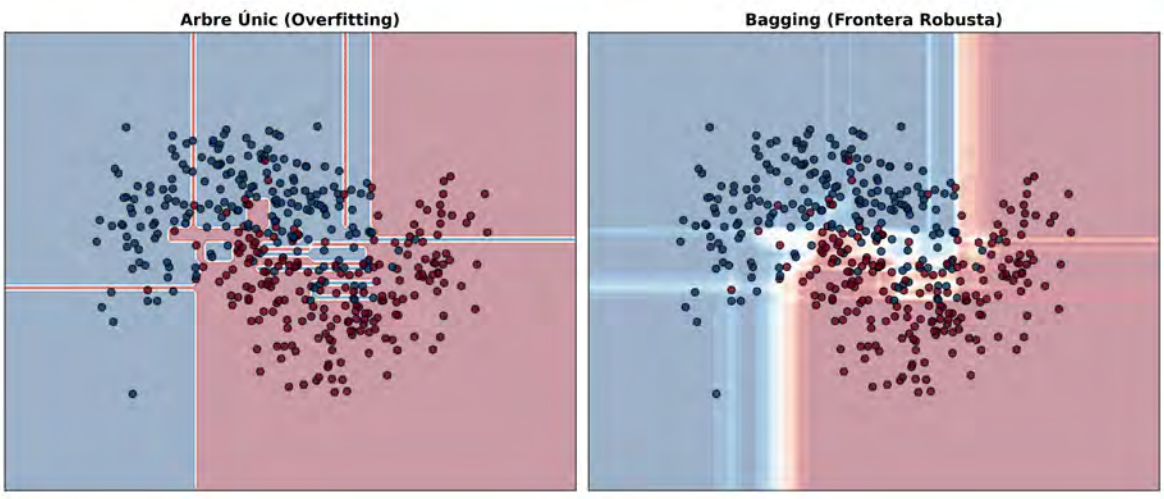


Figura 6.1: Comparativa de Fronteres de Decisió. A l'esquerra, un sol arbre de decisió memoritza cada punt de soroll, creant una frontera irregular (Overfitting). A la dreta, l'Ensemble Bagging suavitza aquestes irregularitats: la mitjana de 100 fronteres és una frontera estable que captura l'estructura i generalitza millor.

El Sistema de Predicció per Conjunts (EPS) *(continuació)*

6. **Aggregating:** Miren els resultats per a demà a les 17h. Si en 15 de les 51 simulacions plou, la probabilitat és $15/51 \approx 29.4\%$.

Els famosos "Spaghetti Plots" dels meteoròlegs són la visualització d'aquestes 51 línies divergents. Bagging salva vides permetent quantificar la incertesa d'un huracà.

6.2 Random Forests: Descorrelació d'Arbres

Al principi del capítol hem vist que la variància d'un ensemble (la mitjana de M models) ve donada per:

$$\text{Var}(H) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2 \tag{6.6}$$

On ρ és la correlació mitjana entre els errors dels models individuals.

Ara bé, imagineu un dataset mèdic on hi ha una "super-variable" (ex: "Té Càncer?"). En el 99% de les mostres Bootstrap, aquesta variable serà tan predictiva que tots els arbres la triaran com a node arrel. Això fa que tots els arbres s'assemblin ($\rho \approx 1$), anul·lant el benefici de fer la mitjana. Per això, l'any 2001, Leo Breiman va proposar una modificació brillant per trencar aquesta correlació: els **Random Forests**. La idea és fer els arbres "més ximples" individualment, però més diversos col·lectivament.

Un Random Forest introdueix aleatorietat en dues dimensions:

1. **Aleatorietat de Dades (Bagging):** Cada arbre s'entrena amb una mostra Bootstrap diferent ($\mathcal{D}_i$).
2. **Aleatorietat de Característiques (Feature Bagging):** A **cada node** de l'arbre, quan l'algorisme busca la millor partició, no se li permet mirar totes les d característiques disponibles. Només se li permet triar entre un subconjunt aleatori de m característiques ($m < d$).

Selecció de l'Hiperparàmetre m

La mida del subconjunt m (`max_features` en scikit-learn) controla la diversitat:

- Si $m = d$: Tenim Bagging estàndard. Arbres correlacionats.
- Si $m = 1$: L'arbre tria particions totalment a l'atzar. Arbres molt diversos però molt dolents (biaix alt).

L'heurística d'enginyeria estàndard és:

$$m \approx \sqrt{d} \quad \text{(Classificació)}, \quad m \approx \frac{d}{3} \quad \text{(Regressió)} \tag{6.7}$$

Explicació per a Dummies: El Consell d'Administració

Imagineu que sou el CEO d'una empresa i voleu prendre una decisió.

- **Arbre Únic:** Li pregunteu a la millor persona experta. És llesta però arrogant (biaix) i canvia d'opinió si li canvies les dades (variància).
- **Bagging:** Reuniu 100 persones expertes. Però tots han anat a la mateixa universitat (Harvard) i han llegit els mateixos llibres. Tots opinaran igual. La reunió és inútil.
- **Random Forest:** Reuniu 100 persones expertes, però a cadascun li tapeu els ulls parcialment. A un només li deixeu veure els comptes financers. A l'altre només el màrqueting. A l'altre només els recursos humans. En forçar-los a decidir amb informació parcial, cada persona experta desenvolupa una perspectiva única. Quan ajunteu les 100 opinions parcials, obteniu una visió holística indestructible.

Per entendre-ho bé: OOB com a estimació interna del rendiment

En cada arbre t del Random Forest, algunes mostres no han entrat en el bootstrap. Aquestes mostres *out-of-bag* poden servir per estimar com respon aquell arbre a dades que no ha vist:

1. Per a cada mostra $\mathbf{x}_i$ del dataset original, identifiquem quins arbres **no** l'han vist durant l'entrenament.
2. Fem una predicció per a $\mathbf{x}_i$ agregant només els vots d'aquests arbres.
3. Calculem l'error (Accuracy o MSE) comparant aquesta predicció agregada amb l'etiqueta real y_i.

Es demostra que l'error OOB és un estimador imparcial de l'error de generalització, equivalent a fer Cross-Validation.

6.2.1 Importància de les Variables (Feature Importance)

A diferència de les xarxes neuronals ("caixes negres"), Random Forest ofereix interpretabilitat. Podem quantificar quines variables són les més importants per a la predicció. Existeixen diferents maneres de calcular-ho:

Mean Decrease Impurity (MDI) És el mètode per defecte en scikit-learn. Quan un arbre fa una partició usant la variable X_j, la impuresa (Gini o Entropia) disminueix. La importància de X_j és la suma ponderada de totes les disminucions d'impuresa aconseguides per X_j en tots els nodes de tots els arbres del bosc.

- **Avantatge:** Molt ràpid de calcular (es fa durant l'entrenament).

- **Inconvenient:** Té un biaix cap a variables amb moltes categories (ex: DNI o ID) o variables contínues, ja que permeten fer més talls.

Permutation Importance (Mètode més robust) Després d'entrenar el model, agafem el dataset de validació i **barregem aleatòriament** (permutem) els valors de la columna X_j, trencant qualsevol relació amb y. Mesurem quant empitjora la precisió del model.

$$Imp(X_j) = \text{Score}_{original} - \text{Score}_{permutat} \tag{6.8}$$

Si el model cau en picat, la variable era crítica. Si el model no s'immuta, la variable era soroll. Aquest mètode és computacionalment més car, però sovint és més informatiu que la importància basada en impuresa, especialment quan hi ha variables contínues o d'alta cardinalitat.

Extremely Randomized Trees (ExtraTrees) Existeix una variant encara més aleatòria: a més de triar un subconjunt de variables, els llindars de tall es trien aleatòriament en lloc de buscar el millor llindar local. Això pot accelerar l'entrenament i reduir la variància, però també pot augmentar el biaix. En alguns datasets funciona millor que Random Forest; en altres, no. Com sempre, cal validar-ho.

Consells de programació: Importància no és causalitat

La importància d'una variable en un Random Forest indica fins a quin punt el model l'ha utilitzada per millorar les seves prediccions. En el cas de la importància per permutació, es mesura quant empitjora el rendiment quan trenquem la relació entre aquesta variable i la sortida. Ara bé, això no vol dir que la variable sigui la causa del fenomen. Si dues variables estan molt correlacionades, la importància es pot repartir entre elles, quedar amagada o concentrar-se artificialment en una de les dues. Per tant, interpreta sempre la importància com una propietat diagnòstica del model, no com una prova causal.

Random Forest vs. Gradient Boosting (XGBoost)

Avui dia, **XGBoost / LightGBM** (Gradient Boosting) solen guanyar en precisió pura.
Però: Random Forest és molt més fàcil de posar en marxa.

- Random Forest és **paral·lelitzable** (pots entrenar X arbres a la vegada en X nuclis). XGBoost és seqüencial (l'arbre 2 corregeix l'arbre 1).
- Random Forest gairebé no necessita ajust d'hiperparàmetres. Amb els valors per defecte funciona molt bé. XGBoost requereix ajust fi (tuning).

Consell: Comenceu sempre amb un Random Forest. És el vostre "Baseline" robust. Només passeu a Boosting si necessiteu esprémer un 0.01% extra de precisió i teniu temps de computació.

Micro-Cas: Scoring de Crèdit Bancari

A la vida real, els Random Forests són els "tot terreny" (workhorses) de la ciència de dades. Abans de l'arribada del Deep Learning, guanyaven quasi totes les competicions de Kaggle sobre dades tabulars.
Per exemple, considerem un banc que té dades heterogènies dels seus clients:

- **Numèriques:** Sou, Deute total, Edat.

Scoring de Crèdit Bancari *(continuació)*

- **Categòriques:** Tipus de feina, Estat civil.
- **Ordinals:** Nivell d'estudis.

Per què RF i no SVM o Xarxes Neuronals?

1. **Robustesa a l'escalat:** No cal normalitzar les dades (els arbres només fan talls $X_j > \theta$). Un sou de 50.000 i una edat de 30 no es barregen matemàticament com en una SVM.
2. **Robustesa a outliers:** Un valor de sou erroni (1.000.000.000) només afecta a un tall, no distorsiona tot l'espai mètric.
3. **No linealitat:** Capta interaccions complexes (si és jove I té sou baix, risc alt; si és gran I té sou baix, risc mitjà).

El banc utilitza RF per aprovar hipoteques automàticament amb un nivell de confiança molt alt i sense necessitat de preprocessament complex.

6.2.2 Implementació i Visualització: El Cas del Titanic

Aplicarem Random Forest per predir la supervivència al Titanic i analitzarem quines variables utilitza el model com a senyals predictius. L'objectiu no és establir causalitat, sinó entendre el comportament del model entrenat.

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_openml
import numpy as np
import matplotlib.pyplot as plt

# 1. Carregar dades reals (Titanic)
titanic = fetch_openml('titanic', version=1, as_frame=True)
df = titanic.frame

features = ['pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'embarked']
X = df[features].copy()
y = df['survived'].astype(int)

# 2. Separació abans de qualsevol ajust de preprocessament
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)

# 3. Pipeline de preprocessament
numeric_features = ['pclass', 'age', 'sibsp', 'parch', 'fare']
categorical_features = ['sex', 'embarked']

preprocess = ColumnTransformer([
    ('num', SimpleImputer(strategy='median'), numeric_features),
    ('cat', Pipeline([
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('encoder', OneHotEncoder(handle_unknown='ignore'))
    ]), categorical_features)
])

# 4. Model complet
```

```python
rf = Pipeline([
    ('preprocess', preprocess),
    ('model', RandomForestClassifier(
        n_estimators=100,
        random_state=42,
        oob_score=True,
        bootstrap=True
    ))
])

rf.fit(X_train, y_train)

print(f"Accuracy Test: {rf.score(X_test, y_test):.3f}")
print(f"OOB Score: {rf.named_steps['model'].oob_score_:.3f}")

# 3. Visualitzar Importància de Variables (MDI)
importances = rf.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf.estimators_], axis=0)
indices = np.argsort(importances)

feature_names = np.array(features)

plt.figure(figsize=(10, 6))
plt.title("Variables més utilitzades pel model en el cas Titanic (Random Forest Importance)",
     fontsize=14, fontweight='bold')
plt.barh(range(X.shape[1]), importances[indices], color="#1f77b4", xerr=std[indices], align="
    center")
plt.yticks(range(X.shape[1]), feature_names[indices], fontsize=12)
plt.xlabel("Importància relativa (Mean Decrease Impurity)")
plt.grid(axis='x', linestyle='--', alpha=0.5)

# Afegir text explicatiu
plt.text(0.15, 0.5, "Les barres negres indiquen\nla variabilitat entre arbres", fontsize=10,
    color='gray')

