HERRAMIENTAS DE CLASIFICACIÓN AUTOMÁTICA DE REGIONES

Marcos Molero Carrión

Resumen— El cáncer de colon es uno de los cánceres con más incidencia en la población mundial. Su detección precoz necesita de la identificación temprana de pólipos, que son su lesión precursora. Sistemas automáticos son empleados para ayudar a los médicos en su detección precoz. Dicha detección automática presenta algunos problemas derivados de la aparición de reflejos especulares que afectan a la salida de dichos métodos. Una solución para este problema es la detección automática de reflejos, que se suele llevar a cabo en dos etapas: una segmentación previa de la imagen y una posterior clasificación de esas regiones a partir de un análisis de su contenido. En este proyecto afrontamos esta última etapa, realizando un estudio de diversos clasificadores que determinan si una región contiene un reflejo o no a partir del análisis de unas características de las regiones que se han definido específicamente para este caso. Los resultados preliminares ofrecen un buen rendimiento de los clasificadores estudiados, mostrando asimismo la incidencia de parámetros clave como el tamaño de los conjuntos de entrenamiento y test o los porcentajes de solapamiento a la hora de hacer el entrenamiento.

Palabras Clave— Adaptative Boost, Aprendizaje computacional, Clasificación de reflejos, Imagen médica, Random Forest, Support Vector Machine.

Abstract— Colon cancer is one of the cancers with the highest incidence in the world population. The quick detection requires an early identification of polyps, which are the precursor lesion. Doctors are helped by automatic systems in order to the early detection. This automatic detection poses some problems when it comes to the appearance of specular highlights which affect the output of the said methods. The automatic detection of the specular highlights is a solution for this problem. It is often done in two stages: a previous segmentation of the image and a posterior classification of those regions through an analysis of their content. In this project, we deal with this latter stage by carrying out a study of several classifiers which determine if a region contains or not a specular highlight from the analysis of the characteristics of the regions that have been specifically defined for this case. The preliminary results offer a high performance of the studied classifiers, showing also the incidence of the key parameters like the size of the unit of training set and test set or the percentages of overlapping when it comes to do the training.

Index Terms— Adaptative Boost, Machine Learning, Medical Image, Random Forest, Specular Highlights Clasification, Support Vector Machine.

1. Introducción

El contexto de este trabajo es la detección automática de pólipos en imágenes de colonoscopia. La detección de pólipos es clave para un diagnóstico precoz del cáncer de colon, que es uno de los cánceres con más incidencia en la población mundial.

Un pólipo es un crecimiento anormal de tejido que surge de la capa interior o mucosa del intestino grueso (colon) y sobresalen al canal intestinal (luz) [1]. Por otro lado, una colonoscopia es una prueba médica que sirve para diagnosticar y tratar enfermedades del colon [2].

Uno de los problemas que se presentan en la detección de pólipos mediante una colonoscopia viene dado porque las imágenes obtenidas incluyen regiones de reflejos especulares procedentes de la propia iluminación aplicada para explorar la zona.

Uno de los algoritmos empleados para la detección de pólipos usa una definición de los mismos a partir de valles de intensidad [3]. El funcionamiento de este método se ve afectado por la presencia de reflejos especulares, que también llevan asociada información de valles [4]. Por tanto, una detección de los mismos y una sustitución de su contenido por aproximaciones de la vecindad permitiría reducir la información de valles no asociada a los pólipos.

1.1. Motivación

La motivación principal del proyecto es poder aplicar herramientas vistas durante la carrera a la hora de resolver problemas reales, como es en este caso, ayudar en la detección de pólipos. Asimismo, otra motivación ha sido el afianzar el conocimiento de las técnicas que el aprendizaje computacional puede ofrecer y que gracias a este proyecto se va a poder profundizar en su manejo posibilitando obtener la información necesaria para poder aplicarla a diversos campos.

1.2. Objetivos a alcanzar

El objetivo final de este proyecto es lograr obtener cuál es el mejor clasificador para la detección de regiones de reflejo en imágenes de colonoscopia.

Para ello, el esquema de procesado propuesto parte de una segmentación previa de la imagen en una serie de regiones, que pueden o no contener reflejos. El objetivo principal de este proyecto es verificar qué características y qué

- *marcos.molero@e-campus.uab.cat*
- *Mención de Computación.*
- *Tutorizado por: Jorge Bernal. David Vázquez (Ciències de la Computació).*
- *Curso 2015/16.*

método de clasificación son los más adecuados para que el sistema a la salida sólo proporcione regiones de reflejo.

1.3. Metodología del proyecto

Para el desarrollo de este proyecto se ha decidido seguir la metodología SUM [5] dado que el equipo de desarrollo es de una sola persona y debemos ser muy flexibles a la hora de definir cambios en el ciclo de vida del proyecto.

Esta metodología está basada en SCRUM [6] y tiene como objetivos la mejora continua del proceso para incrementar la eficacia y eficiencia. Pretende obtener resultados predecibles, administrar eficientemente los recursos y riesgos y lograr una alta productividad del equipo de desarrollo. El proyecto se ha planificado a partir de una serie de sprints semanales en los que se ha repasado el trabajo realizado durante el sprint anterior y se han definido nuevas tareas para el siguiente sprint.

1.4. Requisitos del proyecto

Los requisitos del proyecto han sido divididos en dos tipos. Los requisitos funcionales y los requisitos hardware. Estos requisitos se explican a continuación.

1.4.1. Funcionales

Los requisitos funcionales nos definen cómo tiene que funcionar nuestro sistema. Nuestro sistema debe permitir:

- Identificar el mejor clasificador para identificar regiones de reflejo y no reflejo.
- Ajustar los parámetros de cada clasificador por separado para poder optimizar cada uno de ellos.
- Ser ágil a la hora de ejecutar.
- La inclusión de librerías externas o bien debe contener librerías propias de clasificadores.

Por estos motivos el lenguaje de programación y software escogido es MATLAB [7]. Esta elección se debe a que esta herramienta permite un rápido desarrollo de algoritmos ya que no necesita realizar tareas administrativas a bajo nivel para la declaración de variables y la especificación de tipos de datos, entre otros. Además incluye un potente toolbox con funciones propias que permite trabajar con matrices y vectores. Gracias a esto, el tratamiento y procesamiento de imágenes resulta más cómodo.

1.4.2. Hardware

Los requisitos de hardware mínimos para el uso de este software, tanto para entornos Windows como para entonos OS-X son 2-3GB de memoria para una instalación típica de MATLAB y 2GB de memoria RAM [8].

1.5. Planificación del proyecto

La planificación del proyecto ha sido dividida en tres partes. En la primera parte exponemos una lista de tareas en un diagrama de Gantt. En la segunda parte detallamos los recursos disponibles para la realización del proyecto. Finalmente la tercera parte detallamos los riesgos del proyecto.

1.5.1. Lista de tareas

En el ANEXO A podemos ver un diagrama de Gantt en dónde se aprecian las tareas propuestas para la realización de este proyecto. Dichas tareas se desgranan en sub tareas en las cuales se les asocia un periodo en el tiempo para la realización.

