



TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA Y MATEMÁTICAS

Clasificación automática de imágenes biomédicas

Usando técnicas matemáticas y computacionales de visión artificial

Autor

M^a del Mar Alguacil Camarero

Director

Joaquín Fernández Valdivia



ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN



FACULTAD DE CIENCIAS

Granada, septiembre de 2018

Clasificación automática de imágenes biomédicas usando técnicas matemáticas y computacionales de visión artificial

M^a del Mar Alguacil Camarero

Palabras clave: retinopatía diabética, vasos sanguíneos, exudados duros, microaneurismas, segmentación de imágenes, extracción de características, SVM, árboles de decisión, *5-fold cross-validation*

Resumen

La retinopatía diabética es una complicación de la diabetes y una de las causas principales de la ceguera. Su detección temprana, junto con un tratamiento adecuado, puede reducir los riesgos.

En este proyecto se propone un detector automático de la retinopatía diabética basado en la combinación de segmentación de imágenes, extracción de características y clasificación binaria.

El proceso de detección y segmentación de algunas lesiones de la retinopatía diabética se realiza mediante el uso de técnicas de procesamiento de imágenes. Primero se realiza la segmentación de imágenes, que incluye el aislamiento de los vasos sanguíneos, exudados duros y microaneurismas. Posteriormente, se entrena el clasificador de manera supervisada con las características extraídas, lo que permite la clasificación de las imágenes de la retina en sanas o enfermas.

Para comprobar la fiabilidad de los resultados que podemos obtener se ha aplicado la técnica de validación cruzada de 5 iteraciones, evaluando el rendimiento a partir de los parámetros de sensibilidad, especificidad y exactitud.

Todo este proceso se ha realizado con ayuda de MATLAB, una de las herramientas más potentes en el procesamiento de imágenes, que gracias a la potencia y versatilidad de su *toolbox* de procesado de imágenes se simplifican las implementaciones de los algoritmos.

Automatic classification of biomedical images using mathematical and computational techniques of artificial vision

M^a del Mar Alguacil Camarero

Keywords: diabetic retinopathy, blood vessels, hard exudates, microaneurysms, image segmentation, feature extraction, SVM, decision tree, 5-fold cross-validation

Abstract

Diabetic Retinopathy (DR) is a disease of retina, which affects patients with diabetes, and it is a main reason for blindness. Its early detection, together with an appropriate treatment, can reduce the risks. All this has motivated the development of several works related to the inclusion of computational techniques for analysis of retinal images.

Diabetic retinopathy may progress through four stages [1]: mild nonproliferative retinopathy, moderate nonproliferative retinopathy, severe nonproliferative retinopathy and proliferative diabetic retinopathy.

Small areas of balloon-like swelling in the retina's tiny blood vessels, called microaneurysms, occur at the earliest stage of the disease. These microaneurysms may leak fluid into the retina. As the disease progresses, blood vessels that nourish the retina may swell and distort. They may also lose their ability to transport blood. In the third stage, many more blood vessels are blocked, depriving blood supply to areas of the retina. These areas secrete growth factors that signal the retina to grow new blood vessels. And, in the most advanced stage, growth factors secreted by the retina trigger the proliferation of new blood vessels, which grow along the inside surface of the retina and into the vitreous gel. The new blood vessels are fragile, and as a result they are more likely to leak and bleed.

Thanks to the development of digital image processing with methods of information extraction, that improve visualization or highlight interesting features of the images, the diagnosis by specialists is facilitated and even improved in quality. Automated methods of DR screening help to save time, cost and vision of patients, compared to manual methods of diagnosis.

This project proposes an automatic detector of diabetic retinopathy based on the combination of image segmentation, featuring extraction and binary classification.

The process of detection and segmentation of some lesions of diabetic retinopathy is accomplished by the use of image processing techniques. First, the segmentation of image is carried out, which includes the isolation of blood vessels, hard exudates and microaneurysms. Afterwards the classifier is trained so as to allow the extracted features to classify in normal or DR images.

