Sistemes recomanadors aplicats a productes de roba.

Sergi Pérez Muro

Resum—Els sistemes recomanadors avui dia són molt presents a internet, de manera que han canviat la forma en què les persones descobreixen i consumeixen nous continguts o productes. Un sistema recomanador és una eina que estableix un conjunt de criteris i valoracions sobre dades d'usuaris per realitzar prediccions sobre les preferències d'aquests i recomanar elements que els puguin ser d'utilitat. En aquest projecte s'exploren les diferents tècniques que ofereix el filtratge col·laboratiu, el que consisteix a utilitzar les dades de compres existents dels usuaris: aquelles basades en la similitud entre items (item-based), les basades en la similitud entre usuaris (user-based), la factorització de matrius i finalment les basades en aprenentatge profund. Aquestes s'apliquen a un problema real de recomanació de productes de roba a partir de les compres dels usuaris en què caldrà determinar quins són els 12 productes que els usuaris més probablement compraran.

Paraules clau— Sistemes recomanadors, filtratge col·laboratiu, content-based, item-based CF, user-based CF, factorització de matrius, Alternating Least Squares, Stochastic Gradient Descent, similitud de cosinus, similitud de jaccard.

Abstract— Recommender systems today are really present on the internet, so they have changed the way people discover and cosume new content or products. A recommender system is a tool that establishes a set of criteria and assessments of user data to make predictions about their preferences and recommend items that may be useful to them. This project explores the different techniques offered by collaborative filtering, which consists in using data about purchases users have made. These tecniques are based in similarities of items (item-based), similarities of users (user-based), matrix factorization and finally based of deep learning architectures. These are applied to a real problem of recommending clothing products based on users purchases in which it will need to determine which 12 products are most likely to buy.

Index Terms— Recommender systems, collaborative filtering, content-based, item-based CF, user-based CF, matrix factorization, Alternating Least Squares, Stochastic Gradient Descent, cosine similarity, jaccard similarity.

1 Introducció

Els sistemes recomanadors actualment es troben presents a la majoria de botigues digitals i plataformes de contingut, i han canviat la manera en què les persones descobreixen i consumeixen nous productes. Existeixen diverses tècniques i algorismes per tractar d'obtenir els millors resultats a l'hora de recomanar productes i l'efectivitat d'aquestes depèn de les tendències de consum dels usuaris i el tipus de problema a resoldre.

La motivació d'aquest treball és descobrir com utilitzar l'aprenentatge per computador per a elaborar i implementar algoritmes que busquin similituds entre usuaris i productes, i que extreguin informació útil sobre aquests per fer prediccions sobre com es comportaran en el futur i a més recomanar aquells productes no descoberts que puguin ser d'interès per als usuaris.

En aquest projecte s'aplicaran sistemes recomanadors basats en filtratge col·laboratiu per recomanar productes de roba, un tipus de productes en què les tendències de compres canvien constantment, aplicant diferents algoritmes basats en les tècniques més utilitzades en l'actualitat, per analitzar quins donen millors resultats en aquest àmbit de compres de roba.

Aquest article descriu els objectius del treball, l'estat de l'art sobre el qual es parteix, la metodologia a seguir, l'elecció i justificació del dataset triat per treballar amb algoritmes recomanadors, el preprocessament que s'ha hagut de fer a les dades seguidament del desenvolupament del treball amb els diferents algorismes implementats, la descripció i anàlisi dels resultats i decisions preses, i les conclusions finals extretes del treball.

2 Objectius

L'objectiu principal del projecte és analitzar el

- *E-mail de contacte: sergi.perezmur@autonoma.cat*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Ramon Baldrich Caselles (Ciències de la computació)*
- *Curs 2021/22*

funcionament i comparar les diferents tècniques i algorismes per elaborar sistemes recomanadors aplicats a la recomanació de productes de roba. Així com trobar quins obtenen millors resultats en el problema aplicar i quina de les tècniques és més adient en aquest cas. En concret els aspectes a assolir són els següents:

- Implementar els algoritmes sense utilitzar llibreries amb implementacions ja fetes, per tal d'aprendre les tècniques més importants per elaborar sistemes recomanadors com a punt de partida per en un futur enfrentar-se a solucions més complexes.

- Obtenir resultats satisfactoris en comparació amb els que altres persones obtenen a la competició de Kaggle tenint en compte que el treball està més orientat a aprendre sistemes recomanadors.

- Adaptar les solucions a un problema que no és el típic: no és un problema de *ratings* (feedback explícit) de recomanació de pel·lícules sinó que cal predir productes més costosos de predir com ara productes de roba.

- Aplicar solucions de visió per computador a les imatges dels productes.

3 ESTAT DE L'ART

L'estat de l'art actual dels sistemes recomanadors és certament complex, i això requereix solucions complexes, que consisteixen en ecosistemes d'algoritmes que treballen conjuntament. No només s'intenta tenir en compte quins productes agradaran als usuaris, sinó que recomanar a nous usuaris (cold-start problem), quin ranking donar a les recomanacions, com aportar diversitat a aquestes recomanacions i factors que poden condicionar els resultats, com ara socials i temporals.

Existeixen dos grans grups de tipus de sistemes recomanadors, els basats en contingut (content-based filtering), que utilitzen les característiques pròpies dels productes per recomanar productes similars als que han comprat els usuaris anteriorment. Aquests tipus de sistemes recomanadors són fàcilment escalables a un nombre gran d'usuaris i poc costosos computacionalment, ja que no necessiten dades sobre altres usuaris, però presenten limitacions perquè no tenen en compte els hàbits de compra dels altres usuaris. Per altra banda, els sistemes recomanadors basats en filtratge col·laboratiu, que utilitzen les dades de compra de tots els usuaris per trobar similituds entre usuaris i productes. També es fan servir sistemes recomanadors híbrids que combinen els dos tipus anteriors.