1.5.2. Recursos disponibles

Como recursos para la realización del proyecto contamos con un ordenador portátil para la programación y un ordenador de sobremesa para ejecutar de manera eficiente el código generado. Ambos cumplen los requisitos mínimos para la utilización del software de programación escogido. También contamos con librerías MATLAB para la implementación de clasificadores. Adicionalmente disponemos de acceso a la base de datos CVC Clinic-DB [3] y a una librería de segmentación de imágenes.

1.5.3. Riesgos del proyecto

Debido a que pueden aparecer imprevistos durante la realización del proyecto, se ha elaborado la TABLA 1 con una lista de posibles causas de riesgo. Estas causas están asociadas a la probabilidad de que sucedan y al nivel de impacto. Además, se ha trazado un plan de contingencia como solución a la aparición de alguno de los riesgos detallados.

TABLA 1
RIESGOS DEL PROYECTO

Riesgo	Causa	Probabilidad	Impacto	Solución
Planificación temporal optimista	Sucede cuando asociamos un tiempo no realista para alguna de las tareas.	Baja	Crítico	Ser coherente en el momento de la planificación. No querer ver más allá sino saber ajustar el tiempo en el que se pueda realizar con éxito el proyecto.
Pérdida de rendimiento	Sucede por acumulación de trabajo y baja productividad.	Baja	Crítico	Llevar un rol constante y diario de trabajo.
No disponer de suficientes imágenes	Sucede si la BBDD que tenemos no satisface las necesidades.	Baja	Marginal	Solicitar a nuestro tutor o co-tutor más imágenes para incorporar a nuestra base de datos o bien buscar otra fuente.
Falta de información	Sucede al no disponer de la información necesaria para desarrollar el proyecto.	Media	Crítico	Variar el enfoque del objetivo de manera que con el nuevo enfoque tengamos más visión/información.
Mala documentación	Sucede cuando los experimentos no quedan documentado.	Media	Marginal	Realizar un documento de log y anotar todos los progresos que realicemos.

1.6. Estructura del documento

El resto del documento está organizado de la siguiente forma: Empezaremos mostrando el estado del arte de los clasificadores en la actualidad. A continuación explicaremos la metodología seguida para la extracción automática de regiones de reflejo. Más adelante, mostraremos la evaluación de los experimentos realizados. Finalizaremos el artículo mostrando las conclusiones principales y el trabajo futuro. Adicionalmente, después de los agradecimientos y la bibliografía, el apéndice contendrá información que ayuda a completar el documento.

2. Estado del arte

En esta sección revisaremos el estado del arte de la segmentación de imágenes y de la clasificación de reflejos.

El contexto de este trabajo es el desarrollo de sistemas inteligentes para colonoscopia, en los cuales se diseñan sistemas de apoyo al diagnóstico para ayudar a los médicos durante la intervención. Uno de los sistemas más demandados es aquél que ayude a la detección automática de pólipos. Estudios recientes [3] demuestran que los pólipos pueden ser detectados usando detectores de valles pero que hay otros elementos en la escena, como los reflejos especulares, que tienen información de valle asociada. Por tanto, uno de los elementos a tener en cuenta será la detección automática de reflejos especulares para poder ayudar a mejorar la detección de pólipos.

2.1. Segmentación de regiones

La detección automática de reflejos especulares ha sido afrontada por diversos autores en la literatura aunque estudios recientes informan de que los métodos generales [9] no ofrecen buen rendimiento en imágenes endoscópicas, debido a las particularidades del medio en el cual se refleja la luz. Debido a ello, han surgido diferentes alternativas específicas dentro del campo de procesado de imagen endoscopia, y, particularmente, dentro del contexto siguiente: cómo puede afectar la detección de los mismos a otras etapas de procesado.

Actualmente existen solamente dos métodos reportados en la literatura [10], [4], siendo el segundo un refinamiento del primero. En este caso la mayor dificultad viene dada por la imposibilidad de delimitar correctamente la región de reflejo así como por la aparición de reflejos en zonas oscuras de la imagen, dificultando su simple detección por umbralización respecto al nivel de gris.

La alternativa propuesta en este artículo pretende usar una segmentación previa de la imagen en regiones homogéneas para luego clasificarlas en reflejo o no según su contenido, ajustándose a un modelo de reflejo introducido a partir de las características a usar en el proceso de clasificación.

2.2. Características

La detección de reflejos no ha sido tratada en la literatura, hasta donde se ha observado, como un problema de clasificación clásico de regiones. La aplicación de algoritmos clásicos de extracción de características, ya sea basados en textura como SURF [11] o basados en contornos como EHD [12] podrían ser usados como mecanismos para detectar dónde están los reflejos en la imagen pero no para segmentarlos, que es uno de los objetivos a alcanzar para lograr mitigar completamente su efecto para la detección de pólipos.

2.3. Clasificadores

Los diferentes tipos de algoritmos de aprendizaje automático se pueden agrupar según el tipo de aprendizaje. A continuación se explican brevemente los tipos de algoritmos más representativos de cada clase.

2.3.1. Aprendizaje inductivo supervisado

Los algoritmos de aprendizaje inductivo supervisado disponen de un agente externo que en la fase de entrenamiento, determina si la respuesta generada debe o no pertenecer a una clase.

En la actualidad existen distintos tipos de clasificadores supervisados. *Naïve Bayes* es un clasificador probabilístico fundamentado en el teorema de Bayes [13]. Además de la clasificación de datos, *Support Vector Machine* (SVM), permite realizar regresiones de datos [14] aplicando varios tipos de Kernels (Lineal, Polinomial, Radial Basis Function). *Decision Trees* es una representación en forma de árbol cuyas ramas se bifurcan en función de los valores tomados por las variables. Estas bifurcaciones acaban terminando en una acción concreta [15]. *Random Forest* es una combinación de árboles predictores [16] tal que cada árbol predictor depende de los valores de un vector aleatorio probado independientemente y con la misma distribución de probabilidad para cada uno de estos. Otro algoritmo de aprendizaje inductivo supervisado es *Adaptative Boosting*, se trata de un algoritmo iterativo [17] que se ajusta adaptativamente a los errores generados aumentando los pesos asociados a los patrones mal clasificados y a bajando los pesos asociados de los patrones bien clasificados.

2.3.2. Aprendizaje inductivo no supervisado

Al contrario que los algoritmos anteriores, los algoritmos de aprendizaje inductivo no supervisado no disponen de un agente externo que determine si la respuesta generada pertenece o no a una clase.

Todo el proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas al sistema. No se tiene información sobre las categorías de esos ejemplos. Por lo tanto, en este caso, el sistema tiene que ser capaz de reconocer patrones para poder etiquetar las nuevas entradas.

Uno de los algoritmos más sencillos de aplicar y con buenos resultados de clasificación es *K-means*. El procedimiento de clasificación de datos de *K-means* [18] pasa por definir previamente un número de clases k. La idea principal es definir los centroides de k, uno para cada clase y agrupar los datos calculando la distancia mínima a cada uno de estos centroides. El algoritmo *Mixture Gaussian* [19] es un modelo probabilístico que asume que todos los puntos de datos se generan a partir de una mezcla de un número finito de distribuciones gaussianas con parámetros desconocidos.