The input retinal images undergo segmentation to detect blood vessels, exudates and microaneurysms separately, according to processes that are effectively described by mathematical formulations. In the RGB images, the green channel exhibits the strongest contrast between the vessels and the background, while the red and blue ones tend to be noisier. Due to this reason, for the segmentation of retinal blood vessels, Contrast-Limited Adaptive Histogram Equalization (CLAHE) is initially applied on this channel. Then, intensity is normalized by expanding through its range, on this image a median filter is used to obtain a background image that will be subtracted from the previous one. A threshold estimated with the Otsu's method is applied to this image to achieve a binary image. Afterwards, a closing with a linear structuring element is applied to accentuate the vessels, in addition to eliminate the small connected elements, in order to remove the noises contained in the binary image.

Hard exudates are yellow flecks made up of lipid residues. Such exudates cause clear lesions, so they can be detected from the red channel on which the Toh-Hat transform with disk-shaped structuring element is applied to obtain the bright components, and then the circular edge is removed. The maximum entropy method is used to achieve a thresholded image and, finally, the remaining parts of blood vessels and optic disc are extracted out of it.

Microaneurysms are the first clinical sign of diabetic retinopathy and they appear as small red dots on retinal fundus images. First of all, a gray-scale image is created from the green and red channel. A median filter is applied and the resulting image is subtracted from the gray image. CLAHE is used for contrast enhancement, and the gray threshold is selected from Otsu's method and an error-correction factor. Then we extract anatomical structures such as blood vessels and exudates. Microaneurysms are dark-reddish in colour and appear as small red dots of 10-to-100 microns of diameter and, as they are circular in shape, we select the circular elements whose radius are contained in the interval [5,50].

Texture means repeating patterns of local variation of pixel intensities. It gives information about the arrangement of surface pixels and their relationship with the surrounding pixels. The features given to the classifier

include neither only the areas of these segmented structures or these areas and textural features obtained based on Gray Level Co-occurrence Matrix (GLCM). These features are the area of blood vessels, area of exudates, area of microaneurysms, contrast, homogeneity, correlation and energy. Area of blood vessels is determined by finding the total number of white (vessel) pixel in the vessel-segmented image. Similarly, the area of exudates and microaneurysms are determined by finding the number of white pixels in the exudates image and microaneurysms image respectively. On the other hand, contrast, homogeneity, correlation and energy are the commonly extracted textural features from GLCM.

Contrast is a measure of the intensity contrast between a pixel and its neighbor over the whole image, homogeneity measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal, correlation calculates the linear dependency of the gray level values in the co-occurrence matrix and energy is the sum of squared elements in the co-occurrence matrix.

In this work, in search of the best result, it was decided to make a comparison between decision tree and Support Vector Machine (SVM) classifier with linear, polynomial of order 4 and Gaussian or Radial Basis Function (RBF) kernel, so the feature vectors are trained with them separately. Both are classification algorithms of supervised learning.

The decision tree classifiers organized a series of test questions and conditions in a tree structure, while SVM performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors.

The 5-fold cross-validation is applied to verify the reliability of results we can obtain, evaluating the efficiency based on parameters of sensitivity, specificity and accuracy.

The cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample and is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and allows us to predict the fit of a model to a hypothetical testing set when an explicit one is not available.

Sensitivity is the percentage of abnormal images (affected by DR) correctly, specificity is the percentage of normal image (not affected by DR) classified as normal by the screening and accuracy is the percentage of image classified correctly. These parameters are estimated from the true positive,

true negative, false positive and false negative, which correspond to the number of cases correctly or incorrectly identified as sick or healthy, respectively.

During the testing process, it was discovered that sensitivity and specificity are inversely proportional, meaning that as the sensitivity increases, the specificity decrease and vice versa. It was also observed that the error rate of classification is influenced by the errors produced when detecting blood vessels, hard exudates and microaneurysms, because they affect the feature vectors necessary for the classification, and these in turn depend on many parameters that may be suitable for some images but not for others.

In this project is used MESSIDOR database. It consists of 1200 images captured using 8 bits per color plane at 1440×960 , 2240×1488 or 2304×1536 pixels. These images have been cropped, thus removing some unnecessary pixels. Two diagnoses have been provided in this database by medical experts for each image such as retinopathy grade and risk of macular edema. In this case, we are only interested in the first parameter, which is simplified by changing the numbers 1, 2 and 3 by 1 (abnormal images), and to 0 (normal images) the others.

All functions related to the automatic detection of diabetic retinopathy have been implemented with the help of MATLAB, one of the most powerful image processing tools which, thanks to the power and versatility of its images processing toolbox, simplifies the implementations of the algorithms.