Els sistemes recomanadors basats en filtratge col·laboratiu són els més utilitzats i tenen un major potencial per obtenir bons resultats. És per això que aquest treball s'orientarà a treballar amb aquest tipus de sistemes recomanadors, encara que no s'hauria de descartar la possibilitat d'implementar un híbrid amb els dos tipus per intentar millorar els resultats.

Molts dels algorismes més eficients de filtratge col·laboratiu són els que es basen en factorització de matrius, amb un desenvolupament que es va accelerar amb el concurs de Netflix que es va dur a terme entre el 2006 i el 2009. Actualment, la majoria de sistemes recomanadors que obtenen millors resultats són sistemes d'aprenentatge profund complexos que també utilitzen la factorització de matrius.

A la competició de Kaggle, les solucions que aporten els millors resultats fan servir Gradient Boosted Trees, amb models Lightgbm i Bayesian Personalizated Rakings amb factorització de matrius entre altres tècniques. Aquestes aconsegueixen fins a un 3.792% de Mean Average Precision (MAP).

4 METODOLOGIA

La metodologia que s'ha seguit ha estat incremental, explorant les principals tècniques i estratègies desde les més bàsiques fins les que més s'utilitzen en l'actualitat i han tingut més éxit en els darrers anys.

En primer lloc s'haurà de seleccionar el dataset amb el que es vol treballar. Aquest ha de contenir principalment les dades de compra dels usuaris per poder aplicar les tècniques de filtratge col·laboratiu, informació útil sobre els usuaris i productes per si fos necessari utilitzar-la, i estaria bé que comptés amb les imatges dels productes per poder elaborar alguna solució basada en visió per computador i trobar productes similars a partir de les imatges.

Els algorismes s'han implementat en codi, sense haver de cridar llibreries, amb una perpectiva que serveixi per aprendre com aquests funcionen matemàticament i després comparar els resultats que s'obtenen amb l'objectiu d'apropar-se el máxim possible a les millors solucions de la competició de Kaggle.

4.1 Eines utilitzades

- **Spyder IDE [2]:** és un entorn de desenvolupament en Python. Utilitzat per implementar els algorismes i la visualització de resultats.

- **Google Colab [3]:** és una eina per escriure i executar codi Python al núvol de Google. Utilitzat per entrenar el model de Deep Learning amb GPU.

- **Pytorch Lightning [4]:** llibreria que proporciona una interfície d'alt nivell per Pytorch. Utilitzat per implementar el model de deep learning.

- **Altres llibreries:** s'ha utilitzat la llibreria Pandas per treballar amb els fitxers csv, scipy per tractar amb matrius esparses i matplotlib per representar els resultats obtinguts.

5 Desenvolupament

5.1 Dades utilitzades

Per treballar amb sistemes recomanadors s'ha triat una competició de la plataforma *Kaggle* (H&M Personalized Fashion Recommendations) [1] que proporciona un dataset sobre les compres de productes de roba amb les que compta la cadena de tendes H&M, en la que l'objectiu és predir les compres dels usuaris en una setmana recomanant 12 productes a cada usuari del dataset.

S'ha seleccionat aquest dataset perquè és prou complet, compta amb més de 31 milions de transaccions que s'hi han produït al llarg de 2 anys per aplicar tècniques de filtratge col·laboratiu, és un problema interessant, ja que no és el típic de recomanació de pel·lícules, i a més compta amb imatges de tots els productes, cosa que permetria aplicar alguna solució basada en visió per computador per trobar similituds entre productes.

Els fitxers csv del dataset que incorpora la competició són els següents.

- **transactions_train.csv**
  Inclou totes les transaccions que s'han realitzat entre el 21/09/2018 i el 21/09/2020. Una transacció consisteix en una compra d'un únic client a un únic producte, amb els següents atributs:

  t_dat: data YYYY/MM/DD en la que s'ha realitzat la transacció.
  customer_id: identificador únic del client que ha realitzat la transacció.
  article_id: identificador únic del producte que s'ha comprat en la transacció.
  price: preu del producte normalitzat.
  sales_channel_id: identificador del canal de venda en el que s'ha produit la transacció.

- **articles.csv**
  Inclou les dades de més de 100.000 productes de roba amb el seu identificador.
- **customers.csv**
  Inclou les dades de més de 1.300.000 usuaris que han comprat algun producte.
- **sample_submission.csv**
  Exemple de com ha de ser el fitxer amb les recomanacions per avaluar els resultats a Kaggle.

5.2 Preprocessament de les dades

Abans de començar a implementar els algoritmes, serà necessari realitzar un processament de les dades. En primer lloc, separar-les en els conjunts d'entrenament i validació, i al treballar amb filtratge col·laboratiu, cal representar les dades de les compres. Es representen en format matricial per utilitzar-les com entrada per elaborar els models i en format de diccionaris per accedir a les dades amb més facilitat.

Separació de dades (training/validation)

La plataforma Kaggle utilitza com a test les transaccions realitzades 1 setmana després de les dades que es proporcionen. Per tal de realitzar una validació similar s'agafa l'última setmana de les transaccions proporcionades com a validació i la resta com a entrenament. Una setmana, de mitja, compta amb 250.000 transaccions.

Generació matriu de compres

Representar les dades de compra entre usuaris i productes en una matriu representa un problema d'espai important, ja que el dataset compta amb un total de 1.371.980 usuaris i un total de 105.542 productes. No obstant, com la gran majoria de valors de la matriu són 0, es pot representar com una matriu esparsa amb *csr_matrix* de la llibreria scipy, solucionant aquest problema d'espai.

Diccionari de compres

També es considera representar les dades de les transaccions en diccionaris.
Un diccionari *customer2article* on les claus són els clients i els valors són diccionaris amb els productes comprats com a claus i el nombre de vegades comprat com a valor.
Un diccionari *article2customer* on les claus són els productes i els valors són diccionaris amb els usuaris que els han comprat com a claus i el nombre de vegades comprat com a valor.