2.3.3. Aprendizaje semisupervisado

Los algoritmos de aprendizaje semisupervisados se sitúan entre los algoritmos de aprendizaje supervisado y los algoritmos de aprendizaje no supervisado. Cada algoritmo, además de los datos no etiquetados [20], [21], está provisto de alguna información supervisada pero esta información no está disponible necesariamente para todos los ejemplos.

2.3.4. Aprendizaje por refuerzo

Los algoritmos de aprendizaje por refuerzo permiten al agente determinar un comportamiento ideal sin un contexto específico para así maximizar el rendimiento. Los agentes disponen de un sistema de recompensas [22] que les permite puntuar los datos obtenidos según la clasificación obtenida.

3. METODOLOGÍA

Referente a la metodología, el primer paso es segmentar las imágenes de entrada. A continuación es necesario describir las regiones segmentadas usando características definidas ad-hoc para ajustarse a los reflejos especulares. Finalmente, se integrarán estas características dentro de un esquema de clasificación para poder diferenciar regiones que son parte de un reflejo de las que no. Finalmente es necesario el empleo del clasificador escogido para clasificar las regiones en reflejos y no reflejos.

3.1. Pipeline general

En la Fig. 1 podemos visualizar el pipeline general que seguimos.

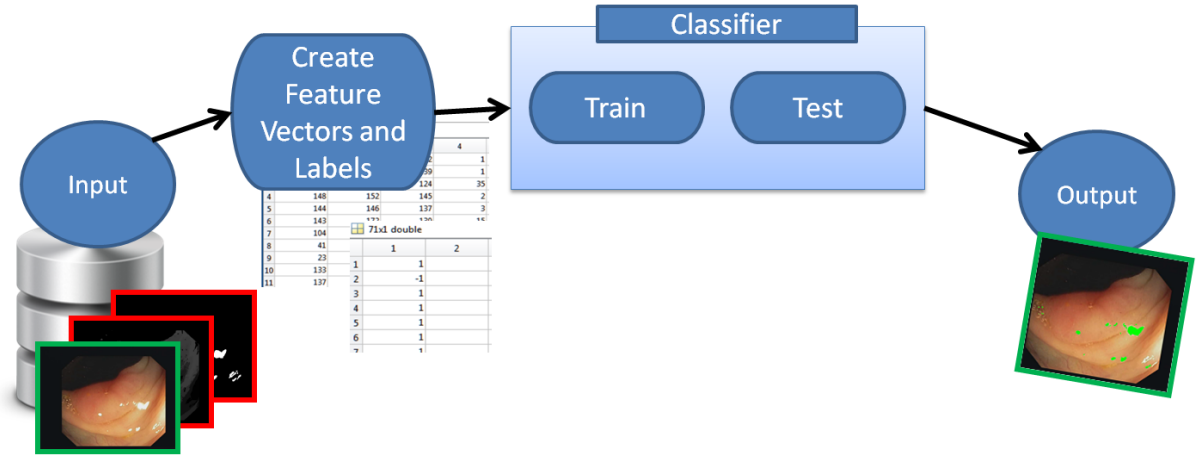


Fig. 1. Pipeline general. El proceso que seguiremos está dividido en tres fases. La primera fase se realiza en Input y consiste en una segmentación de regiones. A continuación, la segunda fase corresponde a la obtención de los descriptores/características. Seguidamente, la tercera fase corresponde a la clasificación de la región en reflejo o no reflejo. Como salida Output, obtenemos si una cierta región es reflejo o no.

3.2. Segmentador automático de regiones

Con el fin de reducir la dimensionalidad del problema, se ha optado por realizar una segmentación previa de la imagen enfocada a la detección de reflejos. Esta segmentación se podría considerar encuadrada dentro de los métodos de semilla, en los cuales se selecciona un píxel inicial como representante de una región, que se hace crecer a partir de la similaridad de píxeles vecinos respecto a la semilla. En este caso se ha escogido como semilla los máximos locales de la imagen (definiendo que los reflejos suelen estar formados por los píxeles de mayor intensidad de la imagen) y el criterio de crecimiento ha sido el minimizar el gradiente de intensidad entre la semilla y el pixel a añadir. El criterio de parada de crecimiento de una región se define a partir de un valor de gradiente medio umbral del contorno de la región en cada instante: en el momento en que este gradiente medio respecto a la semilla supera un determinado valor, se detiene el crecimiento de la región.

3.3. Características utilizadas

El segundo de los pasos de la metodología corresponde a la extracción de características, o features, de las regiones segmentadas. Esta descripción de features se ha realizado de dos formas, la primera corresponde a la elaboración de features manuales o hand-crafted mientras que la segunda corresponde al uso de una librería de llamada "regionprops" [23] dentro de MATLAB.

Las features hand-crafted (H) han sido diseñadas específicamente para la detección de brillos en la imagen. Una muestra de este tipo puede ser el contraste y la saturación.

Las features obtenidas mediante regionprops (R) son más genéricas. Dentro de este tipo podemos encontrar el área de la región, la excentricidad o la orientación. En el ANEXO B se describen las features utilizadas.

3.4. Clasificadores

El último paso es el de clasificación de las regiones resultantes en dos categorías: reflejo y no reflejo. Para ello se ha optado por la utilización de algoritmos de aprendizaje

supervisado. Estos algoritmos toman como entrada un conjunto de ejemplos para realizar la fase de entrenamiento (train). Cuando el entrenamiento del clasificador concluye, el algoritmo crea un modelo que predice la clase binaria de la siguiente muestra a evaluar. Esta parte es denominada fase de validación (test). Hemos utilizado los siguientes tres clasificadores.

3.4.1. Support Vector Machines

El algoritmo Support Vector Machines, es un algoritmo de aprendizaje inductivo supervisado desarrollado originalmente en 1963 por Vladimir N. Vapnik y Alexey Ya. Chervonenkis. El estándar actual que hoy conocemos fue publicado en 1995 por Corinna Cortes y Vladimir N. Vapnik [24]. La idea básica de SVM es la de construir un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta para realizar una clasificación o una regresión. El objetivo de SVM es que estos hiperplanos separen de forma óptima las clases de las que se compone el espacio. Esta optimización se produce cuando los márgenes del hiperplano quedan maximizados, es decir que la distancia entre los puntos del espacio sea la mayor.

El clasificador SVM utilizado para realizar los experimentos es LIBLINEAR para MATLAB de la National Taiwan University [25].

En la fase de entrenamiento, el clasificador utilizado, cuenta con una serie de parámetros mostrados en la TABLA 2, que según como sean fijados el comportamiento varía, viéndose así, afectado el resultado.

TABLA 2
PARÁMETROS SVM LIBLINEAR

Parámetro	Descripción
-s	Corresponde al tipo. (Clasificación o Regresión).
-c	Corresponde al parámetro coste.
ftrain-norm	Corresponde a la matriz de feature vectors de entrenamiento.
ltrain	Corresponde a las labels de entrenamiento.
-q	Modo silencioso. No hay salidas.