MATLAB combines numerical computing, graphics and easy-to-use language capabilities. Its language is oriented to matrices, making it ideal for image processing.

Initially, retinopathy diabetic is analyzed in depth in the first chapter, as well as the necessary resources to be able to carry out this work, while, in the second chapter, the planned objectives are defined.

The main purpose of the work cannot be achieved without considering the solid mathematical basis of each function used. For this reason, in the third chapter, the concepts and mathematical tools that have been necessary for the development of the proposed algorithm, which is also described here, are presented and explained.

The fourth chapter is dedicated to the basic functions used and those implemented from them to extract blood vessels, optic disc, circular edge, hard exudates and microaneurysms, in addition to the functions related to

the feature extraction, training and validation.

Finally, a series of tests are carried out to verify the robustness of the algorithm implemented from the retinal images database provided by the research program MESSIDOR. The data included in this database can be used, free of charge, for research and educational purposes.

Índice general

1. Introducción	15
2. Objetivos del trabajo	25
3. Desarrollo del trabajo	27
3.1. Conceptos y herramientas matemáticas	27
3.1.1. De RGB a escala de intensidades	31
3.1.2. Expansión del histograma	31
3.1.3. <i>Contrast-limited adaptive histogram equalization (CLAHE)</i>	32
3.1.4. Filtro de la mediana	33
3.1.5. Operaciones morfológicas	34
3.1.6. Transformada de Hough circular	43
3.1.7. Umbralización	44
3.1.8. <i>Gray Level Co-occurrence Matrix (GLCM)</i>	47
3.1.9. <i>Support Vector Machines (SVM)</i>	48
3.1.10. Árboles de decisión: CART	61
3.2. Algoritmo de detección de retinopatía diabética	68
3.2.1. Detección y segmentación	68
3.2.2. Extracción de características	74
3.2.3. Clasificador	74
4. Diseño e implementación	75
4.1. Funciones básicas	75
4.1.1. Función <code>rgb2gray</code>	75
4.1.2. Función <code>imadjust</code>	75
4.1.3. Función <code>adapthisteq</code>	76
4.1.4. Función <code>medfilt2</code>	77
4.1.5. Función <code>strel</code>	77
4.1.6. Función <code>imerode</code>	77
4.1.7. Función <code>imdilate</code>	78
4.1.8. Función <code>imopen</code>	78
4.1.9. Función <code>imclose</code>	79
4.1.10. Función <code>imfill</code>	79

4.1.11. Función <code>bwconncomp</code>	79
4.1.12. Función <code>imtophat</code>	80
4.1.13. Función <code>imfindcircles</code>	80
4.1.14. Función <code>imbinarize</code>	80
4.1.15. Función <code>graythresh</code>	81
4.1.16. Función <code>immaxentropy</code>	81
4.1.17. Función <code>graycomatrix</code>	81
4.1.18. Función <code>graycoprops</code>	82
4.1.19. Función <code>fitcsvm</code>	82
4.1.20. Función <code>fitctree</code>	83
4.1.21. Función <code>predict</code>	83
4.2. Funciones de detección	83
4.2.1. Funciones auxiliares	84
4.2.2. Función <code>detection_vessels</code>	84
4.2.3. Función <code>detection_hardexudates</code>	85
4.2.4. Función <code>detection_microaneurysms</code>	85
4.3. Funciones de extracción de características	85
4.3.1. Función <code>extraction</code>	85
4.3.2. Función <code>extractions</code>	86
4.4. Funciones de entrenamiento y validación	87
4.4.1. Función auxiliar	87
4.4.2. Función <code>train</code>	88
4.4.3. Función <code>test</code>	88
4.4.4. Función <code>validation</code>	88
5. Análisis de resultados	91
6. Conclusiones y trabajos futuros	97
Bibliografía	99
Anexo 1	105
Anexo 2	107