5.3 Item based CF

Els algoritmes basats en filtratge col·laboratiu basat en items consisteixen en trobar similituds entre items, en aquest cas productes, a partir dels hàbits de compra dels clients. En aquest problema es treballarà amb el nombre de vegades que un usuari ha comprat cada producte.

Primera versió

S'ha implementat una primera versió. Per trobar la similitud entre usuaris a partir de la matriu esparsa de compres es calcula la similitud de cosinus.

Similitud de cosinus

La similitud de cosinus troba similituds a partir del cosinus de l'angle θ entre els vectors de dos productes. Quan θ augmenta, el cosinus disminueix fins a arribar a la similitud mínima quan $\theta = 90$, $\cos(90) = 0$, que vol dir que cap usuari ha comprat tots dos productes.
A la figura 1, si dos usuaris compren tots dos productes, el producte 2 (movie_2) s'assembla més al producte 0 que al 3, ja que la proporció entre el nombre de compres és més similar i per tant l'angle més petit [5].

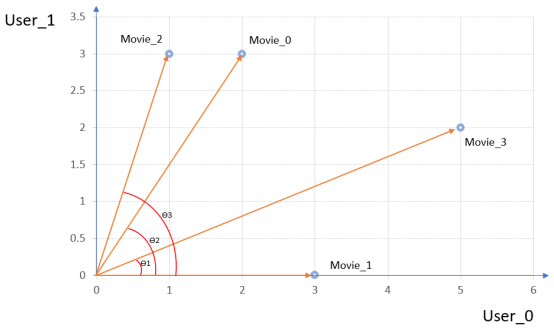

Fig 1. Espai de la similitud de cosinus entre productes

Una vegada obtingudes les similituds entre productes, per generar les recomanacions a cada client, per cada un es recorren els productes que han comprat, s'agafen els més similars i finalment s'ordenen segons la similitud que tenen amb algun dels productes que ha comprat l'usuari. Els 12 primers es recomanen a l'usuari. El següent diagrama de flux, a la figura 2, explica detalladament com es generen les recomanacions.

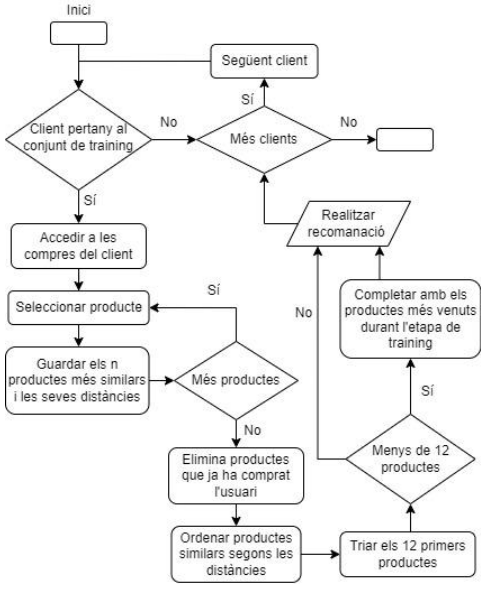

Fig 2. Diagrama de flux de la generació de recomanacions del item-based CF.

Segona versió

S'ha implementat una segona versió, ja que es considera que la primera presenta limitacions, com ara que no penalitza als usuaris que compren massa i que no es calcula la similitud d'un producte amb el conjunt sencer de productes que compra l'usuari, sinó amb productes individuals [6].

En aquesta segona versió es normalitzen els vectors de compra dels usuaris a vectors unitaris, de manera que es penalitza a aquells usuaris que compren molt productes i per tant, la semblança entre aquests serà menor. El que es fa és calcular la magnitud de cada client fent l'arrel quadrada de la suma del nombre de vegades al quadrat que un usuari compra cada producte. Finalment, el nou vector de compres de l'usuari consistirà en els valors que tenia originalment dividits per la magnitud.

$$magnitude = \sqrt{(x^2 + y^2 + z^2 + \ldots)}$$

$$vector = \left(\frac{x}{magnitude}, \frac{y}{magnitude}, \frac{z}{magnitude}, \ldots\right)$$

Amb els nous vectors dels usuaris es calcula de la mateixa manera la similitud de cosinus.
A l'hora de realitzar les recomanacions, en aquest cas no es tindran en compte les similituds entre dos productes concrets, sinó que es recomanaran els productes més similars al conjunt total de les compres de l'usuari fent una ponderació.
Pels productes més similars a cada un dels productes que s'han comprat, es calcula una score *S*. Per trobar l'score entre un usuari u i un producte i, es fa un sumatori dels valors del vector de similituds d'i multiplicats pel nombre de vegades que l'usuari u ha comprat cada producte j. Aquest sumatori es divideix entre el sumatori dels valors del vector de similituds d'i.

$$S(u, i) = \frac{\sum_{j \epsilon N} W_{ij} r_{uj}}{\sum_j |W_{ij}|}$$

Els 12 productes que obtenen millor score són els que es recomanen al client.

5.4 User based CF

Els algoritmes basats en filtratge col·laboratiu basat en usuaris consisteixen a trobar similituds entre usuaris a partir dels seus hàbits de compra.
Per determinar la similitud entre clients, caldrà determinar com s'assembla el conjunt de productes que ha comprat un client als conjunts de productes d'altres clients. Per tant, en aquest cas es considera que la matriu de compres ha de ser binària, ja que no interessa el nombre de vegades que els clients han comprat cada producte sinó el conjunt total de productes comprats. Per aquest motiu, el més adient és utilitzar la similitud de Jaccard per mesurar com s'assemblen els usuaris.

Es descarta fer servir la correlació de Pearson, ja que en aquest cas no es tracta d'un problema de regressió.

Similitud de Jaccard

Es vol calcular la similitud entre dos usuaris a partir de del conjunt de compres *A* del primer client i el conjunt de compres *B* del segon client [7].
La similitud de Jaccard equival a la intersecció dels dos conjunts, és a dir, el nombre de productes en comú a tots dos conjunts, entre la unió dels dos conjunts, és a dir, el nombre de productes diferents als dos conjunts, que es pot simplificar com el nombre de productes al conjunt A més el nombre de productes al conjunt B menys la intersecció entre els dos conjunts.