plt.tight_layout()
plt.savefig('fig_rf_importance.png', dpi=300)
plt.show()
```

Listing 6.2: Random Forest i Feature Importance al Titanic

Per entendre-ho bé: El model no explica la història completa

En el cas Titanic, el Random Forest mostra quines variables són útils per predir supervivència dins d'aquest dataset. Això no és el mateix que explicar causalment què va determinar la supervivència històrica. El model detecta patrons predictius; la interpretació causal requereix més context.

6.3 Xarxes Bayesianes: Dependència i Causalitat

Fins ara, els nostres models han estat màquines de correlació. Un Random Forest pot predir que "si tens els dits grocs, tens càncer de pulmó". Estadísticament funciona (correlació), però biològicament és fals (causalitat). Les **Xarxes Bayesianes (BN)** neixen per modelar explícitament la dependència i la causalitat entre variables.

Una xarxa bayesiana és, en primer lloc, una representació probabilística de dependències condicionals. Són Grafs Acíclics Dirigits (DAGs) on els nodes són variables aleatòries i les arestes representen influència probabilística directa. El seu graf indica com factoritzem una

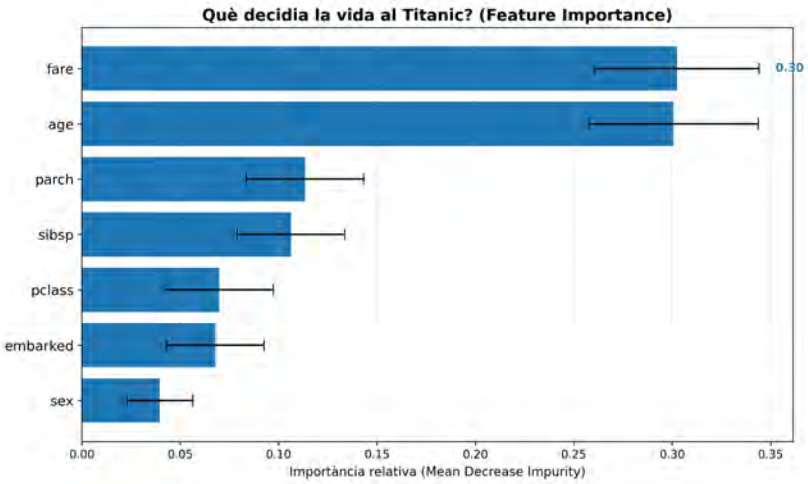


Figura 6.2: Importància de variables estimada pel Random Forest en el dataset Titanic. La figura mostra quines variables són més utilitzades pel model per predir supervivència; no s'ha d'interpretar com una anàlisi causal directa dels factors històrics de supervivència.

distribució conjunta i quines independències condicionals assumim. Només podem interpretar les arestes com a relacions causals si el graf ha estat construït amb aquest propòsit i disposem de supòsits causals addicionals.

Per entendre-ho bé: Una fletxa no sempre és una causa

En una xarxa bayesiana, una fletxa indica una dependència dins del model. Pot tenir interpretació causal si hem construït el graf amb aquest objectiu i acceptem els supòsits causals necessaris. Però una xarxa bayesiana apresa o dibuixada sense aquests supòsits no demostra causalitat per si sola.

6.3.1 La Factorització Eficient

Imaginem un sistema amb n variables binàries $X_1, \ldots, X_n$. Si volguéssim modelar la seva **Distribució de Probabilitat Conjunta (JPD)**, $P(X_1, \ldots, X_n)$, necessitaríem una taula amb $2^n - 1$ paràmetres. Per a $n = 30$ sensors en una fàbrica, $2^{30} \approx 10^9$ (mil milions) de paràmetres, impossible d'aprendre ni emmagatzemar. Les Xarxes Bayesianes resolen això aprofitant la regla de la cadena (Chain Rule):

$$P(X_1, \ldots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots \tag{6.9}$$

Però, en una Xarxa Bayesiana, assumim que una variable X_i només depèn dels seus pares directes en el graf, $Pa(X_i)$. Això permet la **Factorització Local**:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{6.10}$$

Si cada variable té com a màxim k pares, el nombre de paràmetres passa de 2^n (exponencial) a $n \cdot 2^k$ (lineal respecte a n). Aquesta reducció de complexitat és la clau que permet modelar sistemes massius.

La **d-separació** és una propietat del graf: indica quines independències condicionals són compatibles amb l'estructura de la xarxa. Si interpretem el graf causalment, també pot orientar

raonament causal, però aquesta interpretació requereix supòsits addicionals. El graf per si sol no demostra causalitat a partir de les dades.

Per entendre-ho bé: d-separació és bloquejar informació estadística

La d-separació no diu que una variable empenyi físicament una altra. Diu que, un cop coneixem certes variables, observar-ne una altra ja no ens aporta informació addicional sobre la primera. És una regla gràfica per llegir independències condicionals.

El concepte més difícil i potent de les Xarxes Bayesianes és entendre com flueix la informació. Quan el coneixement d'una variable A ens diu alguna cosa sobre B? Això depèn de l'estructura del graf i del que hem observat.

De fet, hi ha tres estructures que governen el flux d'informació (Judea Pearl, 1988):

1. **Cadena (Causalitat Directa)**: $A \to B \to C$
   - *Sense observar res:* A i C són dependents. Si A (Foc) passa, B (Fum) passa, i C (Alarma) sona.
   - *Observant B:* Si sabem segur que hi ha Fum (B), saber si hi ha Foc (A) ja no ens aporta res nou sobre l'Alarma (C). Diem que B bloqueja el camí. $A \perp C|B$.
2. **Causa Comuna (Fork)**: $A \leftarrow B \to C$
   - Exemple: B (Edat) causa A (Mida sabata) i C (Vocabulari).
   - *Sense observar res:* A i C estan correlacionats (nens amb peus grans saben més paraules).
   - *Observant B:* Si fixem l'edat (ex: només mirem nens de 10 anys), la mida del peu ja no ens diu res sobre el vocabulari. B bloqueja el camí (d-separació).
3. **Estructura en V (Collider) - La Trampa**: $A \to B \leftarrow C$
   - Exemple: A (Talent) i C (Treball dur) causen B (Èxit).
   - *Sense observar res:* A i C són independents: talent i esforç no estan relacionats.
   - *Observant B:* Aquí canvia tot. Si sabem que algú té èxit (B) i sabem que **no** té talent (A), aleshores **necessàriament** ha d'haver treballat molt (C).
   - **Resultat:** Observar l'efecte comú **crea** una dependència entre les causes. El camí s'obre quan observem B. Això es coneix com a *Explaining Away*.

Aprenentatge d'Estructura (Structure Learning): De fet, el gran repte no és fer inferència, sinó aprendre les fletxes del graf a partir de dades. És un problema NP-Hard. Per tant, no intenteu aprendre tota l'estructura des de zero (algorisme PC o Hill Climbing): utilitzeu **coneixement expert** per fixar les fletxes òbvies (ex: La pluja causa fang, no a l'inrevés) i deixeu que l'algorisme aprengui només les relacions subtils desconegudes.

La V-Structure i les Cites

Imagineu que observeu gent que té parella (heu observat l'efecte B: "Tenir Parella"). Assumiu que per tenir parella cal ser **Guapo** (A) o **Simpàtic** (C).

- Si coneixeu algú que té parella i veieu que és molt **lleig** (No-A), automàticament deduïu que ha de ser molt **simpàtic** (C).
- Tot i que la bellesa i la simpatia són independents en la població general, dins del grup "Gent amb Parella", estan correlacionades negativament.

La V-Structure i les Cites *(continuació)*

Aquest fenomen (Biaix de Berkson) és una conseqüència matemàtica directa de l'estructura en V de les Xarxes Bayesianes.

6.3.2 Del Sentit Comú a l'Algorisme

Les fórmules anteriors descriuen exactament com raona el cervell humà. Mirem de descomprendre el procés de pensament causal.

1. El concepte: El Cas de l'Alarma de Robatori: Imagina que tens una alarma antirobatori a casa teva. És un sistema imperfecte.

- **Causa 1 (Robatori):** Si entra un lladre, l'alarma sona (gairebé sempre).
- **Causa 2 (Terratrèmol):** Si hi ha un terratrèmol, l'alarma també salta.

Estàs a la feina i el teu veí et truca: "L'alarma està sonant!". Què penses?

1. Immediatament, la creença que hi ha un **Robatori** puja al 99%. T'espantes.
2. Poses la ràdio i sents: "S'acaba de detectar un terratrèmol lleu a la teva zona".
3. Què passa amb la teva por al robatori? **Baixa dràsticament**.

Per què? El lladre i el terratrèmol són esdeveniments independents (els lladres no provoquen terratrèmols). Però un cop saps que l'alarma sona (l'efecte comú), saber que hi ha hagut un terratrèmol "explica" l'alarma, fent innecessària la hipòtesi del robatori. La teva ment acaba de fer un càlcul d'**Explaining Away** (Descartar per explicació alternativa). Una Xarxa Bayesiana automatitza aquest raonament.

2. La Propagació de la creença: en enginyeria, no parlem de "por" o "sorpresa", sinó d'actualització de probabilitats. Una Xarxa Bayesiana és un motor d'inferència que propaga evidència a través del graf.

Considerem el graf: **Núvols** → **Pluja** → **Gespa Mullada**.

1. **Inferència Causal (Top-Down):** Si veig **Núvols**, augmenta la probabilitat de **Pluja**, i per tant augmenta la probabilitat de **Gespa Mullada**. L'evidència flueix en la direcció de les fletxes. És una predicció.
2. **Inferència Diagnòstica (Bottom-Up):** Si veig la **Gespa Mullada** (efecte), augmenta la probabilitat que hagi plogut (causa directa), i això alhora augmenta la probabilitat que hi hagués núvols (causa remota). L'evidència flueix contra les fletxes. És un diagnòstic (com una persona professional de l'àmbit clínic veient un símptoma).
3. **Bloqueig d'Informació:** Si **JA SÉ** que plou (observo la pluja directament), aleshores mirar els núvols és irrellevant per saber si la gespa està mullada. La variable intermèdia "Pluja", un cop observada, es converteix en un mur que aïlla els núvols de la gespa. Això és la **d-separació**.

3. La Factorització de la JPD: Una Xarxa Bayesiana és una afirmació contundent sobre l'estructura de l'espai de probabilitat. És a dir, si volem modelar $P(X_1, \ldots, X_n)$ sense estructura, necessitem 2^n paràmetres. Per a $n = 100$ variables mèdiques, aquesta taula tindria una mida astronòmica. Però la BN ens permet escriure la Probabilitat Conjunta (JPD) com un producte de factors locals de baixa dimensió.

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | Pa(X_i))$$

Aquesta equació té dues implicacions profundes:

1. **Estalvi Exponencial:** Si cada node té com a màxim k pares, la complexitat de representació baixa de $\mathcal{O}(2^n)$ a $\mathcal{O}(n \cdot 2^k)$. Això fa tractable allò intractable.
2. **Semàntica de la Independència:** L'absència d'una aresta és més informativa que la seva presència. Si no hi ha fletxa entre X i Y, estem afirmant una independència condicional forta:

$$I(X, Y|Z) \iff P(X, Y|Z) = P(X|Z)P(Y|Z)$$

Això ens permet utilitzar algorismes de pas de missatges (*Belief Propagation*) per fer inferència exacta en temps polinòmic (en grafs sense bucles).

Pearl i el "do-calculus"

No confoneu **Observació** amb **Intervenció**.

- **Observació:** $P(Y|X)$. "Quina és la probabilitat que la vorera estigui mullada si veig gent amb paraigües?" (Alta, plou).
- **Intervenció:** $P(Y|\text{do}(X))$. "Quina és la probabilitat que la vorera estigui mullada si jo força a tothom a portar paraigües (regalo paraigües al carrer)?" (Baixa, no faig ploure).

Les Xarxes Bayesianes Causals permeten calcular l'efecte de les intervencions ($\text{do}(X)$) modificant el graf (eliminant les fletxes que entren a X). Així, no volem saber si prendre la pastilla està *correlacionat* amb curar-se, volem saber si *causa* la cura.

Visualitzem a continuació com es propaga la probabilitat. Escenari:

- Hi ha Núvols (C) $\rightarrow$ Pluja (R).
- Hi ha Núvols (C) $\rightarrow$ Aspersor encès (S) (Si està núvol, no reguem).
- Gespa Mullada (W) depèn de Pluja (R) i Aspersor (S).

```python
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

# 1. Definició de l'Estructura del Graf (DAG)
edges = [('Núvols', 'Pluja'), ('Núvols', 'Aspersor'),
         ('Pluja', 'Gespa Mullada'), ('Aspersor', 'Gespa Mullada')]
G = nx.DiGraph()
G.add_edges_from(edges)

pos = {          # 2. Definició de posicions per visualització jeràrquica
    'Núvols': (0, 2),
    'Pluja': (-1, 1),
    'Aspersor': (1, 1),
    'Gespa Mullada': (0, 0)
}

plt.figure(figsize=(8, 6))      # 3. Visualització
nx.draw_networkx_nodes(G, pos, node_size=3000, node_color='lightblue', edgecolors='black')
nx.draw_networkx_edges(G, pos, width=2, arrowsize=20, arrowstyle='->')
nx.draw_networkx_labels(G, pos, font_size=12, font_weight='bold')

# 4. Afegir Taules de Probabilitat (CPTs) conceptuals com text
plt.text(-2.5, 2, "P(N) = 0.5", fontsize=10, bbox=dict(facecolor='white', alpha=0.8))
```

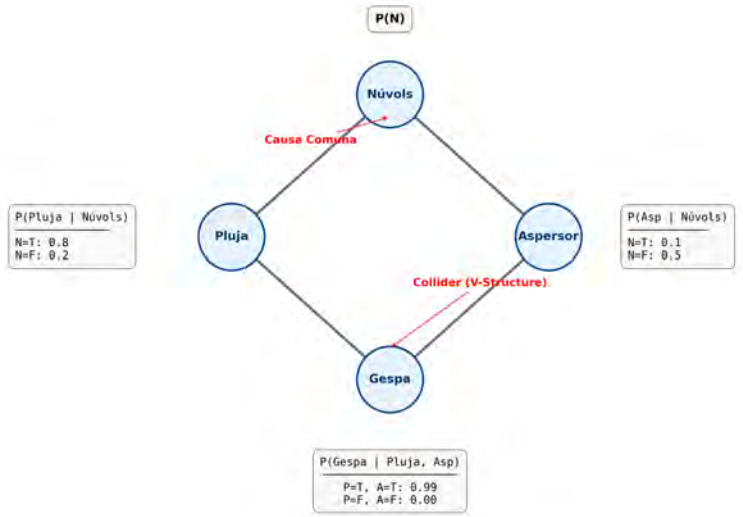


Figura 6.3: Representació gràfica d'una Xarxa Bayesiana. L'estructura "Núvols" és una **Causa Comuna** per a Pluja i Aspersor. "Gespa Mullada" és un **Collider** (estructura en V) entre Pluja i Aspersor. L'estructura defineix com flueix la informació: si sabem que ha plogut, la probabilitat que l'aspersor estigui encès baixa (ja que la pluja "explica" la gespa mullada).

```
25 plt.text(-2.8, 1, "P(P|N)\nSi N=T: 0.8\nSi N=F: 0.2", fontsize=10, bbox=dict(facecolor='white
       ', alpha=0.8))
26 plt.text(1.5, 1, "P(A|N)\nSi N=T: 0.1\nSi N=F: 0.5", fontsize=10, bbox=dict(facecolor='white',
        alpha=0.8))
27 plt.text(-2.0, 0, "P(G|P,A)\nP=T, A=T: 0.99\nP=F, A=F: 0.00", fontsize=10, bbox=dict(
       facecolor='white', alpha=0.8))
28
29 plt.title("Xarxa Bayesiana: Causa Comuna i Estructura en V", fontsize=14, fontweight='bold')
30 plt.axis('off')
31 plt.tight_layout()
32 plt.savefig('fig_bayes_net.png', dpi=300)
33
34 # Simulació d'Inferència (Càlcul manual per imprimir)
35 # Cas: Observem Gespa Mullada = True. Quina és la prob d'Aspersor?
36 # Això requeriria Variable Elimination, aquí només mostrem el concepte.
37 print("Graf generat: inferència propaga evidència des de 'Gespa' cap amunt.")
38 plt.show()
```

Listing 6.3: Visualització i Inferència en una Xarxa Bayesiana

Micro-Cas: Diagnòstic Industrial en una Central Elèctrica: El Misteri de la Turbina

Les Xarxes Bayesianes brillen on Random Forest falla: quan cal raonar amb incertesa i informació incompleta en sistemes complexos. Imaginem que en una central de cicle combinat, tenim sensors i alarmes.

- **Variables:** T (Temp. Alta), P (Pressió Alta), F (Error Flux), $A1$ (Alarma T), $A2$ (Alarma P).
- **Estructura:** La temperatura alta (T) causa que salti l'alarma $A1$. La pressió alta (P) causa l'alarma $A2$. Però una fallada de refrigeració (R) causa tant T com P.

Diagnòstic Industrial en una Central Elèctrica: El Misteri de la Turbina *(continuació)*

Escenari: Salta l'alarma $A1$.

1. La probabilitat de T puja.
2. Com que T puja, la probabilitat de la causa arrel R (Refrigeració) puja.
3. Com que la probabilitat de R ha pujat, el sistema prediu que la pressió P també deu ser alta, i per tant, l'alarma $A2$ hauria d'estar sonant.
4. Si verifiquem i $A2$ està apagada, la probabilitat de R baixa dràsticament, i el sistema dedueix que T és un error de sensor local o una altra causa.

Aquest raonament "de dalt a baix i de baix a dalt" és impossible amb models predictius estàndard.

6.4 Xarxes de Markov Ocultes: la Variable Temps

Hem vist que una xarxa bayesiana factoritza una distribució conjunta mitjançant un graf de dependències. Ara afegirem una estructura temporal molt específica: una cadena d'estats ocults que evoluciona en el temps i genera observacions.

Una Xarxa de Markov Oculta (*Hidden Markov Model*, HMM) és, essencialment, una Xarxa Bayesiana dinàmica que modela una seqüència de dades al llarg del temps t. Així, fins ara hem tractat les dades com a punts aïllats en l'espai ($\mathbf{x}_i \in \mathbb{R}^d$), però en molts problemes d'enginyeria, l'ordre és la informació (veu, text, borsa, robòtica, ...). Entrem en el domini dels **Processos Estocàstics**.

6.4.1 Fonaments Matemàtics: La Propietat de Markov

Un HMM es defineix per dos processos paral·lels que avancen en el temps:

1. **El Procés Ocult (Latent):** Una seqüència d'estats $z_1, z_2, \ldots, z_T$ que no podem observar directament, e.g. l'estat mental d'una persona usuària ("Contenta").
2. **El Procés Observat (Emissió):** Una seqüència de dades $x_1, x_2, \ldots, x_T$ que sí mesurem. Per exemple, el to de veu o les paraules que diu.

Per fer el càlcul tractable, Andrei Markov va proposar l'any 1913 una simplificació radical coneguda com la **Propietat de Markov d'Ordre 1**: *"El futur és independent del passat donat el present."*. Matemàticament:

$$P(z_{t+1}|z_t, z_{t-1}, \ldots, z_1) = P(z_{t+1}|z_t) \tag{6.11}$$

Això significa que per predir on serà un robot a $t+1$, només necessito saber on és a t. No m'importa com ha arribat fins allà. Com a resultat, un HMM queda completament definit per tres matrius de probabilitat, la terna (π, A, B):

1. Probabilitat Inicial (π): Un vector que ens diu on comença el sistema a $t = 1$.

$$\pi_k = P(z_1 = k)$$

2. Matriu de Transició (A): Descriu la dinàmica del sistema ocult. És una matriu quadrada $K \times K$ (on K és el nombre d'estats possibles).

$$A_{ij} = P(z_{t+1} = j|z_t = i)$$

Exemple: Si estic "Content" (i), quina és la probabilitat de passar a estar "Enfadat" (j) al següent instant?

3. Matriu d'Emissió (B): Connecta l'estat ocult amb la realitat observable.

$$B_{ik} = P(x_t = k | z_t = i)$$

Exemple: Si estic "Content" (i), quina és la probabilitat que digui la paraula "Fantàstic" (k)? Si les observacions són contínues (ex: senyal de ràdio), B no és una matriu sinó una distribució de probabilitat (sovint una Gaussiana). Així, un HMM combina dues hipòtesis d'independència:

1. **Hipòtesi de Markov:** l'estat ocult actual només depèn de l'estat ocult anterior: $P(z_t \mid z_{1:t-1}) = P(z_t \mid z_{t-1})$.
2. **Hipòtesi d'emissió:** l'observació actual només depèn de l'estat ocult actual: $P(x_t \mid z_{1:t}, x_{1:t-1}) = P(x_t \mid z_t)$.

Aquestes hipòtesis fan el model tractable, però també limiten quines dependències temporals pot representar.

Per entendre-ho bé: Què amaga exactament un HMM?

Un HMM separa dues coses: l'estat real del sistema, que no veiem, i l'observació, que sí veiem. Per exemple, no veiem directament si una persona està sana o malalta, però veiem símptomes. El model assumeix dues simplificacions: l'estat d'avui depèn sobretot de l'estat d'ahir, i l'observació d'avui depèn de l'estat ocult d'avui.

El Diari d'un Presoner

Imagina un presoner tancat en una cel·la sense finestres. Vol saber quin temps fa a fora (Sol o Pluja). Aquest és l'estat **Ocult** (z_t). L'única informació que té és veure si el guàrdia porta botes brutes de fang o sabates netes quan li porta el menjar. Aquesta és l'**Observació** (x_t).

- **Transició (A):** Si avui fa sol, demà és molt probable que faci sol (el temps té inèrcia).
- **Emissió (B):** Si fa pluja (z_t), és molt probable que el guàrdia porti botes (x_t), però no segur (potser les ha netejat).

L'HMM és l'eina matemàtica que permet al presoner deduir la seqüència de dies de pluja (el clima) mirant només una seqüència de sabates.

Underflow Numèric

Quan treballem amb seqüències llargues ($T = 1000$), calcular la probabilitat conjunta implica multiplicar 1000 probabilitats petites.

$$P = 0.1 \times 0.2 \times \cdots \approx 10^{-1000}$$

Això causa un **Arithmetic Underflow**: l'ordinador ho arrodoneix a 0 i el programa falla. **Solució:** HMM **sempre** treballa en l'espai Logarítmic.

$$\log P = \log(0.1) + \log(0.2) + \ldots$$

Les multiplicacions es converteixen en sumes, els números es mantenen en un rang operable (ex: -2300), i la precisió es conserva. Mai implementeu un HMM sense logaritmes!

6.4.2 Visualització del Graf Temporal

A diferència de les Xarxes Bayesianes estàtiques, un HMM es desplega en el temps. A continuació, generem un diagrama que mostra aquesta estructura dinàmica.