Índice de figuras

1.1.	Estructura del organo visual humano.	16
1.2.	Microaneurismas en un ojo humano.	17
1.3.	Exudados duros en un ojo humano.	18
1.4.	Exudado duro aumentado.	19
1.5.	Hemorragias retinianas.	19
1.6.	Vasos sanguíneos anómalos.	20
1.7.	Algunas lesiones de la retinopatía diabética.	21
1.8.	Esquema teórico de las lesiones de la retinopatía diabética. .	22
3.1.	Conectividad a 4.	28
3.2.	Conectividad a 8.	28
3.3.	CLAHE	33
3.4.	Ejemplo erosión de una imagen binaria - $(B)_z \subseteq A$	35
3.5.	Ejemplo erosión de una imagen binaria - $(B)_z \cap A^c = \emptyset$. ¹ .	35
3.6.	Ejemplo dilatación de una imagen binaria. ²	36
3.7.	Ejemplo erosión de una imagen en escala de grises. ³	37
3.8.	Ejemplo dilatación de una imagen en escala de grises. ⁴ . . .	38
3.9.	Ejemplo apertura y cierre de una imagen binaria. ⁵	39
3.10.	Ejemplo de relleno de huecos. ⁶	41
3.11.	Ejemplo de extracción de componentes. ⁷	42
3.12.	(SVM) Algunos hiperplanos solución ⁸	50
3.13.	(SVM) Hiperplano de separación óptimo.	52
3.14.	(SVM) Hiperplano solución de ejemplos cuasi-separables li- nealmente.	55
3.15.	(SVM) Hiperplano solución de ejemplos no separables lineal- mente.	59
3.16.	Secuencia de imágenes de detección y segmentación del vasos sanguíneos.	69
3.17.	Secuencia de imágenes de detección y segmentación del borde circular.	70
3.18.	Secuencia de imágenes de detección y segmentación del disco óptico.	71

3.19. Secuencia de imágenes de detección y segmentación de los exudados duros.	72
3.20. Secuencia de imágenes de detección y segmentación de los microaneurismas.	73
6.1. Ejemplo detección incorrecta de los vasos sanguíneos.	98
6.2. Ejemplo detección incorrecta de los exudados duros.	98
6.3. Ejemplo detección incorrecta de los microaneurismas.	98

Índice de tablas

5.1. (B1) Porcentajes obtenidos con el vector de características de tamaño 3	93
5.2. (B1) Porcentajes obtenidos con el vector de características de tamaño 7	93
5.3. (B2) Porcentajes obtenidos con el vector de características de tamaño 3	94
5.4. (B2) Porcentajes obtenidos con el vector de características de tamaño 7	94
8.1. (B1) Sensibilidad - SVM lineal y FV3.	107
8.2. (B1) Sensibilidad - SVM lineal y FV7.	108
8.3. (B1) Especificidad - SVM lineal y FV3.	108
8.4. (B1) Especificidad - SVM lineal y FV7.	108
8.5. (B1) Exactitud - SVM lineal y FV3.	109
8.6. (B1) Exactitud - SVM lineal y FV7.	109
8.7. (B1) Sensibilidad - SVM gaussiano y FV3.	109
8.8. (B1) Sensibilidad - SVM gaussiano y FV7.	110
8.9. (B1) Especificidad - SVM gaussiano y FV3.	110
8.10. (B1) Especificidad - SVM gaussiano y FV7.	110
8.11. (B1) Exactitud - SVM gaussiano y FV3.	111
8.12. (B1) Exactitud - SVM gaussiano y FV7.	111
8.13. (B1) Sensibilidad - SVM polinómico y FV3.	111
8.14. (B1) Sensibilidad - SVM polinómico y FV7.	112
8.15. (B1) Especificidad - SVM polinómico y FV3.	112
8.16. (B1) Especificidad - SVM polinómico y FV7.	112
8.17. (B1) Exactitud - SVM polinómico y FV3.	113
8.18. (B1) Exactitud - SVM polinómico y FV7.	113
8.19. (B1) Sensibilidad - Árbol de decisión y FV3.	113
8.20. (B1) Sensibilidad - Árbol de decisión y FV7.	114
8.21. (B1) Especificidad - Árbol de decisión y FV3.	114
8.22. (B1) Especificidad - Árbol de decisión y FV7.	114
8.23. (B1) Exactitud - Árbol de decisión y FV3.	115
8.24. (B1) Exactitud - Árbol de decisión y FV7.	115