3.4.2. Adaptative Boosting

Adaptative Boosting o AdaBoost es un algoritmo de aprendizaje supervisado presentado en 1995 por Y. Freund y R. Schapire. Este algoritmo debe su nombre a que es un tipo de Boosting que se ajusta adaptativamente a los errores generados aumentando los pesos asociados a los patrones mal clasificados y a bajando los pesos asociados de los patrones bien clasificados [26].

En Adaboost inicialmente a todos los datos del conjunto de entrenamiento se les asigna un peso idéntico y se entrena el modelo usando el set de entrenamiento. A continuación se calcula el error del modelo en el set de entrenamiento, se cuentan cuántos ejemplos han sido mal clasificados y se identifican cuales son. Seguidamente se incrementan pesos en los casos de entrenamiento que el modelo calcula erróneamente para posteriormente entrenar un nuevo modelo usando el conjunto de pesos modificados.

Estos pasos pueden repetirse según el número de iteraciones fijadas. Como resultado final, el modelo resultante viene definido por la votación ponderada de los pesos de todos los modelos.

El clasificador Adaboost utilizado para realizar los experimentos es la librería para MATLAB GML_AdaBoost_Matlab_Toobox_0.3 de Alexander Vezhnevets de Moscow State University [27].

De igual manera que el clasificador SVM, Adaboost cuenta con una serie de parámetros de entrada y salida detallados en la TABLA 3.

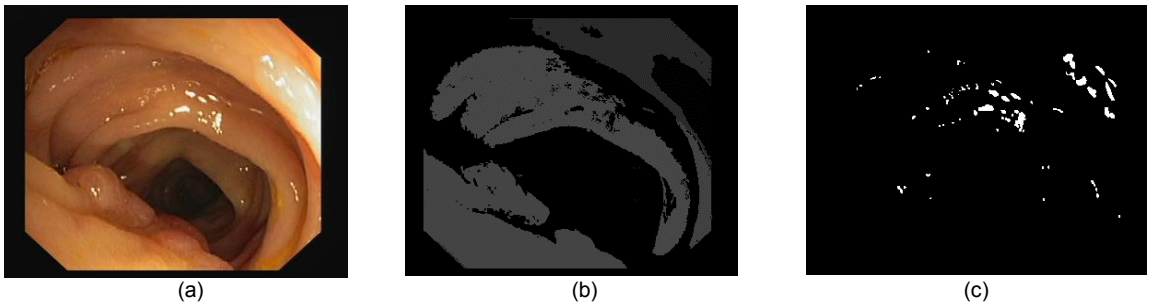
TABLA 3
PARÁMETROS REAL ADABOOST

Parámetro	Descripción
weak_learner	Corresponde a la salida del clasificador débil. En este caso un decision Trees. Este clasificador nos permite aplicar un modelo de predicción que tiene una probabilidad de acertar más alta que en caso de hacerlo aleatoriamente.
ftrainnorm	Corresponde a la matriz de feature vectors de entrenamiento.
ltrain	Corresponde a las labels de entrenamiento.
Num_iter	Define el número de iteraciones.
RWeights	Pesos de árboles anteriores ya entrenados.
RLearners	Función de decisión entrenada anteriormente.

3.4.3. Random Forest

Random Forest es un algoritmo de aprendizaje supervisado propuesto por Leo Breiman y Adele Cutler. Es una combinación de árboles predictores en los que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos [28].

Este algoritmo utiliza la técnica de Bagging. Esta técnica



(a) (b) (c)

Fig. 2. Imagen de ejemplo del dataset: (a) Imagen original; (b) Salida de segmentador de regiones y (c) Ground truth.

de aprendizaje computacional, consiste en modificar el conjunto de aprendizaje mediante la selección aleatoria de elementos de este conjunto, construyendo así los datos para cada clasificador. Los clasificadores son agregados utilizando el voto mayoritario.

El algoritmo Random Forest crea subconjuntos aleatorios de features y labels. Una vez tenemos subconjuntos aleatorios, se aplican árboles de decisión para obtener una predicción del subconjunto. Para obtener la predicción final se reúnen todas las predicciones realizadas por los subconjuntos y se aplica el voto mayoritario para obtener la predicción resultante.

Para realizar los experimentos relacionados con este clasificador, hemos utilizado las funciones ya incorporadas en Classification Ensembles del Toolbox Statistics and Machine Learning de MATLAB [29].

De igual manera que los clasificadores presentados anteriormente, Random Forest cuenta con una serie de parámetros de entrada y salida detallados en la TABLA 4.

TABLA 4

PARÁMETROS RANDOM FOREST

Parámetro	Descripción
n_trees	Define el número de árboles.
ftrainnorm	Corresponde a la matriz de feature vectors de entrenamiento.
ltrain	Corresponde a las labels de entrenamiento.
ltrain	Corresponde a las labels de entrenamiento.
Method	Define el método mediante el cual usaremos la función. En nuestro caso este parámetro debe ser fijado a "Classification".
minLeafs	Define el número de observaciones por hoja.

4. EXPERIMENTOS

En esta sección, explicaremos el contenido de la base de datos desde la cual partimos y las métricas que utilizamos para evaluar los resultados obtenidos. Seguidamente pasamos a explicar el preprocesado necesario antes de la evaluación de los experimentos con los algoritmos de aprendizaje supervisado escogidos.

4.1. Base de datos

La base de datos utilizada a lo largo de este proyecto es CVC-ClinicDB [3]. Esta base de datos cuenta con 612 imágenes procedentes de colonoscopias reales con su ground truth asociado. En la Fig. 2 se muestra un ejemplo para una imagen original (a) regiones iniciales (b) y ground truth (c).

4.2. Métricas de evaluación

Para evaluar los resultados de los experimentos se presenta en la TABLA 5 las métricas utilizadas.

TABLA 5

MÉTRICAS

Nombre	Descripción
True Positive	Correctamente identificado
False Positive	Incorrectamente identificado
True Negative	Correctamente desestimado
False Negative	Incorrectamente desestimado
Precision	$prec = \frac{100 * tp}{tp + fp}$
Recall	$rec = \frac{100 * tp}{tp + fn}$
Accuracy	$acc = \frac{100 * (tp + tn)}{tp + fp + tn + fn}$
F1 score	$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$

4.3. Preprocesado

Antes de empezar a evaluar los diferentes clasificadores debemos realizar un preprocesado de los datos. Una vez se han extraído las características de las imágenes, se procede a normalizar los datos. Esta normalización se produce tanto en filas (NF) como columnas (NC) de nuestro vector de características. Se ha comprobado que aplicando ambas normalizaciones (NC + NF) los resultados de clasificación mejoran. Véase TABLA 6.