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Cal destacar que no és útil tenir en compte aquells casos en els que els productes del conjunt B són els mateixos que els de la intersecció entre A i B, ja que llavors no hi haurà productes que recomanar al usuari. La similitud de Jaccard s'ha implementat en codi a partir de la matriu esparsa de compres, sense haver de cridar cap llibreria.

Generació de recomanacions

Amb la matriu esparsa de similituds entre usuaris que s'obté, cal obtenir per cada client aquells als que són més similars i realitzar les recomanacions.
El següent diagrama de flux, a la figura 3, explica com es realitzen les recomanacions: per cada usuari s'obtenen les seves compres, es recorren els usuaris més similars i es resta el conjunt de compres de l'usuari al conjunt de compres dels usuaris similars. El que s'obté són les recomanacions finals que es completen amb els productes més venuts fins arribar a 12.

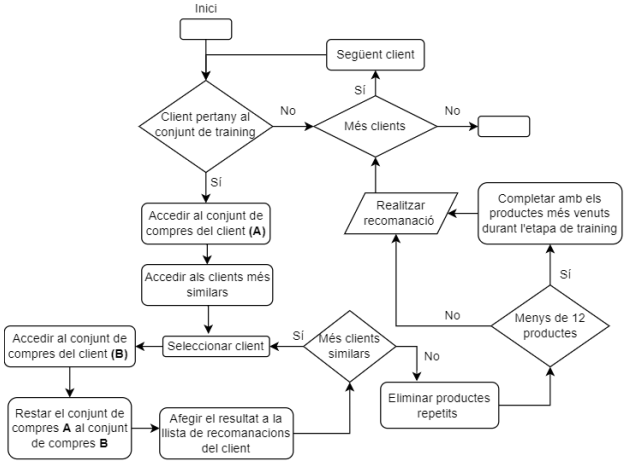

Fig 3. Diagrama de flux de la generació de recomanacions del user-based CF.

5.5 Factorització de matrius

L'objectiu dels algoritmes basats en factorització de matrius és descompondre la matriu de compres en un producte de dos matrius amb menor dimensionalitat.

La matriu de compres R té unes dimensions NxM on N és el nombre de clients i M és el nombre de productes, i aquesta s'expressa com el producte d'una matriu P de mida NxK i una matriu Q transposada de mida KxM on K són les característiques latents que millor descriuen als clients, a la matriu P, i als productes, a la matriu Q.

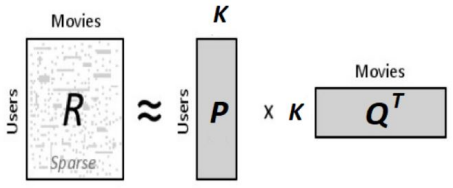

Fig 4. Expressió de la factorització de la matriu de compres

Per tal que el producte de les dues matrius P i Q transposada sigui igual a la matriu R, el nombre de característiques K hauria de ser igual al nombre de files o columnes linealment independents de R. L'objectiu de la factorització de matrius també és reduir el nombre de característiques, utilitzant aquelles que millor descriuen la matriu original, aplicant una reducció de la dimensionalitat, obtenint una aproximació $\widehat{R}$ de la matriu de compres original.

$$R \approx P \, x \, Q^T = \widehat{R}$$

Per predir el nombre de compres d'un client '*i*' a un producte '*j*' es multiplicarà el vector de la matriu P corresponent al client pel vector de la matriu Q corresponent al producte transposada.

$$\widehat{r}_{ij} = p_i q_j^T = \sum_{k=1}^{k} p_{ik} q_{jk}$$

Una de les tècniques més populars per aconseguir reduccions de dimensionalitat és el Singular Value Descomposition (SVD). No obstant, el problema que presenta és que no es pot fer amb una matriu de compres que li faltin dades, i emplenar les dades la matriu suposa un problema important d'espai.

És per això, que per resoldre el problema amb factorització de matrius, s'utilitzaran algorismes iteratius que redueixin a cada iteració l'error respecte la matriu de compres original. Aquests són el Stochastic Descent Gradient (SDG) i l'Alternating Least Squares (ALS).

5.5.1 Stochastic Gradient Descent (SGD)

El descens de gradient és un dels algoritmes basat en factorització de matrius més utilitzats [8].
Per obtenir les matrius P i Q, s'inicialitzen les dues matrius aleatòriament, es calcula l'error entre la matriu R i l'aproximació de R amb la diferència del producte.

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{k} p_{ik} q_{jk})^2$$

L'objectiu és minimitzar aquest error, per fer-ho s'obté el gradient derivant l'error en funció de la matriu P i la matriu Q.

$$\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \widehat{r}_{ij})(q_{jk}) = -2 e_{ij} q_{jk}$$

$$\frac{\partial}{\partial q_{jk}} e_{ij}^2 = -2(r_{ij} - \widehat{r}_{ij})(p_{ik}) = -2 e_{ij} p_{ik}$$

Aquest gradient s'utilitza per recalcular els valors de les matrius P i Q iterativament. Les dues matrius s'actualitzen a cada punt sumant el valor que tenen actualment al gradient multiplicat per un valor α, que és

el learning rate, que indicarà com de ràpid l'algorisme ha d'aprendre per minimitzar l'error.

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial q_{jk}} e_{ij}^2 = p_{ik} + 2\alpha\, e_{ij}\, q_{jk}$$

$$q'_{jk} = q_{jk} + \alpha \frac{\partial}{\partial q_{jk}} e_{ij}^2 = q_{jk} + 2\alpha\, e_{ij}\, p_{ik}$$

A continuació s'implementa el SGD en el codi sense haver de cridar llibreries que tinguin la implementació.
Al final de cada iteració s'ha de calcular el Loss amb les dades de training i amb les dades de validació

5.5.2 Alternating Least Squares (ALS)

El Alternating Least Squares és un altre algorisme iteratiu basat en factorització de matrius.
Una vegada s'inicialitzen les matrius P i Q, la matriu P s'actualitza a partir de la matriu Q i després la matriu Q s'actualitza a partir de la matriu P. Això consisteix en una iteració de l'algorisme i és el mateix quina matriu s'actualitza primer.