```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx

# 1. Definició del Model (Exemple: El Casino Deshonest)
# Estats Ocults: Dau Just (0), Dau Trucat (1)
states = ["Just", "Trucat"]
# Observacions: 1, 2, 3, 4, 5, 6
observations = ["1", "2", "3", "4", "5", "6"]

# Matriu de Transició (A)
# Costa sortir de l'estat trucat (0.9 de probabilitat de quedar-s'hi)
A = np.array([[0.95, 0.05],
              [0.10, 0.90]])

# Matriu d'Emissió (B)
# Dau Just: prob uniforme 1/6. Dau Trucat: 50% de treure un 6.
B = np.array([[1/6, 1/6, 1/6, 1/6, 1/6, 1/6],
              [0.1, 0.1, 0.1, 0.1, 0.1, 0.5]])

# 2. Generació de Dades Sintètiques
def generate_hmm(T):
    z = np.zeros(T, dtype=int)
    x = np.zeros(T, dtype=int)

    # Estat inicial (Assumim que comença Just)
    z[0] = 0
    x[0] = np.random.choice(6, p=B[z[0]])

    for t in range(1, T):
        # Transició
        z[t] = np.random.choice(2, p=A[z[t-1]])
        # Emissió
        x[t] = np.random.choice(6, p=B[z[t]])

    return z, x

# Generar 50 passos de temps
T = 50
z_hist, x_hist = generate_hmm(T)

# 3. Visualització
plt.figure(figsize=(14, 4))
# Dibuixem els estats ocults com a fons de color
plt.imshow(z_hist.reshape(1, -1), aspect='auto', cmap='Pastel1', extent=[0, T, 0, 7], alpha
    =0.5)
# Dibuixem les observacions (daus)
plt.scatter(range(T), x_hist + 1, c='black', s=50, label='Observació (Dau)')
plt.step(range(T), (z_hist * 5) + 1, 'r--', alpha=0.5, where='mid', label='Estat Ocult Real
    (0=Just, 1=Trucat)')

plt.title("Simulació HMM: El Casino Deshonest", fontsize=14, fontweight='bold')
plt.xlabel("Temps (t)")
plt.ylabel("Resultat del Dau")
plt.yticks(range(1, 7))
```

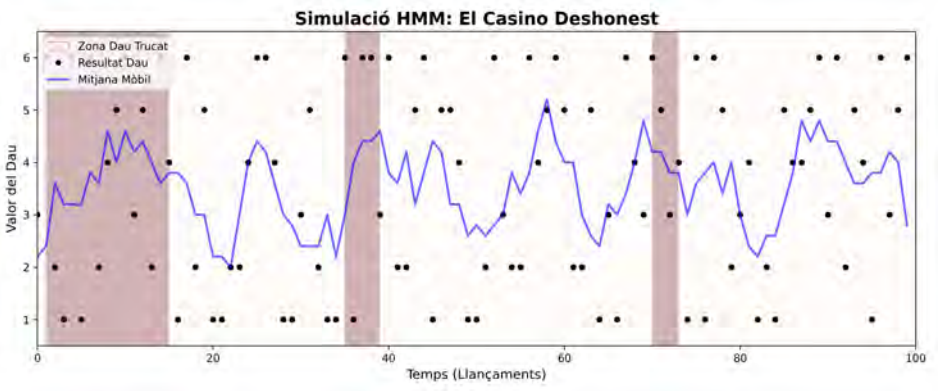


Figura 6.4: Visualització d'una seqüència generada per un HMM. Els punts negres són les dades que veiem (resultats del dau). Les zones de color de fons representen la "veritat oculta" (si el dau usat en aquell moment era just o trucat). Observeu com, en la zona del dau trucat, la freqüència de sisos augmenta dràsticament. L'objectiu de l'HMM és recuperar les zones de color mirant només els punts negres.

```
54 plt.grid(True, axis='x', linestyle='--', alpha=0.3)
55 plt.legend(loc='upper right')
56
57 # Afegir text explicatiu
58 plt.text(1, 6.5, "Zona Verda: Dau Trucat (més sisos)", fontsize=10, color='green', fontweight
       ='bold')
59 plt.text(1, 0.5, "Zona Vermella: Dau Just", fontsize=10, color='red', fontweight='bold')
60
61 plt.tight_layout()
62 plt.savefig('fig_hmm_simulation.png', dpi=300)
63 plt.show()
```

Listing 6.4: Generació i Visualització d'una seqüència HMM

Micro-Cas: Reconeixement de gestos: Controlant la TV amb la Mà

L'HMM ha estat l'estat de l'art en reconeixement de gestos (i veu) durant dècades, abans de les xarxes recurrents (RNN).
Volem que una Smart TV detecti quan la persona usuària fa el gest de "Pujar Volum" (moure la mà de baix a dalt).

- **Input:** Seqüència de vectors (x, y, z) d'un acceleròmetre en un comandament o smartwatch.
- **Model HMM:** Definim un HMM amb 3 estats ocults seqüencials:

$$\text{Inici Repòs} \rightarrow \text{Moviment Ascendent} \rightarrow \text{Fi Repòs}$$

- **Entrenament:** Gravem 100 exemples del gest. L'algorisme (Baum-Welch, equivalent a l'EM) aprèn que a l'estat "Moviment Ascendent", la component y de l'acceleració és positiva i alta.
- **Inferència:** Quan arriba una nova seqüència, calculem la probabilitat que hagi estat generada per aquest model. Si és alta, activem el volum.

La gràcia de l'HMM és que és elàstic temporalment: detecta el gest tant si el fas ràpid (0.5s) com lent (2s), ja que permet quedar-se en l'estat "Moviment Ascendent" tants passos de temps com calgui (gràcies a l'auto-transició A_{ii}).

6.5 Inferència en HMM: L'Algorisme Forward i Viterbi

Tenir un model HMM definit per $\lambda = (\pi, A, B)$ (com vam veure a la secció anterior) no serveix de res si no podem fer-li preguntes. Hi ha tres preguntes fonamentals que volem respondre:

1. **Avaluació (Likelihood):** Donada una seqüència d'observacions $X = x_1, \ldots, x_T$, quina és la probabilitat $P(X|\lambda)$ que el meu model l'hagi generat? (Útil per classificar seqüències).
2. **Decodificació (Decoding):** Donada X, quina és la seqüència d'estats ocults $Z = z_1, \ldots, z_T$ més probable? (Algorisme de Viterbi).
3. **Aprenentatge (Training):** Donada només X, com trobem els paràmetres π, A, B? (Algorisme de Baum-Welch / EM).

Ara ens centrarem en els dos primers problemes que són la base de la inferència. Suposem que volem calcular $P(X|\lambda)$. La probabilitat de les dades depèn de la seqüència oculta Z. Com que no sabem quina és, hem de sumar sobre **totes les possibles** seqüències d'estats:

$$P(X|\lambda) = \sum_{\text{tots } Z} P(X|Z, \lambda)P(Z|\lambda) \tag{6.12}$$

Si tenim N estats possibles i una seqüència de longitud T, hi ha N^T camins possibles, per a $N = 3$ estats i $T = 100$ segons (una frase curta), tardaria $3^{100} \approx 5 \times 10^{47}$. Aquesta és la **maledicció combinatòria**. Necessitem una drecera.

6.5.1 L'Algorisme Forward: Programació Dinàmica

La solució és la **Programació Dinàmica**: recordar resultats parcials per no haver de recalcular-los. Definim la variable forward $\alpha_t(i)$ com: *"La probabilitat d'haver observat la seqüència parcial $x_1, \ldots, x_t$ i acabar a l'estat i al temps t."*

$$\alpha_t(i) = P(x_1, x_2, \ldots, x_t, z_t = i|\lambda) \tag{6.13}$$

Podem calcular $\alpha_t(i)$ de manera recursiva:

1. Inicialització ($t = 1$): La probabilitat de començar a l'estat i i emetre x_1:

$$\alpha_1(i) = \pi_i B_i(x_1) \tag{6.14}$$

2. Inducció ($1 < t \leq T$): Per arribar a l'estat j al temps $t + 1$, podem venir de qualsevol estat i al temps t. Sumem totes les rutes possibles, ponderades per la transició A_{ij} i l'emissió $B_j(x_{t+1})$:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)A_{ij}\right] B_j(x_{t+1}) \tag{6.15}$$

Aquesta fórmula és màgica. Transforma el sumatori global en un càlcul local. La complexitat passa de $O(N^T)$ a $O(N^2T)$, que és tractable (lineal amb el temps!).

3. Terminació: La probabilitat total de la seqüència és la suma de les probabilitats d'acabar en qualsevol estat al temps T:

$$P(X|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{6.16}$$

L'algorisme Forward multiplica moltes probabilitats petites. En seqüències llargues, això pot provocar *underflow*: els valors esdevenen tan petits que l'ordinador els arrodoneix a zero.

El Senderista Perdut

Imagina un excursionista que vol anar del punt A al punt B travessant un bosc espès.

- **Força Bruta**: L'excursionista intenta traçar **tots** les camins possibles des de l'inici fins al final d'una sola tirada. Hi ha bilions de combinacions d'arbres. Es perd i mor.
- **Algorisme Forward**: L'excursionista divideix el camí en etapes (temps t). A cada etapa, només calcula: "Quina probabilitat tinc d'arribar a aquest arbre concret?".
- Per anar a l'arbre del següent pas, no li importa tota la història prèvia, només la probabilitat acumulada fins al punt actual. "Si he arribat viu fins aquí, només he de calcular el següent pas".

Això converteix un problema impossible en una sèrie de problemes trivials.

6.5.2 L'Algorisme de Viterbi: Trobar el Camí Òptim

Sovint no volem només la probabilitat total, sinó saber **què ha passat realment**. Volem la seqüència d'estats Z^* més probable (el "camí crític").

$$Z^* = \arg\max_Z P(Z|X, \lambda)$$

L'algorisme de Viterbi és gairebé idèntic al Forward, però en lloc de sumar les probabilitats (marginalitzar), agafem el màxim (maximitzar).

Definim $\delta_t(i)$ com la probabilitat del camí més probable que acaba a l'estat i al temps t:

$$\delta_t(i) = \max_{z_1,\dots,z_{t-1}} P(z_1, \dots, z_{t-1}, z_t = i, x_1, \dots, x_t|\lambda) \tag{6.17}$$

La **recursió de Viterbi** calcula, per a cada instant $t+1$ i per a cada estat possible j, la probabilitat del millor camí que acaba en aquest estat:

$$\delta_{t+1}(j) = \max_i \left[\delta_t(i)A_{ij}\right] B_j(x_{t+1}). \tag{6.18}$$

Aquí, $\delta_t(i)$ representa la probabilitat del millor camí que arribava a l'estat i en el temps t, A_{ij} és la probabilitat de transitar de l'estat i a l'estat j, i $B_j(x_{t+1})$ és la probabilitat d'observar x_{t+1} si l'estat ocult en aquell instant és j.

Però calcular només el valor màxim no és suficient: també hem de recordar quin estat anterior ha produït aquest màxim. Per això guardem un punter:

$$\psi_{t+1}(j) = \arg\max_i \left[\delta_t(i)A_{ij}\right]. \tag{6.19}$$

Aquest punter, anomenat *backpointer*, indica quin estat anterior i era el millor predecessor de j. Al final de la seqüència, seleccionem l'estat amb valor $\delta_T(j)$ més alt i reconstruïm el camí òptim recorrent els punters ψ cap enrere. Així, Viterbi no tria l'estat més probable localment a cada instant, sinó la seqüència completa d'estats amb probabilitat global màxima.

Consells de programació: Implementa HMM en log-probabilitats

Forward i Viterbi multipliquen moltes probabilitats petites. En seqüències llargues, això pot provocar *underflow*, és a dir, que l'ordinador arrodoneixi el resultat a zero.
Per estabilitat numèrica, és habitual treballar en log-probabilitats:

$$\log(a \cdot b) = \log a + \log b.$$

En Viterbi, això converteix els productes en sumes i manté l'operació de màxim. En Forward, però, cal sumar probabilitats de camins alternatius.
Com que $\log(a+b) \neq \log a + \log b$, aquesta suma s'ha de fer amb el truc **Log-Sum-Exp**:

$$\log\left(\sum_i e^{x_i}\right) = x_{\max} + \log\left(\sum_i e^{x_i - x_{\max}}\right). \tag{6.20}$$

Això permet implementar HMM de manera estable sense tornar a probabilitats normals.

Entendre la Programació Dinàmica és visualitzar α o δ com un mapa de calor temporal.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Definició del Model (Exemple: Genètica simple)
# Estats: 0=Exó (codificant), 1=Intró (no codificant)
states = ["Exó", "Intró"]
# Observacions: 0=A, 1=C, 2=G, 3=T
obs_map = {'A':0, 'C':1, 'G':2, 'T':3}

pi = np.array([0.5, 0.5])           # Probabilitats (Inventades per l'exemple)
A = np.array([[0.9, 0.1],  # Difícil sortir d'un Exó
              [0.1, 0.9]]) # Difícil sortir d'un Intró
B = np.array([[0.1, 0.4, 0.4, 0.1], # Exó: ric en C/G
              [0.3, 0.2, 0.2, 0.3]]) # Intró: ric en A/T

seq_str = "CGGACGTA"                # Seqüència Observada (ADN)
obs_seq = [obs_map[c] for c in seq_str]
T = len(obs_seq)
N = len(states)

# 2. Algorisme Forward (Càlcul de la matriu Alpha)
alpha = np.zeros((T, N))

alpha[0] = pi * B[:, obs_seq[0]]    # Inicialització

for t in range(1, T):               # Inducció
    for j in range(N):
        # Suma ponderada de les probabilitats anteriors
        alpha[t, j] = np.sum(alpha[t-1] * A[:, j]) * B[j, obs_seq[t]]

plt.figure(figsize=(10, 5))         # 3. Visualització (Trellis Diagram)
# Normalitzem per fila per visualitzar millor la distribució de probabilitat
alpha_norm = alpha / alpha.sum(axis=1, keepdims=True)

sns.heatmap(alpha_norm.T, annot=True, fmt=".2f", cmap="Blues",
            xticklabels=list(seq_str), yticklabels=states)

plt.title("Diagrama de Trellis (Forward): Probabilitat d'Estat al llarg del Temps", fontsize
```