TABLA 6

IMPACTO NORMALIZACIÓN DE DATOS

	Baseline	NC	NF	NC+NF
TP	478	1023	665	**1039**
FP	26	42	56	**40**
TN	1631	1615	1601	**1617**
FN	1018	473	831	**457**
Prec	94.84	96.05	92.23	**96.29**
Rec	31.95	68.38	44.45	**69.45**
Acc	66.88	83.66	71.88	**84.23**

4.4. Evaluación Support Vector Machines

4.4.1. Incidencia de parámetros

A pesar que calculamos todas las métricas presentadas en el apartado 4.2 Métricas de evaluación, nos basaremos en la métrica F1-Score para determinar que set de parámetros es mejor.

Durante la evaluación del clasificador SVM vamos a modificar los siguientes parámetros para intentar conseguir un buen resultado. El parámetro "fin_train" corresponde al número de imágenes que utilizamos para el entrenamiento del clasificador. El parámetro "c" es el parámetro coste. El parámetro "solap_pos" corresponde al porcentaje de solapamiento positivo, mientras que "solap_neg" corresponde al porcentaje de solapamiento negativo.

Para el experimento cuyo resultado es mostrado en la TABLA 7 tomamos como referencia un set de entrenamiento de 367 imágenes, que es aproximadamente un 60% del total de nuestra base de datos y utilizamos el 40% restante para realizar el test. Tomamos un array de valores para el porcentaje de solapamiento (tanto positivo como negativo) de [0 20 40 60 80] y otro para el parámetro de coste [0.001 0.01 0.1 1 10 100 1000]. En este experimento se han usado todas las features disponibles (regionprops + hand-crafted).

TABLA 7
RESULTADOS SVM

fin_train	solap_pos	solap_neg	c	tp	fp	tn	fn	prec	rec	acc	f1
367	20	0	10	9267	1099	436	120	89,40	98,72	88,84	**93,83**
367	20	0	100	9262	1101	434	125	89,38	98,67	88,77	93,79
367	20	0	1000	9261	1100	436	126	89,38	98,66	88,78	93,79
367	20	0	1	9267	1108	430	120	89,32	98,72	88,76	93,79
367	20	0	0,01	9330	1187	325	59	88,71	99,37	88,57	93,74
367	20	0	0,1	9268	1121	409	121	89,21	98,71	88,63	93,72
367	20	0	0,001	9382	1346	153	8	87,45	99,91	87,57	93,27

TABLA 8
RESULTADOS SVM HAND-CRAFTED FEATURES

fin_train	solap_pos	solap_neg	c	tp	fp	tn	fn	prec	rec	acc	f1
367	20	0	1	9263	1117	354	124	89,24	98,68	88,57	**93,72**
367	20	0	10	9256	1114	354	131	89,26	98,60	88,53	93,70
367	20	0	100	9256	1114	354	131	89,26	98,60	88,53	93,70
367	20	0	1000	9256	1114	354	131	89,26	98,60	88,53	93,70
367	20	0	0,1	9263	1125	353	124	89,17	98,68	88,50	93,68
367	20	0	0,01	9274	1154	345	113	88,93	98,80	88,36	93,61
367	20	0	0,001	9340	1304	366	50	87,75	99,47	87,76	93,24

4.4.2. Incidencia del set de features

En el siguiente experimento vamos a evaluar qué sucede si nos basamos únicamente en el set de features hand-crafted, que es el diseñado específicamente para detectar brillos, y volvemos a realizar el experimento anterior. Los resultados obtenidos son mostrados en la TABLA 8.

4.4.3. Incidencia de solapamiento

Se ha comprobado empíricamente que los mejores porcentajes de solapamiento son 20% para región positiva y 0% para la región negativa. Si fijamos los parámetros "fin_train" y "c" y variamos únicamente los porcentajes de solapamiento asignando un valor iterativo definido por el array [0 20 40 60 80] obtenemos los resultados mostrados en el ANEXO C.

4.4.4. Discusión

Como resultado a los experimentos realizados con SVM, obtenemos que con respecto a la ejecución con todas las features obtenemos un F1-Score de 93,83 mientras que para la ejecución con las features hand-crafted obtenemos un F1-Score de 93,72. En ambas ejecuciones, las siete primeras combinaciones se han obtenido mediante un porcentaje de solapamiento positivo de 20 y un porcentaje de solapamiento negativo de 0. Por lo tanto podemos afirmar que esta es la combinación que mejor funciona con SVM. Por otro lado, el parámetro coste, debería ser fijado entre 1 y 100 para obtener un buen resultado.

4.5. Evaluación Adaptative Boosting

4.5.1. Incidencia de parámetros

A pesar que calculamos todas las métricas presentadas en el apartado 4.2 Métricas de evaluación, nos basaremos en la métrica F1-Score para determinar que set de parámetros es mejor. Dadas las complicaciones surgidas para obtener resultados con este clasificador, solamente se muestran los mejores resultados en la TABLA 9.

Para el experimento cuyo resultado es mostrado en la TABLA 9 tomamos como referencia un set de entrenamiento de 367 imágenes, que es aproximadamente un 60% del total de nuestra base de datos y utilizamos el 40% restante para realizar el test. Tomamos un array de valores para el porcentaje de solapamiento (tanto positivo como negativo) de [0 20 40 60 80]. Para el parámetro "MaxIter" se ha utilizado un bucle que toma valores de 50 en 50 en cada iteración hasta 500. Para el parámetro "size_tree" se ha utilizado un bucle que para cada iteración toma un valor incremental hasta 20. En este experimento se han usado todas las features disponibles (regionprops + hand-crafted).

TABLA 9

RESULTADOS ADAPTATIVE BOOSTING

tp	fp	tn	fn	prec	rec	acc	f1
9387	1303	374	36	87,81	99,60	87,94	**93,33**

4.5.2. Discusión

Como resultado a los experimentos realizados con Adaptative Boosting, obtenemos que con respecto a la ejecución con todas las features obtenemos un F1-Score de 93,33 usando el set completo de features.

4.6. Evaluación Random Forest

4.6.1. Incidencia de parámetros

A pesar que calculamos todas las métricas presentadas en el apartado 4.2 Métricas de evaluación, nos basaremos en la métrica F1-Score para determinar que set de parámetros es mejor.

Durante la evaluación del clasificador Random Forest vamos a modificar los siguientes parámetros para intentar conseguir un buen resultado. El parámetro "fin_train" corresponde al número de imágenes que utilizamos para el entrenamiento del clasificador. El parámetro "n_trees" es el parámetro que define el número de árboles. El parámetro "minLeafs" corresponde al número de hojas. El parámetro "solap_pos" corresponde al porcentaje de solapamiento positivo, mientras que "solap_neg" corresponde al porcentaje de solapamiento negativo.

Para el experimento cuyo resultado es mostrado en la TABLA 10 tomamos como referencia un set de entrenamiento de 367 imágenes, que es aproximadamente un 60% del total de nuestra base de datos y utilizamos el 40% restante para realizar el test. Tomamos un array de valores para el porcentaje de solapamiento (tanto positivo como negativo) de [0 20 40 60 80]. Para el parámetro "n_trees" se ha utilizado un bucle que toma valores de 84 en 84 en cada iteración hasta 840. Para el parámetro "minLeaf" se ha utilizado el mismo método que en el parámetro n_trees pero cada iteración es de 167 hasta llegar a 1000. En este experimento se han usado todas las features disponibles (regionprops + hand-crafted).