Per actualitzar una matriu en funció de l'altre, es parteix de què la matriu de compres és igual a la seva aproximació, que és la matriu P multiplicada per la matriu Q transposada.

$$\sum_j (q_j^T p_i)\, q_j = \sum_j r_{ij}\, q_j$$

Es pot aïllar el valor de la matriu P, ja que no canvia dins del sumatori, obtenint la següent igualtat en què s'actualitza la matriu P a partir de les dades de la matriu Q i la matriu de compres original R amb l'objectiu de minimitzar l'error amb l'aproximació de R.

$$p_i = \left(\sum_j q_j q_j^T \right)^{-1} \sum_j r_{ij}\, q_j$$

Es pot fer de la mateixa manera en aquest cas per actualitzar la matriu Q.

$$q_j = \left(\sum_i p_i p_i^T \right)^{-1} \sum_i r_{ji}\, q_i$$

Seguidament, s'ha implementat el ALS en codi, sense cridar llibreries que comptin amb la implementació del algorisme.

5.6 Deep Learning

L'èxit que ha obtingut en els darrers anys el deep learning també ha beneficiat els sistemes recomanadors, sent una de les tècniques més utilitzades juntament amb la factorització de matrius.
Per implementar un sistema recomanador basat en deep leaning s'utilitza la llibreria Pytorch Lightning per elaborar el model i la llibreria TensorBoard per visualitzar els resultats [9].
En primer lloc, caldrà binaritzar la matriu de compres, convertint-se en un problema de classificació en el que es passaran dues entrades al model: l'índex corresponent a l'usuari i l'índex corresponent al producte, i es compara la sortida amb una etiqueta amb valor 0 o 1 en funció si l'usuari ha comprat o no el producte.

Es passen totes les transaccions del conjunt de training al model, però també serà necessari introduir exemples de transaccions que no s'hi hagin produït. Per generar aquestes transaccions negatives, per cada transacció positiva s'agafen 4 productes aleatoris que l'usuari no hagi comprat assignant-les una etiqueta amb valor 0. Totes aquestes transaccions es pasaran a cada epoch, tornant a carregar les dades per què els productes aleatoris de les transaccions negatives no siguin sempre els mateixos i s'obtingui un model molt esbiaixat.

5.6.1 Neural Collaborative Filtering (NCF)

S'utilitza una arquitectura basada en deep learning per a sistemes recomanadors, aquesta consisteix en capes d'embedding per usuaris i productes, capes fully-connected i una capa final que decideix amb quina probabilitat el usuari comprarà el producte.

Arquitectura del model

Les dues entrades que rep el model són els vectors one-hot codificats dels usuaris i dels productes, que aquests es passen per unes capes que obtindran els embeddings.
Els embeddings consisteixen en espais de menor dimensionalitat en els que es troben representats els usuaris i els productes, de manera que els que s'assemblen més es troben a una menor distància. La representació dels embeddings s'obté a partir de filtratge col·laboratiu utilitzant les transaccions.
Seguidament, els embeddings de usuari i producte es concatenen i es pasen a les capes fully-connected que mapejen les entrades fins a la predicció final. Finalment, s'aplica la funció sigmoide per obtenir la probabilitat de què l'usuari compri el producte.

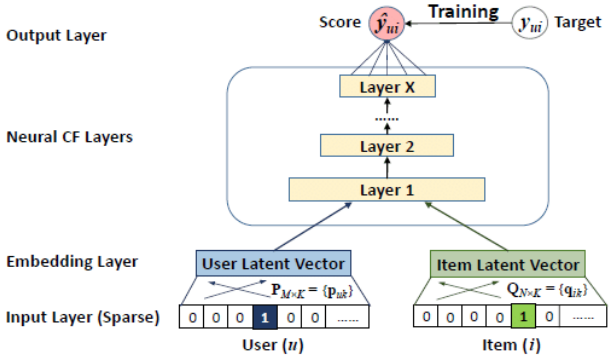

Fig 5. Arquitectura del model Neural Collaborative Filtering

Per avaluar els resultats, es realitzaran canvis en l'arquitectura com ara modificar la dimensió dels embeddings o afegir més capes fully-connected. S'utilitzarà l'eina TensorBoard per mostrar la evolució del Loss de training i validació.

6 Resultats

Com s'ha comentat anteriorment, s'agafa l'última setmana de transaccions com a conjunt de validació amb 250.000 transaccions i la resta com a entrenament. Els millors resultats que els participants aconsegueixen a la competició són d'un 3.7% de MAP, i els que arriben a la posició 200 aconsegueixen un 2.4% de MAP.

6.1 Item-based CF

Primera versió

Utilitzant totes les transaccions del conjunt de training per crear el model de similitud entre items i productes s'obté un Mean Average Precision (MAP) d'un 0.231% d'encert. Les tendències de compra en qualsevol àmbit solen canviar al llarg del temps, i més en aquest àmbit amb compres de roba. És per això que caldria provar de reduir el conjunt de training per obtenir un model que s'ajusti més amb les tendències de la setmana que cal predir.

Reduint el conjunt de training fins a un mes, el MAP augmenta un 300% fins a un 0.65%. A la següent gràfica, a la figura 6, s'observa com milloren els resultats a partir dels dies de training que s'agafen, obtenint el major MAP de 0.9097% agafant l'últim dia de training. No obstant, aquest augment també és degut al fet que agafant pocs dies de training més usuaris cauen fora del model i a aquests se'ls hi recomana productes que prèviament ja havien comprat.