8.25. (B2) Sensibilidad - SVM lineal y FV3.	115
8.26. (B2) Sensibilidad - SVM lineal y FV7.	116
8.27. (B2) Especificidad - SVM lineal y FV3.	116
8.28. (B2) Especificidad - SVM lineal y FV7.	116
8.29. (B2) Exactitud - SVM lineal y FV3.	117
8.30. (B2) Exactitud - SVM lineal y FV7.	117
8.31. (B2) Sensibilidad - SVM gaussiano y FV3.	117
8.32. (B2) Sensibilidad - SVM gaussiano y FV7.	118
8.33. (B2) Especificidad - SVM gaussiano y FV3.	118
8.34. (B2) Especificidad - SVM gaussiano y FV7.	118
8.35. (B2) Exactitud - SVM gaussiano y FV3.	119
8.36. (B2) Exactitud - SVM gaussiano y FV7.	119
8.37. (B2) Sensibilidad - SVM polinómico y FV3.	119
8.38. (B2) Sensibilidad - SVM polinómico y FV7.	120
8.39. (B2) Especificidad - SVM polinómico y FV3.	120
8.40. (B2) Especificidad - SVM polinómico y FV7.	120
8.41. (B2) Exactitud - SVM polinómico y FV3.	121
8.42. (B2) Exactitud - SVM polinómico y FV7.	121
8.43. (B2) Sensibilidad - Árbol de decisión y FV3.	121
8.44. (B2) Sensibilidad - Árbol de decisión y FV7.	122
8.45. (B2) Especificidad - Árbol de decisión y FV3.	122
8.46. (B2) Especificidad - Árbol de decisión y FV7.	122
8.47. (B2) Exactitud - Árbol de decisión y FV3.	123
8.48. (B2) Exactitud - Árbol de decisión y FV7.	123

Capítulo 1

Introducción

En la actualidad, la retinopatía diabética es la causa más frecuente de ceguera entre los 20 y los 65 años en los países industrializados[2]. Debido a que se trata de una enfermedad asintomática al principio que va afectando a la visión de forma progresiva, pudiendo incluso provocar daños irreversibles en esta, se debe realizar un diagnóstico adecuado para detectarla a tiempo.

Gracias al gran avance que se ha experimentado en el campo de la bioinformática se pueden obtener imágenes de la retina con la ayuda de las cámaras no midriáticas. Estas pequeñas cámaras permiten obtener imágenes de la retina o el fondo del ojo sin necesidad de provocar una dilatación de la pupila[3].

Por otro lado, los métodos de procesamiento de imágenes han supuesto una revolución en el campo de la medicina. La utilización de aplicaciones y algoritmos automatizados han facilitado la tarea a diferentes especialistas en este ámbito, dando diagnósticos de forma eficiente y objetiva.

Los métodos automatizados de detección de la retinopatía diabética pueden ayudar a ahorrar tiempo y dinero, en comparación con los métodos manuales de diagnóstico. Todo esto ha motivado el desarrollo de varios trabajos relacionados con la incorporación de técnicas computacionales para el análisis de imágenes de la retina.

Retinopatía diabética

A continuación se realiza un análisis de la información relacionada con la retinopatía diabética necesaria para poder abordar el problema que queremos resolver a lo largo del desarrollo de este trabajo fin de grado, averiguar si un paciente padece esta enfermedad a partir de las imágenes proporcionadas de su retina. Para ello debemos estudiar en profundidad dicha complicación,

reconociendo las lesiones que se producen en la retina en las distintas fases de esta enfermedad y que resultarán de interés para facilitar el reconocimiento de aquellas imágenes pertenecientes a los pacientes que la padecen.

Como podemos leer en [4], la **retinopatía diabética** es una complicación de la diabetes y una de las causas principales de la ceguera. Ocurre cuando la diabetes daña a los pequeños vasos sanguíneos de la retina.

A continuación se muestra la etimología de dicha enfermedad para mayor claridad.

- **retino:** retina, un delicado y fino tejido que recubre la parte interior del ojo, en su parte posterior (como podemos observar en la figura 1.1). Contiene células receptoras de luz, neuronas que comienzan a procesar la información, y fibras nerviosas que llevan el impulso nervioso hacia el nervio óptico.