TABLA 10

RESULTADOS RANDOM FOREST

fin_train	n_trees	minLeafs	solap_pos	solap_neg	tp	fp	tn	fn	prec	rec	acc	f1
367	168	167	20	0	9265	1051	477	125	89,81	98,67	89,23	**94,03**
367	588	167	20	0	9265	1052	469	124	89,80	98,68	89,22	94,03
367	504	167	20	0	9264	1050	477	126	89,82	98,66	89,23	94,03
367	252	167	20	0	9263	1050	470	126	89,82	98,66	89,22	94,03
367	840	167	20	0	9263	1051	466	126	89,81	98,66	89,21	94,03
367	672	167	20	0	9262	1049	471	128	89,83	98,64	89,21	94,03
367	336	167	20	0	9261	1049	474	128	89,83	98,64	89,21	94,03

4.6.2. Incidencia de set de features

En el siguiente experimento vamos a evaluar qué sucede si nos basamos únicamente en el set de features hand-crafted, que es el diseñado específicamente para detectar brillos, y volvemos a realizar el experimento anterior. Los resultados obtenidos son mostrados en la TABLA 11.

TABLA 11

RESULTADOS RANDOM FOREST HAND-CRAFTED FEATURES

fin_train	n_trees	minLeafs	solap_pos	solap_neg	tp	fp	tn	fn	prec	rec	acc	f1
367	504	167	20	0	9251	1084	471	139	89,51	98,52	88,83	**93,80**
367	840	167	20	0	9251	1088	467	139	89,48	98,52	88,79	93,78
367	168	167	20	0	9248	1085	474	142	89,50	98,49	88,79	93,78
367	336	167	20	0	9252	1090	467	138	89,46	98,53	88,78	93,78
367	588	167	20	0	9251	1092	463	139	89,44	98,52	88,75	93,76
367	420	167	20	0	9250	1091	468	140	89,45	98,51	88,76	93,76
367	756	167	20	0	9250	1091	460	140	89,45	98,51	88,75	93,76

4.6.3. Incidencia de solapamiento

Se ha comprobado empíricamente que los mejores porcentajes de solapamiento son 20% para región positiva y 0% para la región negativa. Si fijamos los parámetros "fin_train"; "n_trees"; "minLeafs" y variamos únicamente los porcentajes de solapamiento asignando un valor iterativo definido por el array [0 20 40 60 80] obtenemos los resultados mostrados en el ANEXO D.

4.6.4. Discusión

Como resultado a los experimentos, obtenemos que con respecto a la ejecución de Random Forest con todas las features obtenemos un F1Score de 94,03 mientras que para la ejecución con las features hand-crafted obtenemos un F1Score de 93,80. En ambas ejecuciones, las siete primeras combinaciones se han obtenido mediante un porcentaje de solapamiento positivo de 20 y un porcentaje de solapamiento negativo de 0. Por lo tanto podemos afirmar que esta es la combinación que mejor funciona con Random Forest. Por otro lado, el parámetro n_trees tiene un valor muy aleatorio según nuestros experimentos mientras que el parámetro minLeafs debe tener un valor bajo, como podemos ver en los experimentos realizados, los mejores resultados se obtienen fijando el valor a 167.

4.7. Comparativa de clasificadores

Como observamos en la TABLA 12, en cuanto a resultados cuantitativos, el mejor se alcanza utilizando el clasificador Random Forest. Este clasificador tiene un inconveniente y se trata del tiempo de ejecución puesto que SVM es más rápido que Random Forest. Se puede observar como con Adaboost se obtiene el menor número de regiones de reflejo perdidas aunque a coste de un mayor número de falsas detecciones. Por otro lado, SVM presenta resultados muy similares a los de Random Forest, siendo la pequeña diferencia relacionada con la mejor capacidad de este último en cuanto a determinar qué región no tiene contenido de reflejo.

TABLA 12
RESULTADOS COMPARATIVOS

	tp	fp	tn	fn	prec	rec	acc	f1
SVM	9267	1099	436	120	89,40	98,72	88,84	**93,83**
AB	9387	1303	374	36	87,81	99,60	87,94	**93,33**
RF	9265	1051	477	125	89,81	98,67	89,23	**94,03**

5. CONCLUSIONES

En esta sección se analizan las conclusiones más importantes extraídas durante la realización de este proyecto. También se realiza una revisión de la planificación junto con los problemas encontrados. Finalmente se cierra la sección detallando cuales son las líneas futuras a seguir.

5.1. Conclusiones extraídas del proyecto

La primera conclusión que se puede extraer es que se ha verificado el potencial de herramientas de aprendizaje computacional para resolver problemas de detección de estructuras en imágenes.

Dentro de estas herramientas, se ha verificado que la selección del clasificador concreto a emplear tiene tanta importancia como la optimización de sus parámetros, siendo los más relevantes el tamaño de los conjuntos de entrenamiento y test así como la selección de ejemplos positivos y negativos con los que entrenar al sistema.

La comparativa entre los tres clasificadores empleados nos ha permitido obtener un rendimiento similar, aunque presentan diferencias en cuanto a complejidad de configuración y parámetros específicos. El rendimiento observado es prometedor de cara a resolver el problema de clasificación de reflejos.

5.2. Revisión de planificación y problemas encontrados

Durante la realización del proyecto hemos sufrido algunas incidencias. A pesar de ello se ha seguido la planificación tal y como fue prevista.

Entre estas incidencias, encontramos un error al escoger la librería Adaboost. Esta incidencia fue debida a que la librería que escogimos no permitía utilizar un set de entrenamiento adecuado para nuestra base de datos. Para solucionarlo se optó por utilizar la librería que hemos utilizado en este artículo. Aún con esta librería ha sido complicado lograr una ejecución eficiente del algoritmo, lo cual ha dificultado obtener la misma cantidad de resultados que con los otros clasificadores.

Además, hubo dos problemas adicionales: uno con el cálculo de solapamiento de una región dada cuando confluían más de una región del ground truth debajo de una región dada. Como solución se optó por usar como solapamiento el mayor de los solapamientos de todas las regiones del ground truth debajo de la región objetivo. El otro problema fue que al diseñar los experimentos, no valía con usar sólo una variable para diferenciar entre ejemplos positivos y negativos, dado que si cogíamos un porcentaje muy alto, estábamos directamente diciendo que regiones con solapa-miento 60% son ejemplos negativos, cuando un 60% de solapamiento es muy grande. Por ello, usamos dos umbrales. Uno positivo (solapamiento mayor o igual es región positiva) y otro negativo (solapamiento menor o igual es negativo).

5.3. Líneas futuras

El futuro del aprendizaje computacional parece que se enfoca en el uso de redes neuronales convolucionales (CNN) dentro de un entorno conocido como deep learning [30]. Gracias a las capas en las que se componen sus algoritmos, la extracción de información durante el entrenamiento es mucho más eficaz. Por consecuencia el resultado a obtener debería ser más satisfactorio del que los algoritmos de aprendizaje inductivo supervisado nos proporcionan.