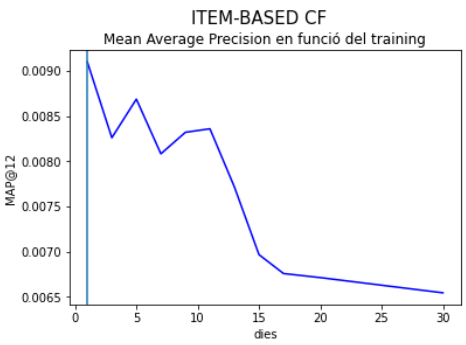

Fig 6. MAP del item-based en funció del nombre de dies de training

Per tant, com en aquest problema els usuaris poden tornar a comprar els mateixos productes a l'etapa de validació i es tenen indicis de què això pot millorar els resultats, es considera recomanar als usuaris com a base aquells productes que prèviament ja han comprat juntament amb les recomanacions personalitzades. En aquest cas, com es pot observar a la figura 7, el MAP millora considerablement amb un màxim agafant els últims 11 dies de training d'un 2.098%.

S'observa també un comportament extrany en agafar més de 15 dies, on el MAP para de baixar i comença a fer-ho més progressivament, similar al comportament de la gràfica anterior. Es podria dir que en agafar més de dues setmanes de training hi ha un punt d'inflexiónhj en els hàbits de compra dels usuaris.

Per tant, per avaluar els resultats i comparar-los es considera utilitzar per una banda l'últim dia de training recomanant nous productes, i utilitzar 11 dies de training recomanant també aquells productes que previament han comprat els usuaris, que és la manera d'obtenir els millors resultats.

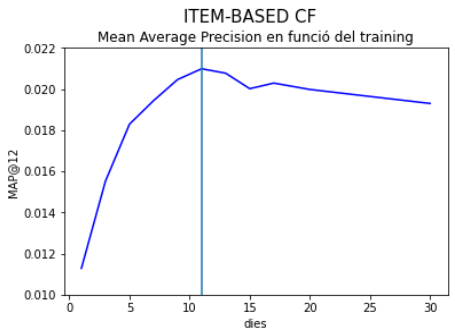

Fig 7. MAP del item-based en funció del nombre dies de training recomananant productes ja comprats.

Segona versió

Executant amb l'últim dia de training recomanant productes no comprats s'obté un MAP de 0.9046%.

Es torna a executar recomanant també els productes que els usuaris han comprat, amb diferents dies de training i la corba és bastant similar a la primera versió, obtenint el pic també als 11 dies de training, però en aquest cas els resultats no superen la primera versió, obtenint un MAP del 2%. En aquest cas es pot considerar que és millor obtenir productes similars a productes individuals i no similars al conjunt total de productes.

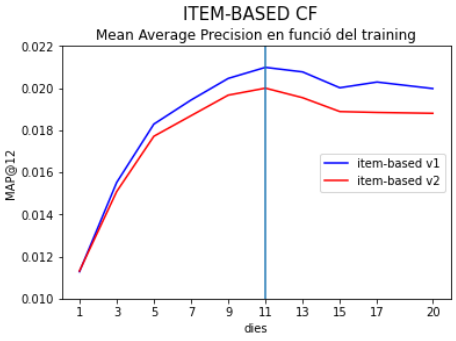

Fig 8. MAP de les dues versions del item-based en funció del nombre de dies de training.

6.2 User-based CF

Aplicant la tècnica de similitud entre usuaris (user-based), s'obté una corba bastant similar a les dues versions del item-based, amb un pic també agafant els últims 11 dies de training, i en aquest cas s'obtenen els millors resultats amb un MAP de 2.13%. Es pot concloure que en aquest problema és millor trobar similituds entre els usuaris que entre els productes, sempre agafant els últims 11 dies abans de l'etapa que cal predir.

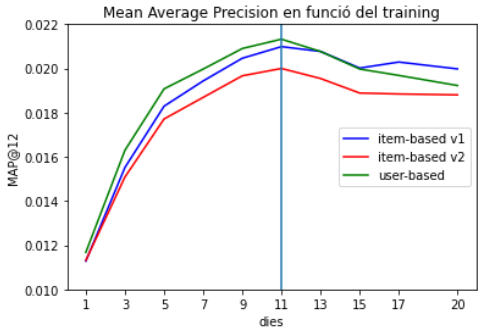

Fig 9. MAP dels item-based i user-based en funció del nombre de dies de training.

6.3 Factorització de matrius

Per avaluar els resultats dels algoritmes basats en factorització de matrius caldrà ajustar els diferents hiperparàmetres i observar la evolució dels Loss de training i validació. Com a Loss s'utilitza el Mean Squared Error (MSE) al treballar amb un problema de regressió. El paràmetre K és el nombre de característiques latents per representar a usuaris i productes.

6.3.1 Stochastic Gradient Descent (SGD)

S'executa el SGD amb una K=100 amb diferents learning rates i s'observa que el que aprén més ràpid i millor és amb un valor de 0.001. No obstant, en tots els casos es produeix overfitting ja que el Loss de validació, representat amb línies discontinues, baixa molt poc fins que es manté constant. Serà necessari ajustar el paràmetre de regularització per sortir del mínim local.

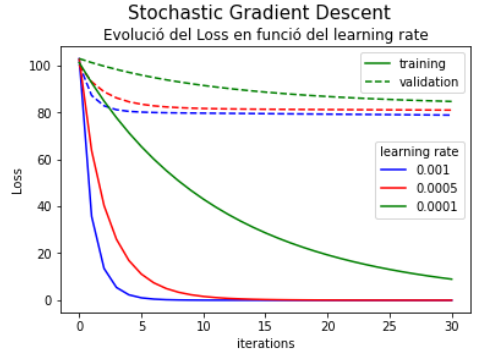

Fig 10. Evolució del Loss de training i validació en funció del learning rate aplicant SDG.