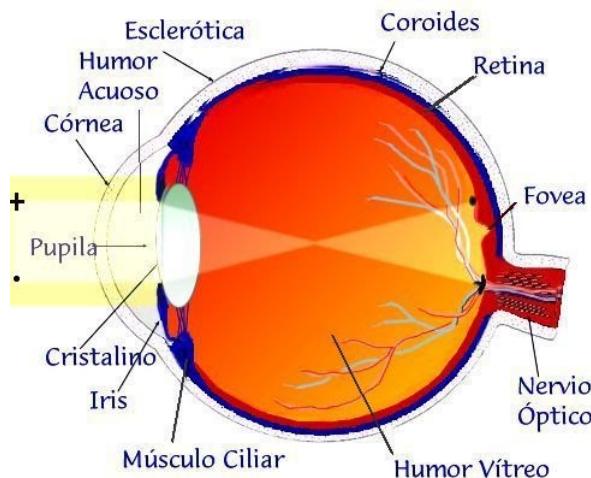


Figura 1.1: Estructura del organo visual humano.

- **-patía:** patología, enfermedad.
- **diabética:** enfermedad frecuente en la que el organismo del paciente no controla adecuadamente los niveles de glucosa en la sangre.

La diabetes es una enfermedad que afecta principalmente a los vasos sanguíneos. La retinopatía diabética es secundaria a un daño producido en la circulación de la sangre por los vasos de pequeño calibre, es decir, por las ramas más finas de las arterias, las venas y los capilares.

La retinopatía diabética tiene cuatro etapas, las cuales se explican a continuación [1, 4, 5]:

1. **Retinopatía no proliferativa ligera:** Etapa más temprana de la enfermedad en la que aparecen pequeñas áreas de inflamación en los pequeños vasos sanguíneos de la retina llamadas **microaneurismas**. Estos microaneurismas pueden filtrar líquido en la retina.

En la exploración, los microaneurismas se ven como manchas rojas de pequeño tamaño, normalmente al lado de un vaso sanguíneo.

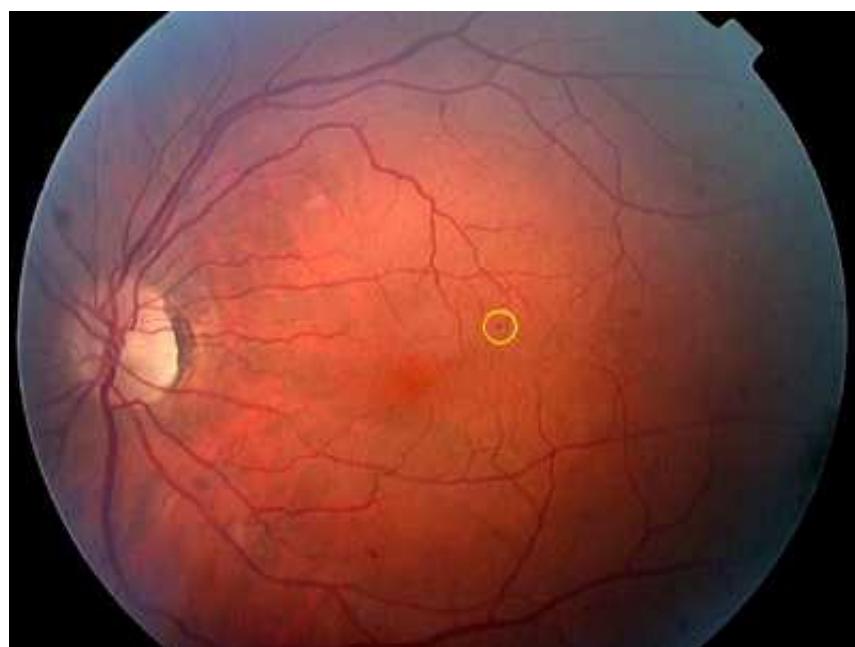


Figura 1.2: Microaneurismas en un ojo humano.

2. **Retinopatía no proliferativa moderada:** Al avanzar la enfermedad, algunos vasos sanguíneos que alimentan la retina se obstruyen.
3. **Retinopatía no proliferativa severa:** Varias partes de la retina dejan de recibir sangre debido a que muchos más vasos sanguíneos se bloquean. En consecuencia, estas áreas de la retina envían señales al cuerpo para que haga crecer nuevos vasos sanguíneos.
4. **Retinopatía proliferativa:** Se produce el crecimiento de nuevos vasos sanguíneos a causa de las señales enviadas por la retina para alimentarse. Estos nuevos vasos sanguíneos son anormales y frágiles, y además crecen a lo largo de la retina y de la superficie del **gel vítreo**, gel incoloro que llena el interior del ojo.