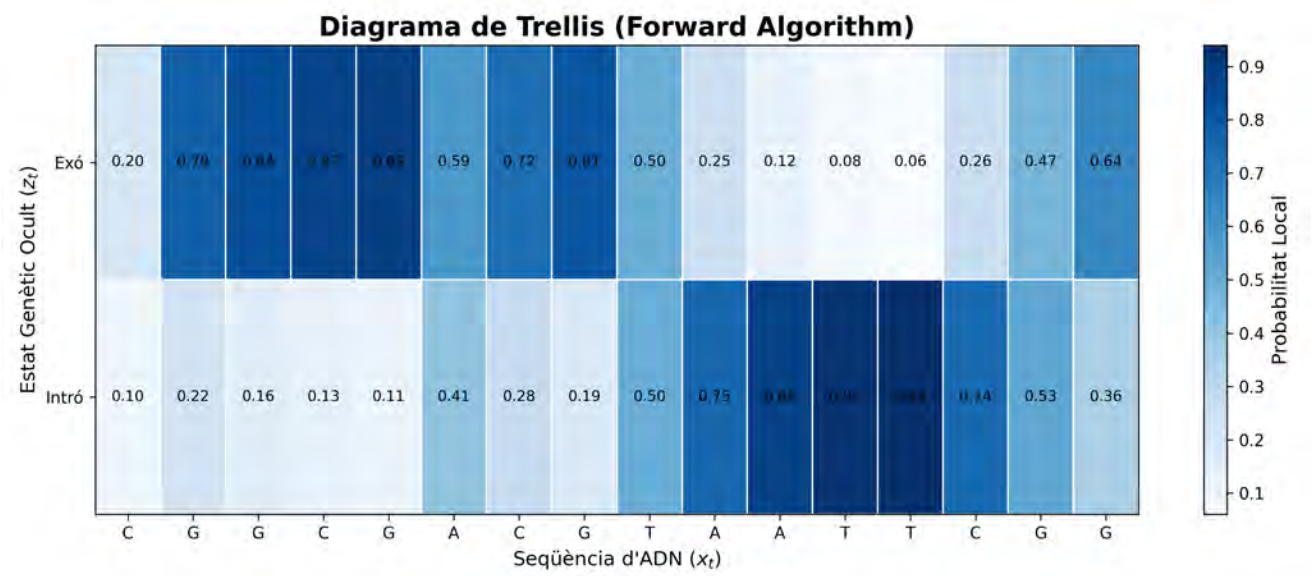


Figura 6.5: Mapa de calor de la matriu Forward (α_t). L'eix X representa el temps (la seqüència d'ADN) i l'eix Y els estats ocults. El color fosc indica alta probabilitat. Observeu com en les zones riques en C/G, el model assigna major probabilitat a l'estat "Exó", mentre que en A/T es decanta per "Intró". L'algorisme propaga aquesta incertesa pas a pas.

```
    =14, fontweight='bold')
plt.xlabel("Seqüència Observada ($x_t$)", fontsize=12)
plt.ylabel("Estat Ocult ($z_t$)", fontsize=12)
plt.tight_layout()
plt.savefig('fig_hmm_trellis.png', dpi=300)
plt.show()

print(f"Probabilitat total de la seqüència: {np.sum(alpha[-1]):.2e}")
```

Listing 6.5: Implementació i Visualització de l'Algorisme Forward (Trellis)

Micro-Cas: Exons vs. Introns

Abans del Deep Learning, els HMM eren l'eina estàndard per trobar gens en el genoma humà (projectes com GENSCAN).
L'ADN és una cadena de lletres (ACGT). Només certes parts (Exons) codifiquen proteïnes; la resta (Introns) és material no codificant o regulador.

- **Problema:** Al microscopi, tot l'ADN es veu igual. No hi ha espais ni comes.
- **Model HMM:** Els biòlegs saben que els Exons tenen una estadística diferent (més dènols de C-G) i que els Introns tendeixen a tenir A-T. També saben que els gens tenen una longitud mínima.
- **Aplicació:** Entrenem un HMM amb aquestes estadístiques (B) i transicions (A). Passem el genoma d'una persona pacient nova per l'algorisme de Viterbi.
- **Resultat:** L'algorisme ens retorna la segmentació òptima: "De la posició 1000 a la 1500 hi ha un Gen". Això permet diagnosticar malalties genètiques buscant mutacions només a les zones rellevants.

6.6 Deep Ensembles i MC Dropout

En els capítols anteriors hem vist com els models probabilístics (GMM, HMM, Naive Bayes) ens donen una mesura natural de la seva confiança. Però, què passa amb el Deep Learning? Una xarxa neuronal estàndard és, per defecte, **excés de confiada** (*overconfident*). Fins i tot quan s'equivoca completament davant d'una imatge que mai ha vist, la seva sortida Softmax sol ser un vector extrem com $[0.99, 0.01, 0.00]$.

En sistemes crítics (conducció autònoma, medicina), no podem permetre que un model digui "estic 100% segur" quan en realitat no en té ni idea. Necessitem quantificar la incertesa.

Abans de calcular res, hem de distingir entre dos tipus fonamentals de desconeixement (Kendall & Gal, 2017):

1. Incertesa Aleatòria (Random Uncertainty) És el soroll inherent a les dades.

- *Exemple:* Una imatge borrosa d'un número "3" que sembla un "8".
- **Causa:** Error de sensor, oclusió, soroll tèrmic.
- **Propietat:** És **irreductible**. Encara que tinguem infinits dades d'entrenament, la imatge continuarà sent borrosa.

2. Incertesa Epistèmica (Epistemic Uncertainty) És la manca de coneixement del model.

- *Exemple:* Un cotxe autònom entrenat a Califòrnia (sol) veu neu per primera vegada.
- **Causa:** Manca de dades en aquella regió de l'espai $\mathbb{R}^d$. El model no ha vist mai res semblant.
- **Propietat:** És reductible. Si recollim més dades de neu, la incertesa desapareix.

Per entendre-ho bé: Soroll o ignorància?

La incertesa aleatòria és soroll del món: encara que tinguéssim moltes dades, el cas continuaria sent ambigu. La incertesa epistèmica és ignorància del model: amb més dades representatives, podria reduir-se. Si una imatge és borrosa, hi ha incertesa aleatòria; si mostra una malaltia que el model no ha vist mai, hi ha incertesa epistèmica.

Confiança no és calibratge

Una probabilitat de sortida alta no implica necessàriament que el model estigui ben calibrat. Un model pot predir 0.99 i equivocar-se sovint si ha estat entrenat amb dades esbiaixades, si està fora de distribució o si la seva funció de sortida és sobreconfiada. Per això cal distingir:

- **probabilitat predictiva:** el valor numèric que retorna el model;
- **calibratge:** si aquests valors corresponen a freqüències reals d'encert;
- **incertesa fora de distribució:** si l'exemple és diferent del que el model ha vist.

6.6.1 Mètode 1: Deep Ensembles (La Força Bruta)

La manera més senzilla d'estimar la incertesa epistèmica és, de nou, la "Saviesa dels Grups". Lakshminarayanan et al. (2017) van proposar entrenar M xarxes neuronals idèntiques però amb **inicialització aleatòria dels pesos** diferent.

1. Entrenem M xarxes $\{f_{\theta_1}, \ldots, f_{\theta_M}\}$.
2. Per a una nova entrada $\mathbf{x}^*$, obtenim M prediccions.
3. La predicció final és la mitjana: $\mu_* = \frac{1}{M} \sum f_{\theta_i}(\mathbf{x}^*)$.
4. La incertesa és la variància de les prediccions: $\sigma_*^2 = \frac{1}{M} \sum (f_{\theta_i}(\mathbf{x}^*) - \mu_*)^2$.

Si les xarxes discrepen molt (variància alta), vol dir que estem en una zona desconeguda de l'espai. **Problema:** Entrenar i mantenir 5 o 10 xarxes profundes és computacionalment molt car.

6.6.2 Mètode 2: Monte Carlo Dropout

El 2016, Yarin Gal va demostrar que l'ús del **Dropout** (apagar neurones a l'atzar) no és només una tècnica de regularització, sinó una aproximació variacional d'un Procés Gaussià (Bayesian Neural Network).

Normalment, el Dropout només s'activa durant l'entrenament (`train mode`) i es desactiva durant la inferència (`eval mode`). La idea genial és **mantenir el Dropout encès durant la inferència** (test time).

1. Agafem una sola xarxa entrenada.
2. Passem la mateixa dada $\mathbf{x}^*$ per la xarxa T vegades (ex: $T = 50$), mantenint el Dropout actiu.
3. A cada passada, s'apaguen neurones diferents, generant una xarxa "lleugerament diferent" cada cop.
4. Obtenim T prediccions $\{y_1, \ldots, y_T\}$.
5. La mitjana és la predicció robusta. La variància és la incertesa epistèmica.

Això ens permet simular un ensemble de T xarxes sense haver d'entrenar-ne més d'una! És eficient i matemàticament sòlid.

MC Dropout consisteix a mantenir el dropout actiu durant la inferència i fer diverses prediccions amb màscares aleatòries. La dispersió d'aquestes prediccions dona una estimació pràctica de la incertesa del model. Aquesta tècnica es pot interpretar com una aproximació bayesiana sota certes hipòtesis, però no substitueix una anàlisi completa de calibratge ni garanteix detecció fiable d'exemples fora de distribució.

MC Dropout: més enllà de la confiança Softmax

Una probabilitat Softmax alta no garanteix que el model sàpiga què està veient. Un model pot estar ben calibrat en dades semblants a les d'entrenament —per exemple, si diu "70% gat", encerta aproximadament el 70% de les vegades— i, alhora, comportar-se malament davant d'exemples fora de distribució (*out-of-distribution*, OOD).
L'**MC Dropout** estima aquesta incertesa fent diverses prediccions sobre la mateixa entrada amb el dropout actiu durant la inferència. Si les prediccions són molt semblants, el model és estable i la incertesa és baixa. Si canvien molt entre passades, el model està dubtant i la incertesa és alta.
La intuïció és com preguntar moltes vegades el mateix a una persona estudiant, però cada cop tapant-li una part diferent dels apunts. Si domina el tema, respondrà de manera consistent; si no, les respostes variaran. Per això, en sistemes oberts al món real, no n'hi ha prou amb mirar la Softmax: cal comprovar també l'estabilitat de la predicció.

Implementarem una xarxa neuronal simple en PyTorch per fer regressió i visualitzarem com l'MC Dropout captura la incertesa en zones on no hi ha dades.

```
import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)                  # 1. Generar Dades (Sinus amb buit)
X = np.concatenate([np.linspace(-3, -1, 50), np.linspace(1, 3, 50)])
y = np.sin(X * 2) + np.random.normal(0, 0.1, 100)
X_train = torch.tensor(X, dtype=torch.float32).unsqueeze(1)
y_train = torch.tensor(y, dtype=torch.float32).unsqueeze(1)

class Net(nn.Module):               # 2. Definir Model amb Dropout
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(1, 64)
        self.dropout = nn.Dropout(p=0.2) # Dropout actiu
        self.fc2 = nn.Linear(64, 64)
        self.fc3 = nn.Linear(64, 1)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.dropout(self.relu(self.fc1(x)))
        x = self.dropout(self.relu(self.fc2(x)))
        return self.fc3(x)

model = Net()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
criterion = nn.MSELoss()

for epoch in range(1000):           # 3. Entrenament
    optimizer.zero_grad()
    loss = criterion(model(X_train), y_train)
    loss.backward()
    optimizer.step()

# 4. Inferència MC Dropout
model.train() # Mantenir Dropout actiu! (Clau del mètode)
X_test = torch.linspace(-4, 4, 200).unsqueeze(1)
predictions = []

for _ in range(100): # 100 passades estocàstiques
    with torch.no_grad():
        predictions.append(model(X_test).numpy())

predictions = np.array(predictions).squeeze() # Shape (100, 200)
mean_pred = predictions.mean(axis=0)
std_pred = predictions.std(axis=0)

plt.figure(figsize=(10, 6))        # 5. Plotting
plt.plot(X_test, mean_pred, 'b-', label='Predicció Mitjana')
plt.fill_between(X_test.squeeze(),
                 mean_pred - 2*std_pred,
                 mean_pred + 2*std_pred,
                 color='blue', alpha=0.2, label='Incertesa (2$\sigma$)')
plt.scatter(X, y, c='red', s=20, label='Dades d\'Entrenament')
plt.title("MC Dropout: Detecció d'Incertesa Epistèmica", fontsize=14, fontweight='bold')
plt.legend()
```

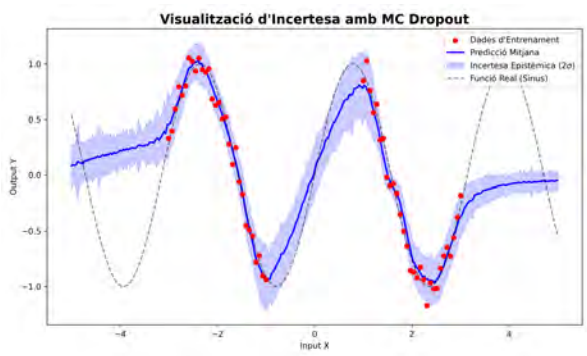


Figura 6.6: Regressió amb incertesa estimada. Observeu la zona central ($-1 < x < 1$) i els extrems. No hi ha dades d'entrenament (punts vermells). En aquestes regions, les 100 passades de la xarxa discrepen, causant que la banda blava (incertesa) s'eixampli. El model "sap que no sap". Una xarxa normal hauria traçat una línia recta confiada i errònia pel mig.

```
plt.tight_layout()
plt.savefig('fig_mc_dropout.png', dpi=300)
plt.show()
```

Listing 6.6: Estimació d'Incertesa amb MC Dropout en PyTorch

Micro-Cas: Conducció Autònoma: El Semàfor Desconegut

En un cotxe autònom, distingir entre incertesa aleatòria i epistèmica és vital.
Un sistema de visió per computador (Segmentació Semàntica) classifica píxels.

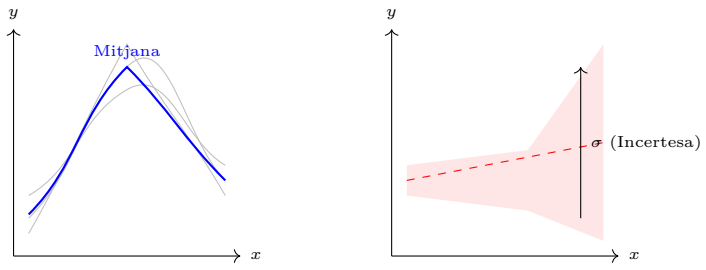
- **Cas A (Nit plujosa):** La imatge és fosca i sorollosa. El model dóna una **Incertesa Aleatòria** alta.
- **Acció:** El cotxe pot continuar, però reduint la velocitat i confiant més en el radar/Lidar (fusió de sensors).
- **Cas B (Objecte estrany):** Un camió bolcat que transporta estàtues gegants de dinosaures. La imatge és nítida (dia assolellat), però el model mai ha vist un dinosaure. La **Incertesa Epistèmica** es dispara.
- **Acció:** El cotxe ha de fer una "parada d'emergència" o demanar control humà remot. No pot "confiar" en la seva predicció ("Això és un arbre"), perquè està al·lucinant.

L'MC Dropout permet mesurar aquest risc en temps real (tot i que requereix maquinari potent per fer T passades en mil·lisegons).

6.7 Conclusions i Debat

En aquest sisè capítol, hem acceptat que la nostra visió del món és imperfecta per combinar moltes opinions imperfectes i reconèixer que el futur és incert.

1. **La Saviesa de la multitud (Ensembles):** Hem après que la mitjana de molts models inestables (Bagging) o la combinació d'experts parcials (Random Forest) redueix la variància sense augmentar el biaix. En enginyeria, la robustesa neix de la diversitat, no de la perfecció individual.
2. **Causalitat i Estructura (Xarxes Bayesianes):** Hem vist que la correlació no és suficient. Per diagnosticar fallades o entendre malalties, necessitem models causals (DAGs) que ens permetin raonar sobre intervencions i efectes comuns.



Síntesi:
1. Diversitat → Estabilitat (Bagging)
2. Temps → Estat Ocult (HMM)
3. Estructura → Causalitat (Bayes Nets)
4. Variància → Epistèmica (MC Dropout)

Figura 6.7: El paradigma del Capítol 6: De la predicció puntual a la distribució de probabilitat i l'estabilitat col·lectiva.

3. **El Temps com a Dimensió (HMM):** Hem incorporat la dinàmica temporal. El món no és estàtic; el passat condiciona el futur. Els HMM ens permeten descodificar la realitat oculta (el clima, la intenció, el gen) a partir de senyals sorollosos.