A mesura que s'augmenta el valor de regularització el Loss de validació baixa més ràpid i és capaç d'arribar a un mínim més global. Triant un valor de regularització de 10 amb diferents learning rates el Loss de validació aconsegueix baixar fins a 1.63. El valor 0.0001 de learning rate es descarta perquè triga molt aprendre, mentre que els altres dos aconsegueixen arribar al mínim de 1.63 però amb un learning rate de 0.001 aprèn més ràpid.

No es tria un valor de regularització superior a 10 perquè el Loss de training després d'arribar a un mínim comença a pujar i no s'aprenen tan bé les dades de validació. D'aquesta manera, finalment es tria un learning rate de 0.001 i un valor de regularització de 10.

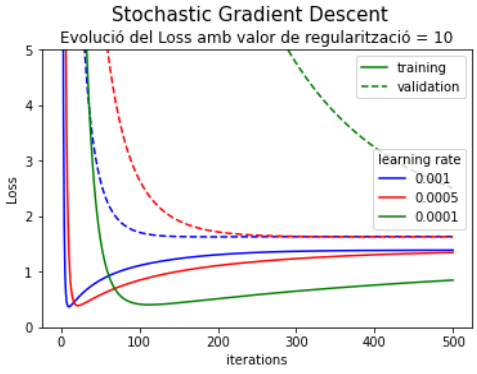

Fig 11. Evolució del Loss de training i validació amb un valor de regularització de 10.

Amb els hiperparàmetres ben ajustats s'executa el SGD amb 1 dia i 11 dies de training amb diferents valors de K fins que el Loss de validació es manté estable. En la taula 1 s'observa que el nombre de característiques K no influeix gaire en el MAP, segurament perquè no aprèn massa, amb valors finals de Loss de validació al voltant de 1.65. Obté resultats similars al item-based collaborative filtering, amb un MAP màxim del 2.08%.

K	MAP (1 dia)	MAP (11 dies)
50	0.8960%	2.0795%
100	0.8972%	2.0797%
200	0.8961%	2.08%
300	0.8962%	**2.0802%**

Taula 1. MAP resultant d'aplicar el SDG en funció del nombre de característiques latents K.

6.3.2 Alternating Least Squares (ALS)

A diferència del SGD, l'Alternating Least Squares no compta amb cap hiperparámetre més enllà del nombre de característiques latents K.

Amb una K=100, s'observa que els Loss de training i validació decreixen bastant més ràpid que al SGD, fins que el Loss de validació s'estabilitza fins a 0.69.

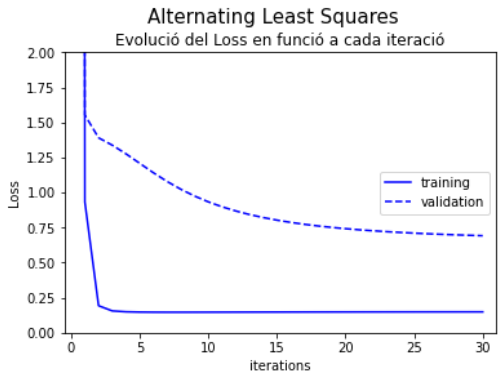

Fig 12. Evolució del Loss de training i validació aplicant ALS.

S'executa el ALS amb 1 dia i 11 dies de training amb diferent nombre de característiques latents K fins que el Loss de validació es manté estable. En aquest cas sí que s'observa, en la taula 2, que la K influeix en el MAP, obtenint millors resultats quan és més gran, excepte amb K=300 que es manté en 2.16%.
Els resultats que s'obtenen són els millors fins ara, tant agafant 1 dia o 11 dies de training, amb un MAP de gairebé un 2.17%

K	MAP (1 dia)	MAP (11 dies)
50	0.9103%	2.1280%
100	0.9168%	2.1487%
200	0.9179%	**2.1685%**
300	0.9192%	2.1632%

Taula 2. MAP resultant d'aplicar el ALS en funció del nombre de característiques latents K

6.4 Deep Learning

Per avaluar els resultats del model de la xarxa NCF s'utilitzarà com a Loss el Binary Cross Entropy (BCE), ja que al binaritzar les dades estem en un problema de classificació.

S'entrena el model amb un embedding de 8 dimensions, agafant només 1 dia de training. S'observa que el Loss de training, en taronja, i de validació, en blau, baixen amb un comportament gairebé idèntic fins al epoch 250 on comença a haver-hi un petit overfitting. El Loss de validació arriba al seu valor mínim de 0.44 a la epoch 400, després puja produint-se més overfitting.

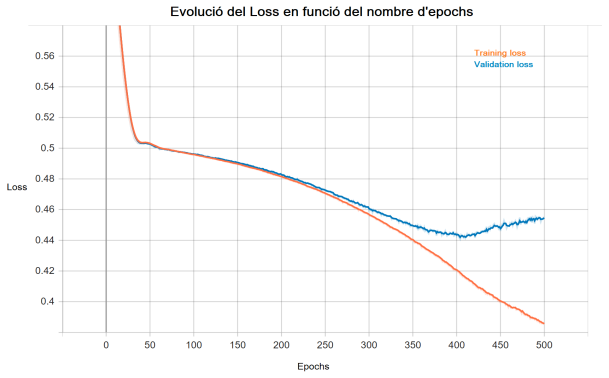

Fig 13. Evolució del Loss de training i validació del model NCF.

A continuació es modifica l'arquitectura de la xarxa NCF. Es prova amb una dimensió dels embeddings més gran, de 16 i 32, i encara que els resultats dels Loss són similars, el model aprèn més ràpid. Finalment s'afegeix una capa fully-connected més amb un embedding de 36 dimensions i s'observa que encara aprèn més rapid. En aquest cas arriba a un Loss de validació mínim lleugerament menor a 0.44 al epoch 250.
Per realitzar les recomanacions, s'executa l'algorisme que fins ara ha obtingut millors resultats, el ALS, i amb els primers 100 productes que obté per cada usuari, s'entren al model i els 12 que millor probabilitat obtenen es recomanen a l'usuari. Amb 1 dia de training s'obté un MAP de 0.9067%.