AGRADECIMIENTOS

El autor de este artículo desea agradecer el apoyo y el soporte a los tutores del proyecto: Jorge Bernal y David Vázquez. Doy las gracias por guiarme en todo momento durante la realización del proyecto, en ayudarme a profundizar en mi conocimiento del entorno del aprendizaje computacional y, sobre todo, por proporcionarme una experiencia satisfactoria en cuanto a una iniciación a la investigación científica.

El autor también desea agradecer a su familia y amigos por el apoyo recibido durante la realización.

BIBLIOGRAFÍA

[1] The American Society of Colon and Rectal Surgeons (2015) Pólipos del Colon y el Recto. Consultado el 14 de Enero de 2016 en https://www.fascrs.org/patients/disease-condition/polipos-del-colon-y-el-recto

[2] Rivas, P. Colonoscopia. . Consultado el 14 de Enero de 2016 en: http://www.webconsultas.com/pruebas-medicas/colonoscopia-8111

[3] Bernal, J., Sánchez, F. J., Fernández-Esparrach, G., Gil, D., Rodríguez, C., & Vilariño, F. (2015). WM-DOVA maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. Computerized Medical Imaging and Graphics, 43, 99-111. Consultado el 3 de Diciembre de 2015.

[4] Bernal, J., Sánchez, J., & Vilarino, F. (2013, July). Impact of image preprocessing methods on polyp localization in colonoscopy frames. In Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE (pp. 7350-7354). IEEE. Consultado el 7 de Diciembre de 2015.

[5] GEMSERK. SUM para Desarrollo. Consultado el 9 de Octubre de 2015 en: http://www.gemserk.com/sum/

[6] Nicolas Acerenza, Gustavo Meza, and Ariel coppes. (2009) Una metodología para el desarrollo. Consultado el 9 de Octubre de 2015.

[7] Mathworks (2015) Matlab. Consultado el 20 de Septiembre de 2015 en: http://es.mathworks.com/products/matlab/

[8] Mathworks (2015) Matlab System Requirements. Consultado el 20 de Septiembre de 2015 en: http://es.mathworks.com/support/sysreq/current_release/index.html

[9] Feris, R., Raskar, R., Tan, K. H., & Turk, M. (2004, October). Specular reflection reduction with multi-flash imaging. In Computer Graphics and Image Processing, 2004. Proceedings. 17th Brazilian Symposium on (pp. 316-321). IEEE. Consultado el 2 de Diciembre de 2015.

[10] M. Arnold, A. Ghosh, S. Ameling, and G. Lacey, "Automatic segmentation and inpainting of specular highlights for endo-scopic imaging," Journal on Image and Video Processing, vol. 2010, p. 9, 2010. Consultado el 7 de Diciembre de 2015.

[11] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In Computer vision–ECCV 2006 (pp. 404-417). Springer Berlin Heidelberg. Consultado el 20 de Enero de 2016.

[12] Won, C. S., Park, D. K., & Park, S. J. (2002). Efficient use of MPEG-7 edge histogram descriptor. Etri Jour-nal, 24(1), 23-30. Consultado el 20 de Enero de 2016.

[13] Scikit Learn (2015) Naive Bayes. Consultado el 26 de Septiembre de 2015 en: http://scikit-learn.org/stable/modules/naive_bayes.html

[14] Scikit Learn (2015) Support Vector Machines. Consultado el 26 de Septiembre de 2015 en: http://scikit-learn.org/stable/modules/svm.html

[15] Scikit Learn (2015) Decision Trees. Consultado el 26 de Septiembre de 2015 en: http://scikit-learn.org/stable/modules/tree.html

[16] Breiman, L. Random forest Statistics Department, University of California. Consultado el 9 de Octubre de 2015 en: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[17] Emer, Eric. Boosting (adaboost algorithm). Consultado el 26 de Septiembre de 2015 en: http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Eric-Boosting304FinalRpdf.pdf

[18] Andrew, M. K-means and Hierarchical Clustering - Tutorial Slides. Consultado el 4 de Octubre de 2015 en: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html

[19] Scikit Learn (2015) Gaussian mixture models. Consultado el 30 de Septiembre de 2015 en: http://scikit-learn.org/stable/modules/mixture.html

[20] Scikit Learn (2015) Semi-supervised. Consultado el 30 de Septiembre de 2015 en: http://scikit-learn.org/stable/modules/label_propagation.html

[21] Chapelle, O. Schölkopf, B. Zien, A. Semi-Supervised Learning. Consultado el 26 de Septiembre de 2015 en: http://www.acad.bg/ebook/ml/MITPress-%20SemiSupervised%20Learning.pdf

[22] Champandard, A J. Reinforcement Learning Warehouse - Artificial Intelligence Depot. Consultado el 4 de Octubre de 2015 en: http://reinforcementlearning.ai-depot.com/

[23] MathWorks (2015) regionprops. Consultado el 20 de Septiembre de 2015 en: http://es.mathworks.com/help/images/ref/regionprops.html

[24] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning 20 (3): 273. doi: 10.1007/BF00994018. Consultado el 10 de Octubre de 2015.

[25] FAN, Rong-En, et al. LIBLINEAR: A library for large linear classification. The Journal of Machine Learning Research, 2008, vol. 9, p. 1871-1874. Consultado el 20 de Septiembre de 2015 en: https://www.csie.ntu.edu.tw/~cjlin/liblinear/

[26] SCHAPIRE, R (2013) Explaining AdaBoost. Consultado el 4 de Noviembre de 2015 en: https://www.cs.princeton.edu/~schapire/papers/explainingadaboost.pdf

[27] VEZHNEVETS, A (2005) GML AdaBoost Matlab Toolbox. Consultado el 25 de Septiembre de 2015 en: http://graphics.cs.msu.ru/ru/science/research/machinelearning/adaboosttoolbox

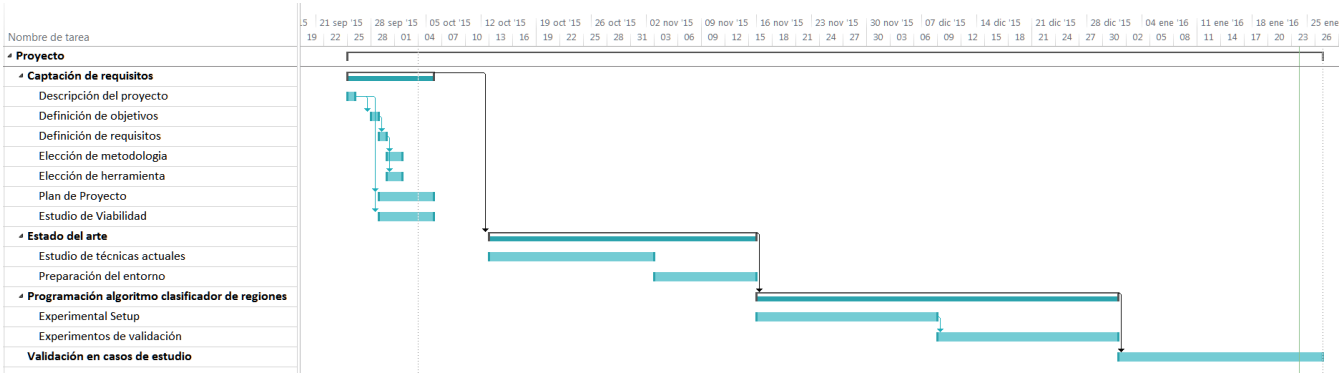
[28] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32. Consultado el 20 de Octubre de 2015.