4. **La Quantificació de la Ignorància:** Finalment, amb MC Dropout, hem dotat les nostres xarxes neuronals de la capacitat més humana de totes: dir "no ho sé".

6.7.1 Debat Ètic: L'Abstenció Algorítmica

Al llarg del llibre hem vist algorismes que classifiquen, prediuen i decideixen. En aquest capítol, hem introduït una nova possibilitat: l'algorisme que es nega a decidir. La capacitat de mesurar la incertesa epistèmica obre la porta a una política de decisió més responsable: permetre que el sistema s'abstingui quan no té prou evidència per actuar automàticament.

Un sistema amb abstenció necessita una regla explícita:

$$\begin{aligned} \text{predir si } \max_c P(y = c \mid x) &\geq \tau, \\ \text{abstenir-se si } \max_c P(y = c \mid x) &< \tau. \end{aligned} \tag{6.21}$$

El llindar τ no s'ha de triar arbitràriament: s'ha d'ajustar en validació segons el compromís entre **cobertura** (quants casos resol automàticament el sistema) i **risc** (quants errors comet entre els casos que sí decideix):

$$\begin{aligned} \text{Cobertura} &= \frac{\text{casos resolts automàticament}}{\text{casos totals}}, \\ \text{Risc} &= \frac{\text{errors entre els casos resolts}}{\text{casos resolts}}. \end{aligned} \tag{6.22}$$

Un bon sistema no busca només maximitzar cobertura, sinó controlar el risc dels casos que decideix resoldre sense intervenció humana. A més cal definir què passa amb els casos abstinguts: revisió humana, prova addicional, segona opinió o aturada segura.

Consells de programació: Si el model s'absté, monitoritza també l'abstenció

Un sistema amb abstenció no es controla només amb accuracy. Guarda tres mètriques: cobertura, risc i percentatge d'abstencions per subgrup. Si el model s'absté molt més en un grup que en un altre, tens un problema d'equitat o de cobertura de dades. La incertesa també s'ha d'auditar.

1. Seguretat: primer, no convertir incertesa en decisió automàtica

En medicina, un error de diagnòstic pot ser fatal. Un sistema de Deep Learning sense mecanisme d'incertesa pot produir una predicció fins i tot davant d'un cas molt diferent dels observats durant l'entrenament. En aquests casos, la sortida del model pot ser excessivament confiada i cal un mecanisme per detectar incertesa o derivar el cas a revisió humana. Amb tècniques com MC Dropout o Deep Ensembles, podem establir un protocol de seguretat:

- Si la incertesa $\sigma < \tau$ (llindar baix), el sistema actua automàticament.
- Si la incertesa $\sigma \geq \tau$, el sistema **s'absté** i deriva el cas a un professional clínic humà.

Això transforma la IA d'un "oracle infal·lible" a una eina de triatge responsable. El debat és: on posem el llindar τ? Qui és responsable si el sistema s'absté en un cas urgent i el personal mèdic no arriba a temps?

2. La Responsabilitat Civil en Conducció Autònoma

Imagineu un cotxe autònom que detecta un obstacle ambigu a la carretera (una bossa de plàstic o una roca?). Un model determinista prendrà una decisió (frenar o no) basada en la classe més probable. I un model probabilístic detectarà una alta incertesa epistèmica ("no sé què és això").

Èticament, hauríem d'exigir per llei que els vehicles autònoms frenin **sempre** que la seva incertesa superi un llindar? Això podria causar "frenades fantasma" constants (falsos positius per seguretat), paralitzant el trànsit. Però no fer-ho implica acceptar un risc calculat d'atropellament. La incertesa no resol el dilema del tramvia, però el quantifica.

3. El Biaix de la Incertesa: Qui és "Desconegut"?

La incertesa epistèmica és alta en regions de l'espai on no tenim dades. Si el nostre dataset d'entrenament té biaixos demogràfics (ex: poques cares de dones negres), el model tindrà una incertesa sistemàticament més alta per a aquest col·lectiu.

- **Conseqüència:** Si utilitzem la incertesa per demanar "verificació humana addicional" (per exemple, en control de fronteres o crèdits), estarem sotmetent a les minories a un escrutini i una molèstia desproporcionats, simplement perquè l'algorisme "no les coneix prou". L'abstenció algorítmica pot convertir-se en una nova forma de discriminació burocràtica.

6.7.2 Preguntes per a l'Aula

1. Enginyeria: La paradoxa de l'Ensemble

Hem dit que per reduir la variància necessitem models **independents**. Però en la pràctica, tots els models d'un ensemble s'entrenen amb el mateix dataset (o versions re-mostrejades).

- **Pregunta:** És possible aconseguir independència real? Quin és el límit teòric de millora d'un Random Forest si les dades tenen soroll intrínsec?

- **Reflexió:** Discutiu la diferència entre l'error reductible (variància del model) i l'error irreductible (soroll de Bayes). Els ensembles ataquen el primer, però són impotents davant del segon.

2. Epistemologia: "No sé què passarà" vs. "No sé res"

Imagineu que heu de predir el llançament d'una moneda i el preu de Bitcoin demà.

- **Cas Moneda:** Sabeu exactament com funciona (50/50), però no podeu predir el resultat. Això és **Incertesa Aleatòria**.
- **Cas Bitcoin:** No sabeu realment quines variables mouen el preu. El vostre model és incomplet. Això és **Incertesa Epistèmica**.
- **Pregunta:** Com a professionals de l'enginyeria, quina d'aquestes dues incerteses es pot reduir amb més pressupost (més dades, més GPUs)? Per què és perillós confondre-les en la gestió de riscos financers?

3. Disseny de Sistemes: El cost de la latència

Un Deep Ensemble de 5 xarxes multiplica per 5 el cost computacional i el temps de resposta (latència) en producció.

- **Escenari:** Un sistema de frenada d'emergència ha de reaccionar en 10ms.
- **Dilema:** És millor tenir una sola xarxa ràpida (que potser s'equivoca amb alta confiança) o un ensemble robust que triga 50ms (i potser detecta l'error, però massa tard per frenar)?
- **Debat:** En quins casos la velocitat és una característica de seguretat més important que la precisió de la incertesa? (Ex: Trading d'alta freqüència vs. Diagnòstic de càncer).

4. Societat: La "Caixa Negra" vs. La "Caixa Borrosa"

Les Xarxes Bayesianes són interpretables (veiem les fletxes de causalitat), però difícils de construir. Els Deep Ensembles funcionen molt bé, però són caixes negres multiplicades per M.

- **Pregunta:** Si una persona professional de l'àmbit clínic ha de triar entre un sistema HMM que entén però que té un 85% d'encert, i un Deep Ensemble opac amb un 95% d'encert i barres d'error ben calibrades, quin hauria de triar segons l'ètica mèdica? La interpretabilitat causal és un requisit indispensable o un luxe intel·lectual?

Tancament docent

Problemes de consolidació.

1. Construir dues mostres bootstrap a partir d'un conjunt petit de dades i explicar per què poden repetir exemples i deixar-ne altres fora. Relacionar-ho amb la reducció de variància en bagging.
2. Comparar un arbre de decisió amb un Random Forest i justificar com l'agregació i la selecció aleatòria de característiques estabilitzen la predicció.
3. Escriure la factorització conjunta d'un petit graf bayesià i identificar les independències condicionals que codifica.
4. Calcular els primers passos de Forward en un HMM senzill i interpretar què representa la probabilitat acumulada.
5. Justificar el camí d'estats més plausible amb Viterbi i explicar per què no sempre coincideix amb la millor decisió local a cada pas.

Tancament docent *(continuació)*

6. Comparar una predicció estable amb una predicció dispersa entre models d'un ensemble i decidir quan cal actuar, abstenir-se o demanar revisió humana.

Criteris breus de resolució. Una bona resolució hauria de mostrar que l'alumnat entén la diferència entre ajustar millor i ser més robust. En els problemes de bootstrap i bagging, cal explicar la diversitat dels models i la reducció de variància, no només descriure el procediment mecànic. En Random Forests, cal vincular la doble aleatorietat amb la descorrelació dels arbres. En xarxes bayesianes i HMM, cal separar clarament dependència, condicionalitat, estat ocult, observació i inferència. En els problemes d'incertesa, cal distingir entre probabilitat predictiva, calibratge i decisió operativa. Una resposta completa ha d'indicar quan el sistema pot decidir automàticament, quan ha d'abstenir-se i quin circuit de revisió s'activa.

Connexió amb pràctica/laboratori. Aquest capítol es desplega en una pràctica de robustesa i incertesa. L'alumnat pot entrenar un arbre individual i un Random Forest, comparar variància i importància de variables, estimar la dispersió de prediccions entre models, visualitzar una banda d'incertesa i definir una regla simple d'abstenció. En una extensió seqüencial, pot implementar un HMM petit o simular una seqüència d'estats ocults per entendre Forward i Viterbi com a formes de raonament temporal.

Capítol 7

Conclusions

Arribem al final del nostre recorregut. Hem passat de la representació vectorial i les distàncies a l'optimització, la classificació, els arbres de decisió, el raonament bayesià, l'aprenentatge no supervisat, els ensembles, les seqüències i la incertesa. Hem après que construir un model no és només aplicar una funció de predicció, sinó decidir com representem les dades, què optimitzem, com validem i quins errors considerem acceptables.

Però, com a professionals de l'enginyeria, la feina no acaba quan el model obté una bona mètrica en un conjunt de test intern o en un benchmark controlat. De fet, sovint és aquí on comencen els problemes reals: biaixos heretats, *data leakage*, manca d'explicabilitat, absència de monitoratge, canvi de distribució, costos energètics i responsabilitat sobre les persones afectades.

En aquest capítol final, integrarem un marc de responsabilitat professional. L'objectiu és entendre que un sistema d'aprenentatge automàtic només és una solució d'enginyeria quan és robust, auditable, mantenible i socialment responsable.

Guia d'activació docent

Repte inicial. Amb model amb un 99% d'accuracy en predicció de risc mèdic, abans de desplegar-lo, cal preguntar-se: quina és la prevalença de la classe crítica, quants falsos negatius comet, hi ha leakage, i el model funciona igual de bé en tots els subgrups?

Checkpoints conceptuals.

1. Situar els models del curs segons interpretabilitat, flexibilitat, risc d'overfitting, cost computacional i dependència de dades.
2. Auditar un pipeline detectant leakage, sobrevalidació, manca d'explicabilitat, absència de monitoratge, drift i riscos ètics.

Errors típics provocats.

1. Acceptar una mètrica alta com a prova suficient de qualitat.
2. Tractar l'ètica com un apartat final i no com una decisió de disseny.

Microcas professional. Comitè de desplegament d'un model de selecció de personal (*escenari d'auditoria il·lustratiu*): el rendiment aparent és bo, però l'auditoria detecta proxies sensibles, manca d'explicació individual i absència de monitoratge.

Connexió ètica o d'enginyeria. L'equip d'enginyeria no només entrena un model: dissenya, valida, desplega, monitoritza i assumeix les conseqüències del sistema.

7.1 La Gran Síntesi: De la Lògica a la Dada

Podem classificar totes les tècniques apreses en un espectre continu definit per la quantitat de **coneixement a priori** que imposem versus la quantitat de **dades** que necessitem:

1. **Lògica i Regles (Cap. 1-2):**
   - *Paradigma:* Deductiu. "Si A llavors B".

- *Avantatge:* Totalment interpretable i no necessita dades (només persones expertes).
- *Límit:* No escala a problemes complexos o sorollosos (ex: reconèixer una cara amb regles 'if' és impossible).

2. **Models Geomètrics i Paramètrics (Cap. 3-4):**
   - *Paradigma:* Inductiu amb forta restricció de forma (Biaix). SVM, Regressió Logística.
   - *Mecànica:* Assumim una forma (hiperplà) i optimitzem paràmetres $(\mathbf{w}, b)$ amb Gradient Descent.
   - *Límit:* La linealitat (si no usem Kernels) i la necessitat d'enginyeria de característiques manual.
3. **Models No Paramètrics i Ensembles (Cap. 5-6):**
   - *Paradigma:* Adaptatiu (Variança). KNN, Arbres, Random Forest.
   - *Mecànica:* La complexitat del model creix amb les dades. Combinem models febles per reduir la variança.
   - *Límit:* Risc d'overfitting massiu i cost computacional en inferència.

Aquest espectre és una eina conceptual per comparar famílies de models, però no s'ha d'interpretar com una jerarquia universal. Un model simple pot rendir millor que un model complex si les dades són escasses o sorolloses; un model complex pot ser auditable si es combina amb bons protocols d'explicabilitat; i el cost computacional només té sentit en relació amb l'ús previst, la mida de les dades i els requisits de desplegament.

Per visualitzar com cada "filosofia" resol el mateix problema, compararem tres paradigmes sobre un dataset complex (Moons).

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from matplotlib.colors import ListedColormap

# 1. Generació de Dades Complexes
X, y = make_moons(n_samples=400, noise=0.3, random_state=42)

# 2. Definició de Models Representatius
models = [
    ("Arbre de Decisió", DecisionTreeClassifier(max_depth=5)),
    ("SVM (RBF Kernel)", SVC(gamma=2, C=1)),
    ("KNN (k=3)", KNeighborsClassifier(n_neighbors=3))
]

fig, sub = plt.subplots(1, 3, figsize=(18, 5))    # 3. Plotting
h = .02  # step size in the mesh
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

```

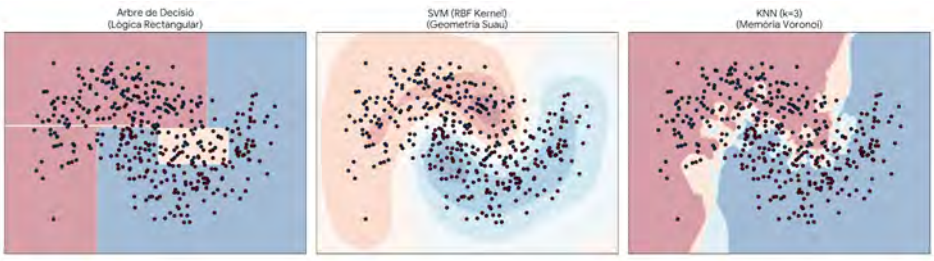


Figura 7.1: Comparativa de Paradigmes. (Esquerra) L'**Arbre** talla l'espai en rectangles ortogonals: és una aproximació lògica "a cops de destral". (Centre) La **SVM** amb kernel RBF crea contorns suaus i orgànics, basats en la geometria de la distància. (Dreta) El **KNN** fragmenta l'espai en cel·les de Voronoi irregulars, basant-se purament en la memòria local. Cada un resol el problema, però la forma de la solució revela la seva naturalesa matemàtica.

```
for clf, (name, model) in zip(models, sub):
    ax = model
    clf[1].fit(X, y)
    Z = clf[1].predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    name_ax = clf[0]
    ax = model

    # Subplots
    sub[models.index(clf)].pcolormesh(xx, yy, Z, cmap=plt.cm.RdBu, alpha=0.3)
    sub[models.index(clf)].scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdBu_r, edgecolors='k',
     s=20)
    sub[models.index(clf)].set_title(name, fontsize=14, fontweight='bold')
    sub[models.index(clf)].set_xticks(())
    sub[models.index(clf)].set_yticks(())

plt.tight_layout()
plt.savefig('fig_meta_comparison.png', dpi=300)
plt.show()
```

Listing 7.1: Arbres (Lògica) vs SVM (Geometria) vs KNN (Memòria)

Per entendre-ho bé: Aquest mapa no és una classificació absoluta

La Figura 7.1 ajuda a situar famílies de models, però no diu quin model és millor sempre. Un model lineal pot guanyar si les dades són petites i netes; un model complex pot ser necessari si el patró és altament no lineal. La decisió no depèn només del model: depèn de les dades, la mètrica, la validació, el cost d'error i el context de desplegament.

7.2 Història dels Biaixos

L'aforisme més famós en ciència de dades és *"Garbage In, Garbage Out"* (GIGO). Si entrenem un model amb dades escombraries, obtindrem prediccions escombraries, per molt sofisticat que sigui l'algorisme. Però què passa quan les dades no són "escombraries" tècniques, sinó un reflex fidel d'una societat injusta?

El biaix algorítmic no sol ser intencionat, sorgeix de la distribució de probabilitat subjacent. Suposem que volem predir l'èxit laboral Y. Tenim característiques X i un atribut protegit A (gènere, raça).

Un model està esbiaixat si:

$$P(\hat{Y} = 1|A = a) \neq P(\hat{Y} = 1|A = b) \tag{7.1}$$

Això es coneix com a **Disparitat d'Impacte**. Les fonts principals són:

1. **Biaix Històric:** Les dades d'entrenament reflecteixen decisions humanes passades (ex: jutges que dictaven sentències més dures a minories). El model aprèn a imitar aquest prejudici per minimitzar l'error d'entrenament.
2. **Biaix de Mostreig:** Si el dataset conté un 90% d'homes i un 10% de dones, el model optimitzarà l'error global ignorant el subgrup minoritari. L'error en el grup minoritari pot ser enorme sense afectar gaire l'Accuracy total.
3. **Variables Proxy:** Eliminar la columna "Gènere" no soluciona res. El model trobarà correlacions latents (ex: tipus d'estudis, horaris, aficions) que actuen com a *proxies* del gènere.

El següent cas s'utilitza com a exemple històric de com un model pot aprendre patrons discriminatoris si les dades d'entrenament reflecteixen decisions històriques esbiaixades: El 2015, Amazon va retirar un sistema d'IA per filtrar currículums.

- **Entrenament:** 10 anys de CVs enviats a l'empresa (majoritàriament homes).
- **Resultat:** L'algorisme va aprendre a penalitzar la paraula "women's" (com en "women's chess club captain") i a afavorir llenguatge típicament masculí ("executed", "captured").
- **Causa:** L'algorisme va concloure correctament (matemàticament) que, històricament, ser home correlacionava amb ser contractat. Va convertir una realitat descriptiva en una norma prescriptiva.

A continuació simularem un escenari on dos grups tenen diferents taxes de Falsos Positius, una mètrica clau en justícia algorítmica.