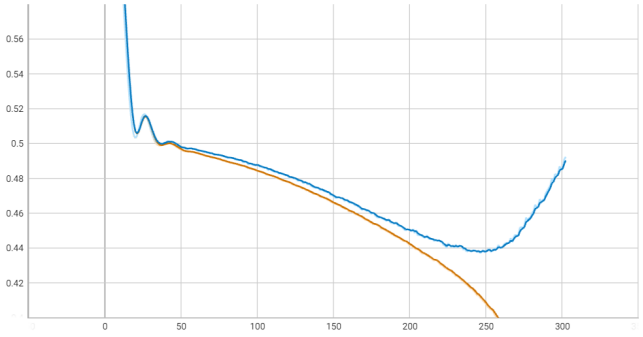

Fig 14. Evolució del Loss de training i validació amb l'arquitectura modificada.

Amb l'arquitectura modificada s'entrena el model ara amb els 11 dies de training. Es para a l'execució a la epoch 270 ja que el model comença a tenir overfitting i el Loss de validació comença a estabilitzar-se amb un valor de 0.37.

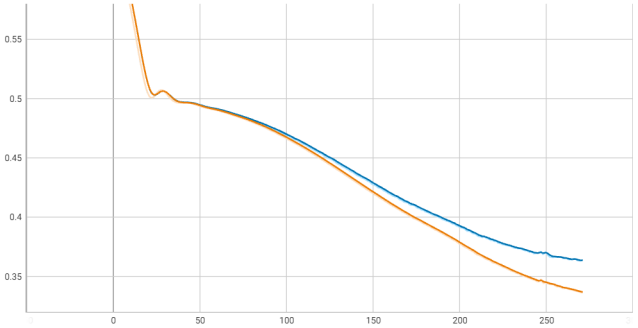

Fig 15. Evolució del Loss de training i validació agafant 11 dies de training.

Una vegada entrenat el model es generen les recomanacions agafant els 100 productes per usuari que tria el ALS i s'obté un MAP de 2.087%. Els resultats són millorables en comparació amb altres algoritmes, possiblement perquè l'arquitectura del model no és prou avançada.

6.5 Resultats finals

Finalment, a la següent taula es mostren els millors resultats obtinguts per tots els algoritmes amb 1 dia i 11 dies de training. L'Alternating Least Squares és l'algorisme que millors resultats obté amb un MAP de gairebé 2.17% seguidament del user-based collaborative filtering, amb un MAP de 2.13%. Les millors solucions de la competició obtenen fins a un MAP de 3.7%, utilitzant tècniques bastant més avançades.

D'altre banda, les modificacions realitzades al item-based no han obtingut la millora que s'esperava i el model NCF podria haver obtingut millors resultats, encara que l'arquitectura que s'utilitza és de les més senzilles.

	MAP (1 dia)	MAP (11 dies)
Item-based CF	0.9097%	2.098%
Item-based CF v2	0.9046%	2%
User-based CF	0.912%	2.13%
Stochastic Gradient Descent	0.8972%	2.0802%
Alternating Least Squares	0.9192%	2.1685%
Neural CF	0.9067%	2.087%

Taula 3. MAP finals obtingut amb els diferents algoritmes agafant 1 dia i 11 dies de training.

7 CONCLUSIONS

Una vegada implementats tots els algoritmes s'obtenen uns resultats força acceptables en comparació als que presenten les millors solucions amb tècniques molt més avançades, que obtenen fins a un Mean Average Precision de 3.7%, mentre que en aquest treball tots els algorismes s'han implementat sense haver de cridar llibreries relacionades amb sistemes recomanadors. Encara que també hi ha hagut algunes solucions de les quals s'esperaven aconseguir millors resultats.

En definitiva, aquest treball ha estat útil per aprendre com funcionen els sistemes recomanadors, des de les tècniques més bàsiques fins a les que s'utilitzen avui dia a les grans companyies, i adaptar-les a un tipus de problema que no utilitza feedback explícit, és a dir valoracions, sinó que es basa en les compres dels clients, en un àmbit que no és tan predictible com el de recomanació de pel·lícules. A més, aquest treball pot ser un bon punt d'entrada per adentrar-se en algoritmes més avançats de recomanació, com ara les Restricted Boltzmann Machines o el Bayesian Ranking, que ha estat utilitzat en algunes de les solucions de la competició.

Com a treball futur, encara hi ha molta varietat de tècniques i estratègies per millorar els resultats en aquest problema. També es podria implementar una solució utilitzant visió per computador, que s'ha elaborat en una altra assignatura per classificar productes de roba, però que per falta de temps no s'ha pogut integrar amb aquest treball. De totes maneres, no és senzill millorar els resultats només a partir d'analitzar les imatges.

BIBLIOGRAFIA

[1] Kaggle, "H&M Personalized Fashion Recommendations", https://www.kaggle.com/c/h-and-m-personalized-fashion-recommendations

[2] Spyder IDE, https://www.spyder-ide.org/

[3] Google Colab, https://colab.research.google.com/

[4] Pytorch Lightning, https://www.pytorchlightning.ai/

[5] Yohan Jeong, "Item-Based Collaborative Filtering in Python", https://towardsdatascience.com/item-based-collaborative-filtering-in-python-91f747200fab

[6] Victor, "Item-item collaborative filtering with binary or unary data", https://medium.com/radon-dev/item-item-collaborative-filtering-with-binary-or-unary-data-e8f0b465b2c3

[7] Compute all pairwise vector similarities within a sparse matrix (Python), (2015). http://na-o-ys.github.io/others/2015-11-07-sparse-vector-similarities.html

[8] Denise Chen, Recommender System — Matrix Factorization (2020).https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b

[9] Towards Data Science, "Deep Learning based Recommender Systems",https://towardsdatascience.com/deep-learning-based-recommender-systems-3d120201db7