Los vasos sanguíneos dañados por la retinopatía diabética pueden causar una pérdida en la visión de dos maneras:

- En la fase más avanzada de la enfermedad, se pueden desarrollar vasos sanguíneos anómalos y frágiles que pueden gotear sangre en el centro del ojo, opacando la visión.
- Puede suceder que llegase a gotear líquido dentro de la **mácula**, la parte del ojo que provee la visión central clara. Este líquido hace que la mácula se inflame, lo cual ocasiona que se nuble la visión. Esta condición se llama **edema macular** y puede ocurrir en cualquier etapa de la retinopatía diabética, aunque es más probable que suceda al desarrollarse la enfermedad.

Pero el edema también se puede producir en otras zonas de la retina y, además de agua, pueden escaparse moléculas grandes que normalmente no deberían salir del torrente sanguíneo. De entre todas ellas, hay algunas que no son transparentes y sí podemos ver, que son las moléculas grasas o lípidos, que tienen un color amarillento característico. Así, al edema (que es difícil de observar debido a que el agua es transparente) se añaden depósitos amarillos llamados **exudados duros**. Estos exudados duros son fáciles de identificar cuando tienen un tamaño suficiente y de un color amarillo brillante, como se observa en la figura 1.3.

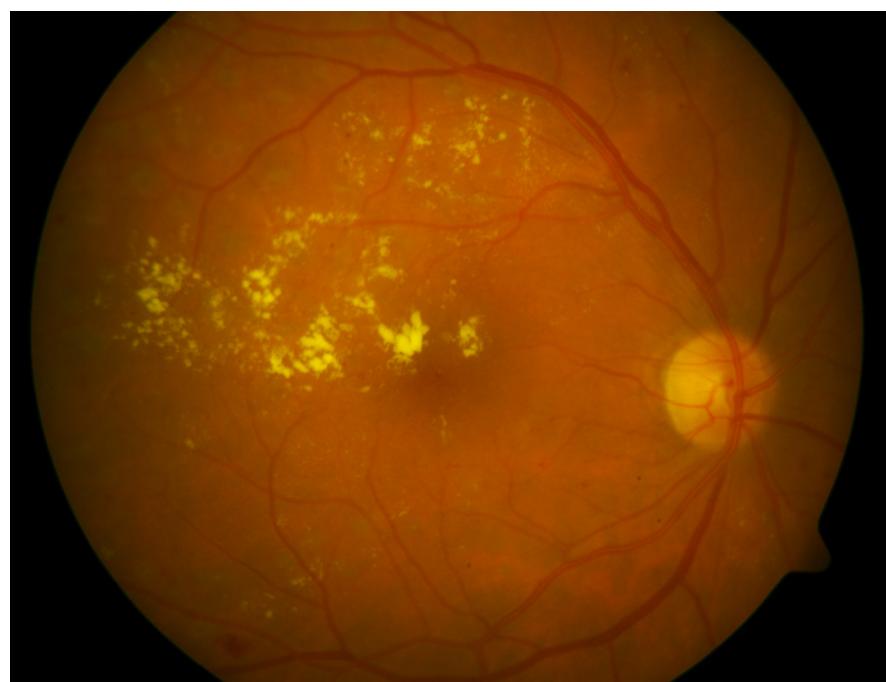


Figura 1.3: Exudados duros en un ojo humano.

Vistos a mayor aumento, estos exudados muestran un aspecto granulado:

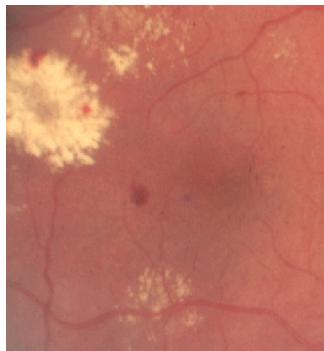


Figura 1.4: Exudado duro aumentado.

Debido al deterioro del vaso sanguíneo, se puede escapar toda la sangre al tejido, produciéndose una hemorragia retinal, que comienzan siendo de pequeño tamaño. A veces, estos sangrados no son redondeados y relativamente pequeños, sino que se ven grandes y alargados:



Figura 1.5: Hemorragias retinianas.