```
import matplotlib.pyplot as plt
import numpy as np

# Dades simulades
groups = ['Grup Majoritari (A)', 'Grup Minoritari (B)']
accuracy = [0.95, 0.94]  # L'Accuracy sembla igual
fpr = [0.05, 0.25]       # False Positive Rate (Inocents marcats com culpables)
fnr = [0.05, 0.30]       # False Negative Rate (Culpables que s'escapen)

x = np.arange(len(groups))
width = 0.35

fig, ax = plt.subplots(figsize=(10, 6))
rects1 = ax.bar(x - width/2, accuracy, width, label='Accuracy Global', color='lightgray')
rects2 = ax.bar(x + width/2, fpr, width, label='Falsos Positius (FPR)', color='salmon')

ax.set_ylabel('Taxa / Probabilitat')
ax.set_title('La Il·lusió de l\'Accuracy: Disparitat d\'Impacte', fontsize=16, fontweight='
    bold')
ax.set_xticks(x)
ax.set_xticklabels(groups)
ax.legend()

# Afegir etiquetes
ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)
```

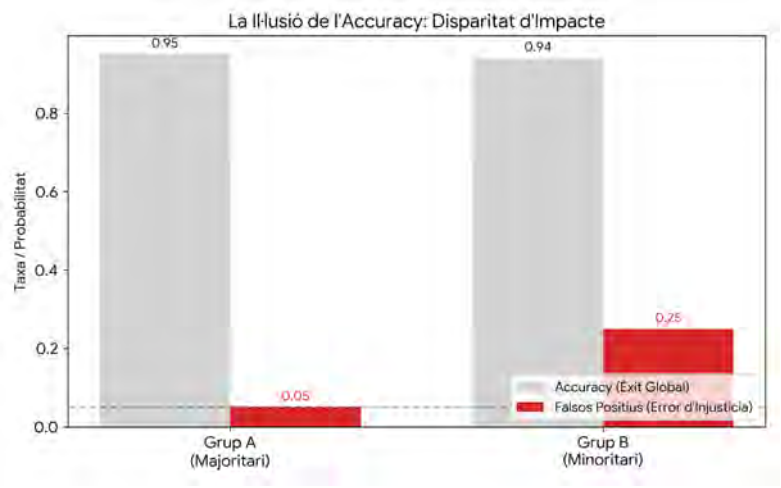


Figura 7.2: Gràfic de Disparitat. Tot i que l'*Accuracy* global és similar per als dos grups (barres grises), el Grup B pateix una taxa de **Falsos Positius** (barres vermelles) cinc vegades superior.

```
26
27 plt.axhline(y=0.05, color='r', linestyle='--', alpha=0.5, label='FPR Ideal')
28 plt.tight_layout()
29 plt.savefig('fig_bias_fairness.png', dpi=300)
30 plt.show()
```

Listing 7.2: Anàlisi de Fairness: Taxes d'Error per Grup

7.3 La Caixa Negra vs. Explicabilitat (XAI)

En enginyeria, no ens podem permetre "màgia". Si un pont cau, hem de saber per què. Si un model mèdic falla, hem de saber per què. El Deep Learning ha creat models "Black Box": milions de paràmetres (w_{ij}) impossibles d'interpretar per un humà. Això ha donat lloc al camp de l'**eXplainable AI (XAI)**.

Existeix sovint una tensió entre flexibilitat i interpretabilitat directa:

- **Models directament interpretables:** regressió lineal, regressió logística simple, arbres curts o regles. Poden ser molt competitius si el patró és simple o les dades són limitades.
- **Models més flexibles:** Random Forests, Gradient Boosting o xarxes neuronals. Poden capturar patrons complexos, però solen requerir eines d'explicació, validació i monitoratge més estrictes.

Per tant, la pregunta professional no és només "quin model rendeix més?", sinó "quin model podem justificar, auditar i mantenir en el context d'ús previst?".

En aquest context, els valors SHAP (*Shapley Additive Explanations*) són una de les famílies d'explicacions post-hoc més utilitzades per interpretar prediccions individuals. Es basen en idees de la teoria de jocs cooperatius: imaginem que la predicció del model és un resultat que s'ha de repartir entre les característiques que hi han contribuït. Així doncs, el valor SHAP ϕ_i estima quant ha contribuït la característica i a moure la predicció des del valor base $\mathbb{E}[f(x)]$ fins al valor final $f(x)$:

$$f(x) = \phi_0 + \sum_{i=1}^{d} \phi_i. \tag{7.2}$$

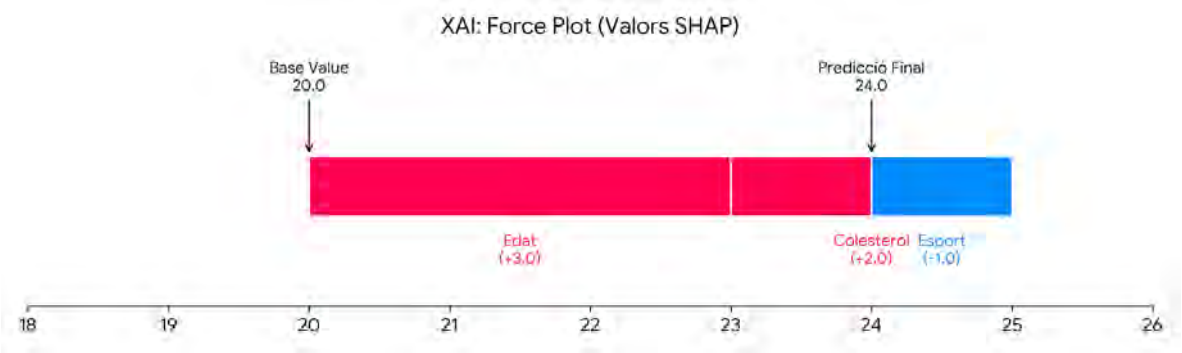


Figura 7.3: Esquema conceptual d'un *force plot* de SHAP. Partint d'un valor base, cada característica contribueix a desplaçar la predicció del model cap amunt o cap avall. Els vectors mostren contribucions del model per a una instància concreta; no s'han d'interpretar automàticament com a causes reals del fenomen.

La propietat clau és que la suma de les contribucions reconstrueix la predicció del model per a aquella instància. Això proporciona una explicació local del comportament del model, no una explicació causal del fenomen real. Segons el tipus de model i l'explainer utilitzat, el càlcul pot ser exacte o aproximat: una contribució SHAP alta no implica causalitat.

Per entendre-ho bé: SHAP explica el model, no el món

Si SHAP diu que l'edat ha augmentat la predicció, vol dir que el model ha utilitzat l'edat per moure aquella predicció. No vol dir automàticament que l'edat sigui la causa real del fenomen. SHAP és una eina per entendre el comportament del model, no una prova causal.

Explicació intuïtiva: atribuir contribucions

Un cotxe té un preu predit de 30.000€, mentre que el preu mitjà del conjunt de dades és 20.000€. SHAP pot descompondre aquesta diferència de +10.000€ en contribucions associades a les variables:

- Motor potent: +5.000€.
- Marca de luxe: +8.000€.
- Molts quilòmetres: -3.000€.

La suma de les contribucions explica com el model ha arribat a aquesta predicció. Això no vol dir que aquestes variables siguin causes reals del preu, sinó que són les contribucions que el model ha utilitzat en aquesta instància.

7.4 El Perill del "Data Leakage" i la Sobrevalidació

Un dels errors més greus en aprenentatge automàtic és presentar resultats excel·lents que després no es reprodueixen en producció perquè el model havia tingut accés, durant l'entrenament, a informació que no estarà disponible en el moment real de la predicció. Aquest fenomen s'anomena **data leakage** o fuita d'informació.

Els tipus de Leakage són:

1. **Target leakage:** una variable codifica directament o indirectament l'etiqueta. Exemple: predir diagnòstic utilitzant una medicació que només es dona després del diagnòstic.
2. **Leakage temporal:** el model utilitza informació del futur per predir el passat o el present.

3. **Leakage de preprocessament:** escalar, imputar, fer PCA o seleccionar atributs utilitzant tot el dataset abans de separar train/test.
4. **Leakage per grups:** dades de la mateixa persona pacient, persona usuària, dispositiu o centre apareixen tant a entrenament com a test.
5. **Contaminació del test set:** usar el test repetidament per prendre decisions de modelatge o hiperparàmetres.

Per entendre-ho bé: Data leakage és copiar sense adonar-se'n

En un examen, seria trampa veure una part de la solució abans de respondre. En machine learning passa igual: si el model veu informació del futur, del test o de l'etiqueta de manera indirecta, sembla molt intel·ligent, però només està copiant pistes que no tindrà en producció.

Consells de programació: No deixis cap `fit()` fora del pipeline

Si una transformació calcula mitjanes, desviacions, imputacions, vocabulari, PCA o selecció d'atributs, és part del model. Ha d'estar dins d'un `Pipeline` o ajustar-se només amb train. Un sol `fit()` abans del split pot convertir una bona pràctica en una estimació contaminada.

El codi següent no pretén entrenar un model complet, sinó visualitzar una situació típica: una corba de rendiment realista i una corba sospitosament alta causada per fuita d'informació. L'objectiu és aprendre a reconèixer el patró d'alarma, no substituir una auditoria formal del pipeline.

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)        # Simulem dades
n_samples = 100
y = np.random.randint(0, 2, n_samples)      # Variable objectiu
# Variable legítima (sorollosa)
X_legit = y * 0.2 + np.random.normal(0, 1, n_samples)
# Variable amb Leakage (proxy directe de y amb una mica de soroll)
# Ex: "Ha pres medicació post-cirurgia?" per predir "Ha tingut cirurgia?"
X_leak = y * 0.95 + np.random.normal(0, 0.1, n_samples)

# Resultats simulats de models
acc_real = [0.55, 0.60, 0.62, 0.61, 0.63] # Model realista
acc_leak = [0.98, 0.99, 0.99, 0.99, 0.99] # Model amb leakage

epochs = range(1, 6)

plt.figure(figsize=(10, 6))
plt.plot(epochs, acc_real, 'b-o', label='Model Correcte (Realitat)', linewidth=2)
plt.plot(epochs, acc_leak, 'r--s', label='Model amb Leakage (Ficció)', linewidth=2)

plt.title('Detecció de Data Leakage: Quan és "Massa bo per ser cert"', fontsize=14,
    fontweight='bold')
plt.xlabel('Èpoques / Iteracions')
plt.ylabel('Accuracy en Validació')
plt.ylim(0.5, 1.05)
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
```

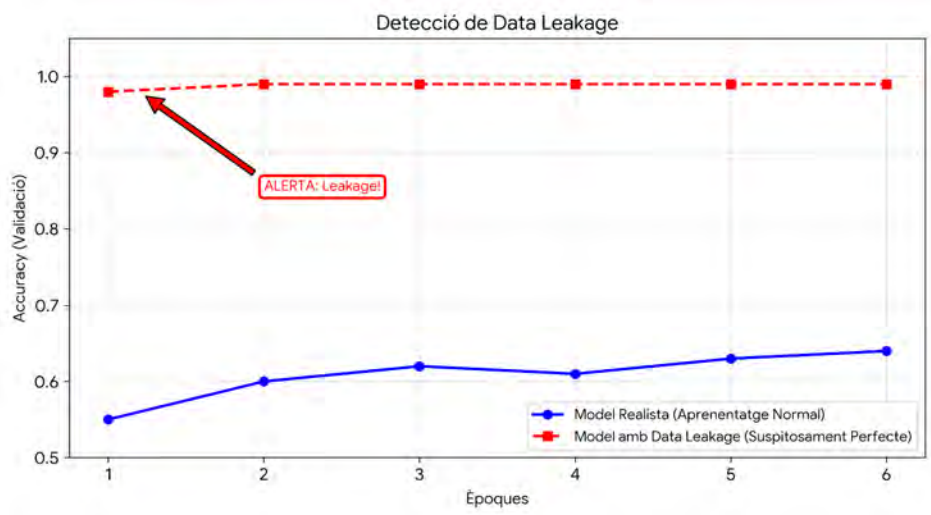


Figura 7.4: Senyal d'alarma de *data leakage*. Si un model obté una accuracy extraordinàriament alta des de la primera iteració en un problema complex, cal sospitar que alguna informació de l'etiqueta o del futur ha entrat indegudament al conjunt d'entrenament.

```
29
30 plt.text(2.5, 0.90, "ALERTA!\nRendiment quasi perfecte\nimmediatament sol indicar\nfiltració
       de la variable objectiu.",
31          color='red', fontweight='bold', bbox=dict(facecolor='white', edgecolor='red'))
32
33 plt.tight_layout()
34 plt.savefig('fig_leakage.png', dpi=300)
35 plt.show()
```

Listing 7.3: Simulació de Data Leakage

Per entendre-ho bé: Una mètrica massa bona és una alarma, no una prova

Si un model obté un rendiment extraordinari en un problema difícil, no vol dir automàticament que hi hagi leakage. Però sí vol dir que cal auditar el pipeline: variables disponibles en el futur, contaminació del test, duplicats, preprocessament fet abans del split o etiquetes filtrades indirectament.

7.5 L'Equip d'enginyeria de Dades: Perfil Professional

A la indústria, entrenar un model en un notebook és només una part del treball. Una part molt important de l'enginyeria real consisteix a portar aquest model a producció: dades fiables, versions, tests, desplegament, monitoratge, detecció de drift i manteniment (**MLOps**, (Machine Learning Operations).

MLOps implica, abans de desplegar un model, redactar una fitxa breu amb:

- objectiu del model i ús previst;
- dades d'entrenament i limitacions conegudes;
- mètriques globals i per subgrups;
- riscos de leakage, biaix i drift;
- criteris d'abstenció, revisió humana o retirada;
- versió del model i responsable de manteniment.

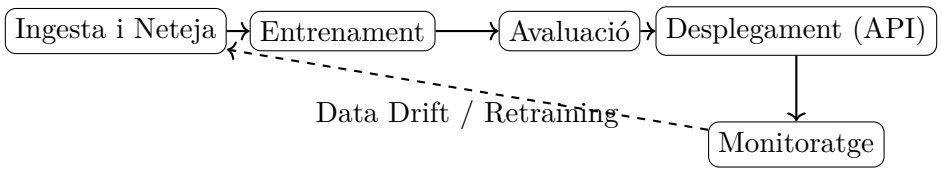


Figura 7.5: El cicle MLOps. El sistema no acaba amb el desplegament. El món canvia (Data Drift), i el model es degrada. El monitoratge tanca el bucle per reentrenar el sistema automàticament.

Data Scientist (Científic)	**Data Engineer / ML Engineer**
Focus: Descobrir patrons, matemàtiques.	Focus: Robustesa, escalabilitat, latència.
Eina: Jupyter Notebooks (prototipatge).	Eina: Docker, Kubernetes, CI/CD (producció).
Output: Un informe o un model '.pkl'.	Output: Una API REST que serveix 10k peticions/seg.
Preocupació: "Com pujo l'AUC?"	Preocupació: "Què passa si cau el servidor?"

Taula 7.1: Rols professionals en l'ecosistema de dades.