[29] MathWorks (2015) Tree Bagger. Consultado el 20 de Noviembre de 2015 en: http://es.mathworks.com/help/stats/treebagger.html

[30] Karger, David. Computer Science Professor at MIT. What Is The Future Of Machine Learning? (2015). Consultado el 9 de enero de 2016 en: http://www.forbes.com/sites/quora/2015/02/12/what-is-the-future-of-machine-learning/#11b467e06e8d

ANEXO

A. Diagrama de Gantt



B. Featurevectors

Nº	Tipo	Feature	Descripción
1	H	valor gris medio	Valor medio gris de la imagen.
2	H	valor gris maximo	Valor máximo gris de la imagen.
3	H	valor gris_minimo	Valor mínimo gris de la imagen.
4	H	valor gris sd	Desviación estándar de valor de gris.
5	R	área	Valor escalar que especifica el número de píxeles de una región.
6	R	boundingbox anchox	Devuelve el valor en el eje x del rectángulo más pequeño que contiene la región.
7	R	boundingbox anchoy	Devuelve el valor en el eje y del rectángulo más pequeño que contiene la región.
8	R	majoraxislength	Devuelve un escalar que especifica la longitud (en píxeles) del eje mayor de la elipse que forma la región.
9	R	minoraxislength	Devuelve un escalar que especifica la longitud (en píxeles) del eje menor de la elipse que forma la región.
10	R	eccentricity	Devuelve un escalar que especifica la excentricidad de la elipse de la región.
11	R	orientation	Devuelve un escalar que especifica el ángulo entre el eje x y el eje mayor de la elipse de la región.
12	R	convexarea	Devuelve un escalar que especifica el número de píxeles en 'ConvexImage', el cual, devuelve una imagen binaria (lógica) que especifica el casco convexo, con todos los píxeles dentro del casco rellenado (activado).
13	R	filledarea	Devuelve un escalar que especifica el número de píxeles en el 'FilledImage', el cual, devuelve una imagen binaria (lógica) del mismo tamaño que el cuadro delimitador de la región. Los píxeles activados corresponden a la región, con todos los agujeros rellenados.
14	R	eulernumber	Devuelve un escalar que especifica el número de objetos en la región menos el número de agujeros en esos objetos.
15	R	equivdiameter	Devuelve un escalar que especifica el diámetro de un círculo con la misma zona que la región.
16	R	solidity	Devuelve un escalar especificando el porcentaje de los píxeles en la envolvente convexa que también están en la región.
17	R	extent	Devuelve un escalar que especifica la proporción de píxeles en la región de píxeles en la región delimitada.

18	R	perimeter	Devuelve un escalar que especifica la distancia alrededor del límite de la región.
19	H	dif media gris región contorno	Devuelve la diferencia media de gris del contorno de la región.
20	H	dif max gris región contorno	Devuelve la diferencia máxima de gris del contorno de la región.
21	H	dif min gris region contorno	Devuelve la diferencia mínima de gris del contorno de la región.
22	H	std region contorno	Devuelve la desviación estándar de gris del contorno de la región.
23	H	contraste dentro región	Devuelve el valor de contraste dentro de la región.
24	H	contraste dentro contorno	Devuelve el valor de contraste del contorno de la región.
25	H	dif contraste	Devuelve la diferencia de contraste de la región.
26	H	grad medio contorno	Devuelve el valor del gradiente medio del contorno.
27	H	std grad contorno	Devuelve el valor de desviación estándar del contorno de la región.
28	H	grad max contorno	Devuelve el valor del gradiente máximo del contorno de la región.
29	H	grad min contorno	Devuelve el gradiente mínimo del contorno de la región
30	H	media sat pix reg	Devuelve la media de la saturación de los píxeles de la región.
31	H	media sat pix cont	Devuelve la media de la saturación de los píxeles de contorno.
32	H	dif media sat pix reg pix cont	Devuelve la diferencia entre la media de la saturación de los píxeles de la región y la media de la saturación de los píxeles de contorno.
33	H	max sat pix reg	Devuelve el valor máximo de la saturación de los píxeles de la región.
34	H	max sat pix cont	Devuelve el valor máximo de la saturación de los píxeles de contorno.
35	H	dif max sat pix reg pix cont	Devuelve la diferencia entre el máximo de saturación de los píxeles de la región y el máximo de saturación de los píxeles de contorno.
36	H	min sat pix reg	Devuelve el mínimo de la saturación de los píxeles de la región.
37	H	min sat pix cont	Devuelve el mínimo de la saturación de los píxeles del contorno.
38	H	dif min sat pix reg pix cont	Devuelve la diferencia del mínimo de saturación de los píxeles de la región y el mínimo de saturación de los píxeles de contorno.
39	H	std sat pix reg	Devuelve la desviación estándar de saturación de los píxeles de la región.
40	H	std sat pix cont	Devuelve la desviación estándar de la saturación de los píxeles de contorno.
41	H	dif std sat pix reg pix cont	Devuelve la diferencia de la desviación estándar de la saturación de los píxeles de la región y la desviación estándar de la saturación de los píxeles de contorno.

C. Incidencia de solapamiento para Support Vector Machines

fin_train	solap_pos	solap_neg	c	tp	fp	tn	fn	Prec	rec	acc	f1
367	20	0	1	9263	1117	354	124	89,24	98,68	88,57	**93,72**
367	0	0	1	10130	1379	291	145	88,02	98,59	87,24	93,00
367	40	0	1	8099	1098	377	128	88,06	98,44	87,36	92,96
367	60	0	1	6319	1066	425	128	85,57	98,01	84,96	91,37
367	0	20	1	9196	1617	938	194	85,05	97,93	84,84	91,04
367	20	20	1	9196	1617	938	194	85,05	97,93	84,84	91,04
367	40	20	1	8030	1594	961	198	83,44	97,59	83,38	89,96

D. Incidencia de solapamiento para Random Forest

fin_train	n_trees	minLeafs	solap_pos	solap_neg	tp	fp	tn	fn	prec	rec	acc	f1
367	504	167	20	0	9260	1053	474	129	89,79	98,63	89,17	**94,00**
367	504	167	40	0	8054	977	543	172	89,18	97,91	88,21	93,34
367	504	167	0	0	10103	1277	393	172	88,78	98,33	87,87	93,31
367	504	167	60	0	6083	803	568	115	88,34	98,14	87,87	92,98
367	504	167	80	0	876	114	493	31	88,48	96,58	90,42	92,36
367	504	167	20	20	9232	1557	998	158	85,57	98,32	85,64	91,50
367	504	167	0	20	9230	1565	990	160	85,50	98,30	85,56	91,45