Aquest document converteix el model en un artefacte auditable, no només en un notebook que funciona.

Per entendre-ho bé: MLOps és mantenir viu el model

Un model en producció no és una foto fixa: és un sistema viu. Les dades canvien, les persones usuàries canvien i els errors apareixen. MLOps és el conjunt de pràctiques que permet saber quina versió del model funciona, amb quines dades es va entrenar, quan s'està degradant i com tornar enrere si alguna cosa falla.
Abans de desplegar un model, escriu una fitxa amb: objectiu, dades d'entrenament, població prevista, mètriques globals i per subgrups, limitacions, riscos coneguts, criteris d'abstenció, responsable del model i data de revisió. Si no pots documentar aquests punts, encara no tens un sistema; tens un experiment.

Així, un projecte d'IA professional segueix el cicle infinit descrit en la Figura 7.5:

1. Versionat de dades, codi, hiperparàmetres i model entrenat.
2. Separació clara entre entrenament, validació i test.
3. Tests automàtics de qualitat de dades i coherència de l'esquema.
4. Registre de mètriques globals i mètriques per subgrups rellevants.
5. Monitoratge de *data drift* i *concept drift*.
6. Protocol de reentrenament i criteris de retirada del model.
7. Capacitat de *rollback* a una versió anterior.
8. Documentació de riscos, limitacions i domini operacional.

Consells de programació: Checklist mínima abans de desplegar

Abans de desplegar un model, comprova:

- dades, codi i model versionats;
- pipeline reproduïble de train a inferència;
- mètriques globals i per subgrups;
- test independent no utilitzat per tuning;
- monitoratge de drift i degradació;
- pla de rollback si el model falla;
- responsable humà del manteniment.

A més, escriu les condicions que obligaran a aturar, revertir o reentrenar el model: caiguda de mètrica, drift de dades, augment d'errors en un subgrup, incidències crítiques o canvi del context d'ús. Un model sense criteris de retirada és un risc latent.

Un model sense aquests elements és un experiment, no encara un sistema d'enginyeria.

7.6 Ètica i Honestedat Acadèmica/Professional

L'enginyeria no és una disciplina neutra. Un pont mal calculat cau; un algorisme mèdic mal entrenat mata. En el segle XX, l'ètica de l'enginyeria se centrava en la física (materials, estructures). En el segle XXI, el risc s'ha traslladat al codi i a les dades. De fet, la potència de la IA Generativa (ChatGPT, Claude, Copilot) i la ubiqüitat dels sistemes de decisió automàtica plantegen nous reptes que no es resolen només amb habilitat tècnica, sinó amb integritat professional.

Les eines d'IA generativa poden accelerar moltes tasques: escriure fragments de codi, generar notebooks inicials, proposar tests o documentar funcions. Però no substitueixen la responsabilitat de qui dissenya el sistema. El criteri professional continua sent validar resultats, detectar errors subtils, entendre les dades, auditar riscos i decidir si el sistema és segur per al context d'ús.

1. La Trampa de la Competència Il·lusòria

Quan l'alumnat utilitza ChatGPT per resoldre una pràctica sencera, no està simplement delegant una tasca avorrida; està **delegant el procés d'aprendre**.

- **L'efecte:** l'alumnat entrega un codi correcte que funciona.
- **La realitat:** l'alumnat no ha construït el model mental del problema.
- **La conseqüència:** quan aquesta futura persona enginyera arribi a una empresa i el sistema de producció falli de manera subtil, no tindrà la intuïció necessària per depurar-lo, perquè no s'ha barallat amb els errors bàsics durant la seva formació.

2. Al·lucinacions i Responsabilitat Tècnica

Els LLMs són "lloros estocàstics" (Bender et al., 2021). No tenen un model de veritat, només de probabilitat de la següent paraula.

- **El Risc:** La IA inventa llibreries que no existeixen (*Hallucination*) o, pitjor encara, suggereix paquets de programari maliciosos (Supply Chain Attack) que tenen noms similars als reals.
- **El Deure:** Una persona experta en enginyeria que fa *Copy-Paste* sense verificar la documentació oficial o testar el codi està cometent una negligència professional. En un

judici per errors de software, "M'ho va dir el ChatGPT" no és una defensa vàlida; és una confessió d'incompetència.

La Regla del "Copilot"

A la indústria, l'ús de Github Copilot és estàndard. Però hi ha una regla d'or: **"Mai acceptis una línia de codi generada per IA que no series capaç d'escriure tu mateix (encara que et costés més temps)."** Si no entens el 'lambda' o la comprensió de llista que t'ha suggerit la IA, i l'acceptes, has introduït una vulnerabilitat tècnica i ètica al sistema. Ets responsable de codi que no entens.

L'ètica no és només evitar errors involuntaris; és resistir la pressió corporativa per dissenyar sistemes maliciosos.

El Cas Volkswagen (Dieselgate): El "Defeat Device" El 2015 es va descobrir que 11 milions de cotxes tenien un software dissenyat explícitament per fer trampes.

- **L'Enginyeria:** El codi llegia l'angle del volant, la velocitat i la durada del motor encès. Si el volant no es movia i el patró coincidia amb el cicle de test de laboratori (NEDC), el cotxe entrava en "Mode Ecològic".
- **La Realitat:** A la carretera, el cotxe desactivava els controls d'emissions per guanyar potència, emetent fins a 40 vegades més òxids de nitrogen (NO_x) del permès.
- **El Dilema:** Aquest codi no es va escriure sol. Un equip sènior d'enginyeria de software el va dissenyar, implementar i ocultar sota noms de variables críptics.
- **La lliçó:** La responsabilitat professional no desapareix perquè una decisió vingui imposada per pressions internes o interessos de negoci. Els codis deontològics de l'enginyeria recorden que la seguretat pública, la salut i la legalitat estan per sobre de l'obediència acrítica a l'organització.

El Cas Therac-25: Quan la "Race Condition" mata Als anys 80, una màquina de radioteràpia va matar 6 persones pacients per sobredosi de radiació.

- **L'Error:** El software tenia una "condició de carrera" (*Race Condition*). Si l'operador teclejava les ordres molt ràpid (en menys de 8 segons), el sistema no actualitzava correctament la posició del filtre protector.
- **L'Ètica:** El codi havia estat escrit per un sol programador, sense revisió de codi (*Code Review*) i sense tests unitaris adequats. L'empresa va respondre als primers informes d'errors dient "És impossible que la màquina falli".
- **Lliçó:** La qualitat del codi (testing, concurrència ben gestionada) és una qüestió ètica quan hi ha vides en joc.

Consells de programació: L'ètica també és fer tests

Quan un sistema pot afectar persones, la qualitat del codi és una qüestió ètica. No n'hi ha prou amb bones intencions: cal revisió de codi, tests unitaris, tests d'integració, registres d'errors, traçabilitat de dades i procediments de resposta a incidències. Un bug silenciós en producció també és una decisió de disseny.

Un nou vector ètic és la sostenibilitat. Els models d'IA no només tenen un cost d'entrenament i inferència en diners i temps, sinó també en energia. En projectes professionals, una millora

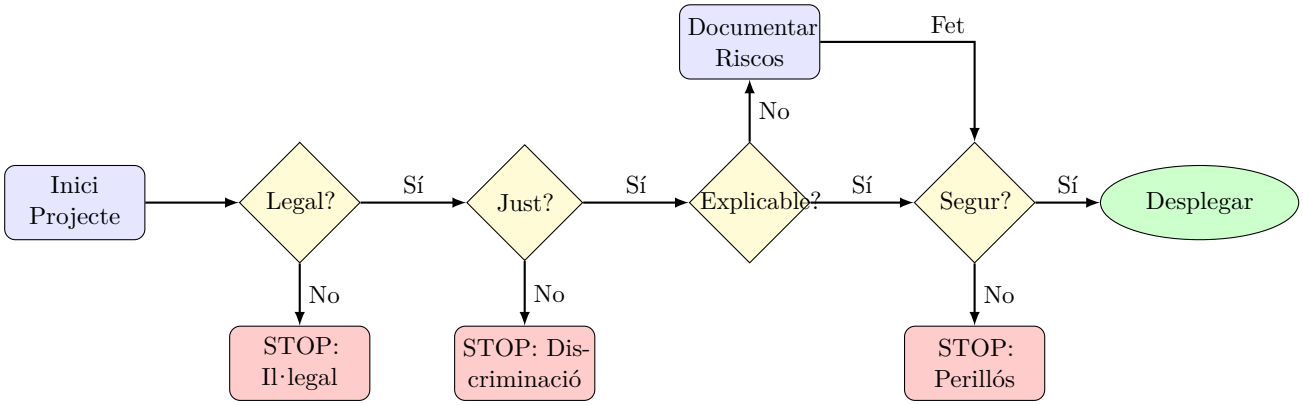


Figura 7.6: **L'Algorisme de l'Ètica (Flux Horitzontal)**. Abans d'escriure codi, el projecte ha de superar aquests filtres seqüencials. Noteu com la manca d'explicabilitat no atura el projecte necessàriament, però obliga a un procés extra de documentació de riscos, mentre que la il·legalitat o la injustícia són condicions d'aturada immediata.

petita de mètrica s'ha de comparar amb el cost computacional, la latència, la petjada energètica i la necessitat real del cas d'ús. Entrenar models d'IA consumeix quantitats massives d'energia:

- **La dada a verificar:** entrenar models grans pot implicar un consum energètic i una petjada de carboni significatius. Aquesta petjada depèn del nombre de paràmetres, les hores d'entrenament, el hardware, l'eficiència del centre de dades i el mix energètic. Si es dona una xifra concreta d'emissions, cal citar-ne la font, l'any i la metodologia de càlcul.
- **Red AI vs. Green AI:**
  - **Red AI:** prioritzar petites millores de mètrica sense considerar el cost computacional, energètic o econòmic.
  - **Green AI:** valorar també l'eficiència: models més petits, destil·lació, quantització, reutilització de models preentrenats, inferència eficient i desplegament en dispositius locals quan sigui adequat.

La pregunta ètica és: "El meu model millora la precisió un 1% respecte a l'anterior, però costa 10 vegades més energia entrenar-lo i executar-lo. Val la pena desplegar-lo?" La resposta no és tècnica, és un judici de valor sobre el cost d'oportunitat i l'impacte ambiental.

En la Figura 7.6, presentem un "algorisme ètic" per decidir si utilitzar una determinada dada o model.

Finalment, hem de considerar la perspectiva de la persona usuària. El Reglament General de Protecció de Dades (GDPR) europeu estableix conceptes que afecten directament l'arquitectura de dades:

1. **Dret a l'Oblit:** Si una persona usuària demana esborrar les seves dades, i heu entrenat un model d'IA amb aquestes dades, heu d'esborrar el model? *Resposta d'Enginyeria:* És un problema obert (Machine Unlearning). Si el model ha memoritzat dades personals identificables o permet recuperar informació sensible d'una persona, pot plantejar problemes legals i tècnics greus. Aquest és un dels motius pels quals el *machine unlearning*, la minimització de dades i les auditories de memorització són línies importants d'enginyeria responsable.
2. **Dret a l'Explicació (Art. 22):** Un ciutadà té dret a no ser objecte d'una decisió basada *únicament* en tractament automatitzat si aquesta té efectes jurídics. Això implica

que dissenyar sistemes "Human-in-the-loop" no és opcional en molts casos, sinó obligatori.

Consells de programació: No reportis només accuracy

Quan comparis dos models, reporta també temps d'entrenament, latència d'inferència, memòria, cost estimat i mida del model. Un model que millora un 0.2% la mètrica però multiplica per deu el cost pot ser una mala decisió d'enginyeria.

7.7 Conclusions Finals i Debat Obert

Hem arribat al final d'aquest recorregut. Hem viatjat des de la lògica simbòlica i determinista dels primers capítols (CRI), on el món es podia descriure amb regles *If-Then*, fins a la complexitat probabilística i dimensional de l'aprenentatge profund (APC), on el coneixement emergeix de milions de paràmetres ajustats per un gradient.

Tancar aquest llibre no és un comiat, sinó una benvinguda a la vostra carrera professional. Abans de tancar l'editor de codi, és imperatiu que reflexionem sobre les tres lliçons fonamentals que distingeixen una persona programadora d'una enginyera de Dades i Intel·ligència Artificial.

Lliçó 1: La Desmitificació (L'IA no és Màgia, és Matemàtica)

Vivim en una època d'hipèrbole mediàtica. Es parla de "consciència artificial", de màquines que "pensen" o "senten". Com a professionals de l'enginyeria que heu construït aquests sistemes des de zero, teniu el deure intel·lectual de ser immunes a aquesta mística.

Durant aquest curs heu obert la caixa negra i heu vist què hi ha a dins:

- No hi ha "pensament", hi ha **optimització de funcions de cost** en espais no convexos.
- No hi ha "intuïció", hi ha **reconeixement de patrons** basat en la mètrica de distància en $\mathbb{R}^d$.
- No hi ha "creativitat", hi ha **interpolació i generació** dins d'una distribució de probabilitat apresa ($P(X)$).

Aquesta visió reduccionista no treu mèrit a la disciplina; al contrari, ens empodera. Com l'Aprenentatge Computacional és matemàtica, la podem analitzar, auditar, corregir i millorar. Quan un model falla, no és un "misteri insondable", és un problema de gradient esvaint, de biaix en el dataset o de sobreajustament (overfitting). Tenim les eines (teoremes, mètriques, diagnòstics) per arreglar-ho.

El Deure de l'Explicació

En la vostra vida professional, sovint sereu l'única persona a la sala que entén com funciona realment el sistema. Quan una persona directiva o clienta us demani "màgia", el vostre deure ètic és explicar les limitacions reals (marge d'error, dependència de les dades, manca de causalitat). L'honestedat tècnica és la base de la confiança en l'enginyeria.

Lliçó 2: El Canvi de Paradigma (Software 2.0)

Estem vivint una transició històrica en la manera de construir programari, un concepte que Andrej Karpathy anomena **Software 2.0**.

- **Software 1.0 (El Codi Clàssic):** L'ésser humà escriu la lògica explícita (C++, Python).

Nosaltres diem a la màquina *com* fer la tasca pas a pas.

$$\text{Programa} + \text{Input} \rightarrow \text{Output}$$

- **Software 2.0 (Machine Learning):** L'ésser humà defineix l'objectiu (la funció de cost) i proporciona els exemples (les dades). L'optimitzador "escriu" el programa (els pesos de la xarxa neuronal).

$$\text{Input} + \text{Output Desitjat} \xrightarrow{\text{Optimització}} \text{Programa}$$

Això implica que **les dades són el nou codi**. En el desenvolupament tradicional, si el programa fallava, anàvem a GitHub i corregíem una línia de codi (un *bug*). En el Software 2.0, si el model falla, sovint no toquem el codi de l'arquitectura; "arreglem" el dataset. Netegem etiquetes errònies, afegim exemples de casos límit o re-balancegem classes.

La qualitat del vostre producte final ja no depèn només de la vostra habilitat per escriure algoritmes eficients (encara que ajuda), sinó de la vostra capacitat per dissenyar, curar i mantenir **datasets d'alta qualitat**. L'enginyeria de dades és l'arquitectura sobre la qual s'aguanta tot l'edifici de la IA.

Per entendre-ho bé: Les dades són el nou codi

En el software clàssic, si el programa falla, busquem el bug al codi. En machine learning, moltes vegades el bug és a les dades: etiquetes incorrectes, exemples que falten, classes descompensades o casos límit no representats. Millorar el model sovint vol dir millorar el dataset.

Consells de programació: Versiona el dataset com si fos codi

En ML, dues execucions amb el mateix codi però dades diferents poden donar models diferents. Desa la versió del dataset, els splits train/valid/test, els hiperparàmetres i el hash del model entrenat. Sense això, no podràs explicar per què el model d'avui no coincideix amb el de la setmana passada.

Lliçó 3: La Responsabilitat és Humana

Un algoritme no té moralitat, de la mateixa manera que una calculadora no en té. Un algoritme no pot ser "racista" ni "just" per voluntat pròpia, perquè no té voluntat, només una funció objectiu. Així, si un sistema d'IA discrimina, comet errors greus o manipula persones usuàries, la responsabilitat no es pot atribuir a la màquina com si tingués voluntat pròpia. Cal revisar totes les decisions en dades, objectius, validació, desplegament, monitoratge i governança.

1. **Dades esbiaixades**: si alimentem el sistema amb dades que reflecteixen desigualtats històriques, el model pot aprendre aquests patrons i reproduir-los com si fossin regularitats objectives.
2. **Objectius mal definits**: si demanem al sistema que maximitzi clics, temps de pantalla o benefici sense restriccions ètiques, pot aprendre estratègies efectives però socialment nocives, com amplificar continguts polaritzadors o explotar vulnerabilitats de les persones usuàries.
3. **Manca de validació**: si despleguem el model en un entorn diferent del d'entrenament sense comprovar-ne la robustesa, el sistema pot fallar per canvi de distribució (*data shift*) o per casos que no estaven ben representats en les dades originals.

4. **Monitoratge insuficient**: encara que el model funcioni bé en el moment del desplegament, el món canvia. Cal controlar degradació del rendiment, deriva de dades, errors per subgrups, ús fora de domini i alertes d'incidents. Un sistema sense monitoratge és un sistema que pot fallar en silenci.
5. **Governança feble**: cal definir qui pot desplegar el model, qui en revisa les mètriques, qui respon davant d'un incident, quan s'ha de retirar o reentrenar i quins mecanismes d'auditoria i apel·lació tenen les persones afectades. Sense governança, la responsabilitat queda diluïda.

El Debat Final: Qui paga els plats trencats?

Per tancar el curs, analitzem el cas d'ús més extrem i complex, on la tecnologia, la llei i l'ètica col·lisionen frontalment.

Escenari: L'Accident del Cotxe Autònom

Un vehicle de Nivell 4 (conducció autònoma completa en certes àrees) circula de nit. Els seus sensors detecten una forma ambigua a la calçada.

- El Lidar diu "Objecte estàtic (bossa de plàstic)".
- La Càmera diu "Incertesa alta (possible vianant ajagut)".
- El Radar diu "No hi ha obstacle metàl·lic".

El mòdul de fusió de sensors, basat en una Xarxa Bayesiana i Deep Learning, calcula una probabilitat de col·lisió del 40%. Per evitar frenades brusques constants ("frenades fantasma"), el llindar de decisió està ajustat al 50%. El cotxe no frena. Resulta ser un vianant que havia caigut. L'atropellament és fatal. **Qui va a la presó? Qui paga la indemnització?** Analitzem els actors:

1. L'Equip d'enginyeria de ML (El Creador de l'Algorisme) L'advocat de l'acusació argumenta que l'equip d'enginyeria va cometre una negligència en fixar el llindar de decisió al 50% i no ser més conservador.

- *Defensa:* L'equip d'enginyeria va seguir les pràctiques estàndard de la indústria. Si el llindar fos del 10%, el cotxe seria inusable (s'aturaria davant de cada fulla que vola). L'equip d'enginyeria va optimitzar la mètrica F1-Score globalment. És just demanar responsabilitat penal per una decisió estadística?

2. L'Equip d'enginyeria de Dades (El Curador del Dataset) La investigació revela que el dataset d'entrenament contenia 10 milions d'imatges, però només el 0.01% eren de "persones ajagudes a terra de nit". El model va fallar per manca d'exemples (Incertesa Epistèmica).

- *Defensa:* És impossible recollir dades de totes les situacions anòmales possibles (*Long Tail*). L'empresa va complir amb la normativa de dades vigent. És responsable l'equip d'enginyeria de dades per no haver previst l'imprevisible?

3. L'Empresa (El Responsable del Producte) L'empresa va vendre el producte com a "Pilot Autònom", generant una falsa sensació de seguretat en el conductor humà, que estava mirant el mòbil i no va reaccionar.

- *Responsabilitat:* Aquí entra el concepte de **Responsabilitat Objectiva** (Strict Liability). En productes d'alt risc, el fabricant és responsable dels danys encara que no hi hagi "culpa" o negligència directa, simplement perquè es beneficia de l'activitat de risc.

La Solució de l'Enginyeria: ODD i Monitoratge

Com a professionals d'enginyeria, no podem resoldre individualment el debat legal, però sí podem dissenyar sistemes que redueixin riscos. La resposta no és aspirar a models perfectes, sinó definir clarament el seu **ODD** (*Operational Design Domain*): en quines condicions el sistema pot operar i en quines s'ha d'abstenir o transferir la decisió. Un sistema hauria d'incloure:

1. **Domini operacional explícit:** condicions d'ús, tipus de dades acceptades, població prevista i escenaris fora d'abast.
2. **Estimació d'incertesa:** detecció de casos desconeguts, fora de distribució o massa ambigus per actuar automàticament.
3. **Fallada segura:** reduir risc quan el sistema no sap prou, i no forçar predicció.
4. **Registre d'auditoria:** conservar evidències de dades d'entrada, versió del model, predicció, confiança i variables explicatives principals.
5. **Human-in-the-loop:** definir quan cal revisió humana i quin nivell d'autoritat té.
6. **Criteris de retirada o reentrenament:** establir llindars de degradació, *drift* o error que obliguin a aturar o actualitzar el model.

Consells de programació: Defineix l'ODD abans de parlar de responsabilitat

En sistemes autònoms, la pregunta clau no és només "qui té la culpa?", sinó si el sistema estava operant dins del seu *Operational Design Domain* (ODD): condicions de llum, clima, velocitat, tipus de via i escenaris previstos. Si surt de l'ODD, el sistema hauria d'entrar en mode segur, demanar intervenció o aturar-se. Sense ODD, monitoratge i registre d'auditoria, la responsabilitat tècnica queda mal definida.

Estem entrant en una era on l'IA serà una *commodity*. Les eines d'IA generativa poden accelerar moltes tasques: escriure fragments de codi, generar notebooks inicials, proposar tests, documentar funcions o explorar alternatives. Això no elimina el paper de l'enginyeria humana; el desplaça cap a tasques de més responsabilitat: formular bé el problema, validar els resultats, detectar errors subtils, auditar riscos, integrar el sistema i decidir si és segur desplegar-lo.

Per tant, el valor de l'enginyeria humana es desplaçarà cap a:

- **Disseny de Sistemes:** Com connectar models, bases de dades i interfícies de manera robusta i escalable.
- **Auditoria i Ètica:** Ser capaç de certificar que un model és just, segur i compleix la normativa (com la *EU AI Act*).
- **Sostenibilitat (Green AI):** Entrenar un model massiu (LLM) consumeix l'energia d'una ciutat petita. L'enginyeria del futur haurà d'optimitzar precisió i eficiència energètica (models petits, quantització, *edge computing*).

Tancament docent

Problemes de consolidació.

1. Donada una llista de models treballats al llarg del curs —KNN, regressió lineal, regressió logística, SVM, arbres, Naive Bayes, K-Means, GMM, Random Forest i HMM—, situar-los en un espectre segons interpretabilitat, dependència de dades,

Tancament docent *(continuació)*

flexibilitat, risc d'overfitting i cost computacional.

2. Analitzar un pipeline amb una accuracy gairebé perfecta des de la primera iteració. Identificar possible *data leakage*, variables proxy, contaminació del test o decisions de validació que podrien explicar el resultat.
3. Donat un model que utilitza variables sensibles o proxies indirectes, proposar una auditoria mínima: variables a revisar, mètriques desagregades, proves de robustesa, explicacions locals i mecanisme de revisió humana.
4. Comparar dues opcions de desplegament: un model molt precís però difícil d'explicar i un model lleugerament menys precís però auditable i interpretable. Justificar quina opció seria preferible en un context mèdic, financer o administratiu.
5. Dissenyar un cicle MLOps mínim per a un model que ja ha estat entrenat: ingesta de dades, validació, desplegament, monitoratge, detecció de *data drift*, reentrenament, registre de versions i criteris d'aturada.
6. Avaluar un cas de Green AI: un model nou millora un 1% la mètrica principal però multiplica per deu el cost energètic d'entrenament i inferència. Discutir si el desplegament és justificable i sota quines condicions.
7. Analitzar l'ús d'una eina d'IA generativa per escriure codi de modelatge. Identificar quines parts pot accelerar, quines no pot substituir i quines responsabilitats continuen sent de l'equip d'enginyeria.

Criteris breus de resolució. Una bona resolució ha de mostrar una visió sistèmica de l'aprenentatge automàtic. No n'hi ha prou amb triar el model amb millor mètrica: cal justificar la qualitat de les dades, el protocol de validació, l'absència de *leakage*, la interpretabilitat, la robustesa, el monitoratge i les conseqüències del desplegament. En els problemes ètics, cal identificar quina decisió tècnica genera el risc i quina mesura d'enginyeria podria reduir-lo. En MLOps, cal entendre que un model en producció és un sistema viu, exposat a canvi de distribució, errors de dades i responsabilitat operativa. Cal emetre una decisió professional justificada: desplegar, revisar, limitar el domini d'ús o aturar, indicant quines evidències sostenen aquesta decisió.

Connexió amb pràctica/laboratori. L'alumnat pot partir d'un notebook aparentment correcte i revisar-lo com si fos un sistema candidat a producció: comprovar particions de dades, buscar leakage, justificar mètriques, afegir una explicació del model, redactar una fitxa de riscos, proposar monitoratge, definir condicions de reentrenament i emetre una decisió final de desplegament, revisió o aturada.

La intel·ligència artificial i l'aprenentatge automàtic són eines poderoses. Poden millorar diagnòstics, optimitzar recursos i ampliar capacitats humanes; però també poden amplificar biaixos, errors i desigualtats si es dissenyen sense criteri. Així, no sigueu només persones que entrenen models. Sigueu professionals d'enginyeria que entenen les dades, validen les hipòtesis, documenten els límits i assumeixen les conseqüències dels sistemes que construeixen.

Fem possible l'enginyeria responsable de la Intel·ligència Artificial.

Fi del Curs.

Bibliografia i Referències Recomanades

Aquesta bibliografia està organitzada temàticament segons els capítols del llibre. S'inclouen tant els articles acadèmics fundacionals (per als qui vulguin anar a la font original) com els manuals moderns de referència per aprofundir en la matèria.

General: Llibres de Capçalera

Aquests textos cobreixen transversalment tant l'assignatura de Computació i Representació de la Informació (CRI) com la d'Aprenentatge Computacional (APC).

- **Russell, S., & Norvig, P.** (2020). *Artificial Intelligence: A Modern Approach* (4th Edition). Pearson. *(La "Bíblia" de la IA. Imprescindible per a la part de lògica, cerca i probabilitat dels primers capítols).*
- **Bishop, C. M.** (2006). *Pattern Recognition and Machine Learning*. Springer. *(El text estàndard per a la visió probabilística i bayesiana del Machine Learning. Rigor matemàtic extrem).*
- **Hastie, T., Tibshirani, R., & Friedman, J.** (2009). *The Elements of Statistical Learning*. Springer. *(Matemàticament exigent, però la referència absoluta per a SVM, Arbres, Ensembles i Regularització).*
- **Géron, A.** (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3rd Edition). O'Reilly Media. *(El millor llibre pràctic per implementar el codi Python d'aquest curs. Molt recomanat per a la part d'enginyeria).*

Capítol 1: La Geometria de les Dades (Vectors i KNN)

- **Strang, G.** (2016). *Introduction to Linear Algebra* (5th Edition). Wellesley-Cambridge Press. *(Fonaments d'àlgebra lineal i espais vectorials explicats pel mestre Gilbert Strang).*
- **Cover, T., & Hart, P.** (1967). "Nearest neighbor pattern classification". *IEEE Transactions on Information Theory*, 13(1), 21-27. *(L'article original que demostra les propietats asimptòtiques del k-NN).*
- **Bellman, R.** (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press. *(On es defineix per primera vegada el terme "Maledicció de la Dimensionalitat").*

Capítol 2: Optimització i Regressió

- **Boyd, S., & Vandenberghe, L.** (2004). *Convex Optimization*. Cambridge University Press. *(Referència definitiva per entendre per què algunes funcions de cost tenen un mínim global únic i altres no).*
- **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). *Deep Learning* (Capítol 4: Numerical Computation). MIT Press. *(Excel·lent explicació moderna del Gradient Descent i els problemes d'estabilitat numèrica i condicionament).*
- **Tibshirani, R.** (1996). "Regression Shrinkage and Selection via the Lasso". *Journal of the Royal Statistical Society*, 58(1), 267-288. *(Presentació històrica de la regularització L1 per a selecció de variables).*

Capítol 3: Classificació (Logística i SVM)

- **Cortes, C., & Vapnik, V.** (1995). "Support-vector networks". *Machine Learning*, 20(3), 273-297. *(L'article que va introduir les SVM i el concepte de marge màxim, dominant el camp durant una dècada).*
- **Fawcett, T.** (2006). "An introduction to ROC analysis". *Pattern Recognition Letters*, 27(8), 861-874. *(Guia detallada i didàctica sobre com avaluar classificadors més enllà de l'Accuracy).*
- **Schölkopf, B., & Smola, A. J.** (2002). *Learning with Kernels*. MIT Press. *(Aprofundiment matemàtic en el "Kernel Trick" i els espais de Hilbert).*

Capítol 4: Estructures de Decisió i Bayes

- **Shannon, C. E.** (1948). "A Mathematical Theory of Communication". *Bell System Technical Journal*, 27, 379–423. *(El naixement de la Teoria de la Informació i el concepte d'Entropia).*
- **Quinlan, J. R.** (1986). "Induction of decision trees". *Machine Learning*, 1(1), 81-106. *(Descripció clàssica de l'algorisme ID3 i l'ús del guany d'informació).*
- **Pearl, J.** (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann. *(Judea Pearl, guanyador del Turing, estableix les bases modernes de les Xarxes Bayesianes i la causalitat).*

Capítol 5: Aprenentatge No Supervisat

- **MacQueen, J.** (1967). "Some methods for classification and analysis of multivariate observations". *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability. (Primera aparició formal de l'algorisme K-Means).*
- **Dempster, A. P., Laird, N. M., & Rubin, D. B.** (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society*, 39(1), 1-38. *(L'article fonamental que defineix l'algorisme Expectation-Maximization).*
- **Arthur, D., & Vassilvitskii, S.** (2007). "k-means++: The Advantages of Careful Seeding". *SODA '07 Proceedings. (La millora moderna per a la inicialització de clústers, utilitzada per defecte a scikit-learn).*

Capítol 6: Ensembles i Seqüències

- **Breiman, L.** (2001). "Random Forests". *Machine Learning*, 45(1), 5-32. *(L'article clàssic on es defineix l'algorisme Random Forest i el concepte de Bagging).*
- **Rabiner, L. R.** (1989). "A tutorial on Hidden Markov Models and selected applications in speech recognition". *Proceedings of the IEEE*, 77(2), 257-286. *(El tutorial definitiu sobre HMM i l'algorisme Viterbi, molt clar i aplicat).*
- **Gal, Y., & Ghahramani, Z.** (2016). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". *ICML Proceedings. (El treball seminal sobre com utilitzar xarxes neuronals per estimar la incertesa epistèmica amb MC Dropout).*

Capítol 7: Ètica, MLOps i Enginyeria

- **O'Neil, C.** (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown. *(Lectura obligatòria sobre els biaixos algorítmics i*

l'impacte social de models mal dissenyats).

- **Sculley, D., et al. (Google)** (2015). "Hidden Technical Debt in Machine Learning Systems". *NIPS Proceedings. (El paper que va fundar el camp de l'MLOps: explica per què mantenir codi de ML en producció és difícil i costós).*
- **Lundberg, S. M., & Lee, S. I.** (2017). "A Unified Approach to Interpreting Model Predictions". *NIPS Proceedings. (Presentació dels valors SHAP per a l'explicabilitat de models de caixa negra).*
- **European Commission.** (2024). *The AI Act (Artificial Intelligence Act). (Legislació europea vigent sobre l'ús de sistemes d'alt risc, imprescindible per a l'enginyer modern).*

Eines i Llibreries (Documentació Tècnica)

Per a la part pràctica d'enginyeria, la documentació oficial és sovint millor que qualsevol llibre, ja que s'actualitza contínuament:

- Gebru, T., et al. (2021). *Datasheets for Datasets.* Communications of the ACM. Referència útil per documentar conjunts de dades i limitacions.
- Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). *On Calibration of Modern Neural Networks.* ICML. Referència recomanada per al tema de calibratge.
- Lundberg, S. M., & Lee, S. I. (2017). *A Unified Approach to Interpreting Model Predictions.* NeurIPS. Referència fonamental sobre SHAP.
- Mitchell, M., et al. (2019). *Model Cards for Model Reporting.* FAT*. Referència útil per documentar models abans de desplegar-los.
- **Scikit-learn:** Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research.* `https://scikit-learn.org/`
- **Pandas:** McKinney, W. (2010). "Data Structures for Statistical Computing in Python". `https://pandas.pydata.org/`
- **SHAP (SHapley Additive exPlanations):** `https://shap.readthedocs.io/`

Colofó

Composició: LaTeX
Edició: Grau en Enginyeria Informàtica, UAB
Versió: 26 de maig de 2026