A pesar de todo esto puede seguir funcionando la parte del circuito sanguíneo de la retina en donde los vasos son de pequeño calibre, pero se debe de tener en cuenta que la función principal del sistema circulatorio es aportar nutrientes, sobre todo oxígeno. Cuando en una zona relativamente grande de la retina los capilares, que son vasos sanguíneos muy finos, no funcionan

adecuadamente, el tejido deja de tener suficiente oxígeno y, en consecuencia, se “ahoga” la retina. Eso tiene varios nombres en medicina, siendo el término más utilizado el de **isquemia**, que significa falta de riego sanguíneo. La isquemia como tal no se ve directamente, sino sus consecuencias.

Una parte de la retina que es muy dependiente del oxígeno son las fibras nerviosas, que normalmente son transparentes y no vemos. Cuando falta oxígeno, estas fibras comienzan a sufrir cambios en su interior, las estructuras del interior de la célula no viajan correctamente y se acumulan, aumenta el agua en su interior, etc. Eso lo podemos ver como una lesión blanquecina, llamada **exudado blando o algodonoso**.

Una isquemia mantenida de la retina conduce a la fase final de la retinopatía diabética: la retinopatía diabética proliferativa. El tejido retiniano, por naturaleza muy necesitado de oxígeno, carece de él. Entonces libera unos mediadores químicos que avisan al sistema circulatorio para que llegue más sangre. Estas señales no deberían producirse en circunstancias normales y lo que parece que va a ayudar a la situación, lo empeora. Esas señales químicas reclaman más aporte sanguíneo, y el árbol arterial de la retina responde produciendo nuevos vasos sanguíneos que antes no existían, son los llamados **neovasos**. Pero son vasos sanguíneos anómalos, que no funcionan bien, lo cual produce un crecimiento descontrolado de gravísimas consecuencias.

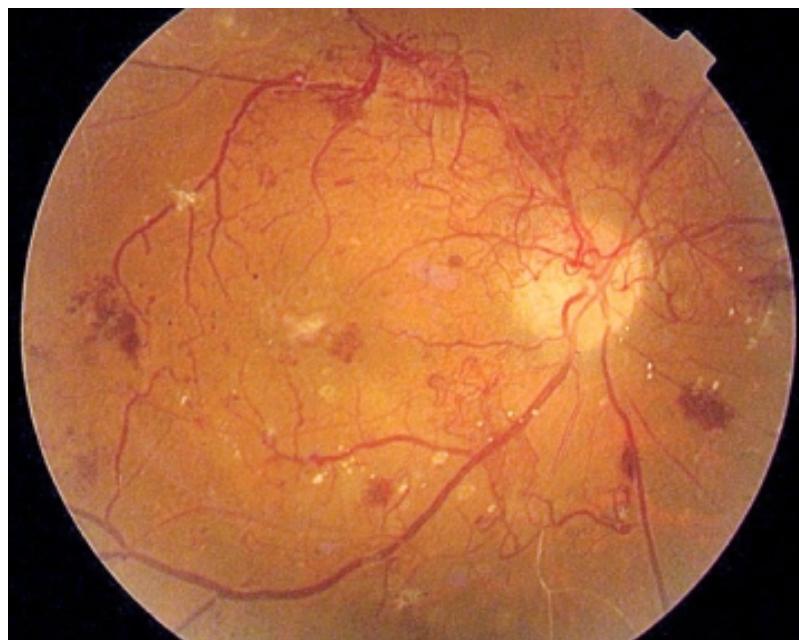


Figura 1.6: Vasos sanguíneos anómalos.

En la figura 1.6 vemos que de los vasos sanguíneos normales (los grandes que tienen una distribución en árbol) salen innumerables vasos pequeños, muy delgados, que se extienden como si fuera una telaraña.

Como hemos visto la retinopatía diabética presenta lesiones microvasculares típicas en la retina de una persona con diabetes: microaneurismas, hemorragias, exudados duros, manchas algonosas y neovasos. Dichas lesiones aparecen como complicación crónica de la diabetes.

Recopilando algunas de las lesiones vistas anteriormente, en la figura 1.7 podemos observar manchas rojas grandes, algunas alargadas en forma de “llamarada”, que son hemorragias. En la parte inferior derecha vemos unas manchas amarillentas, como granos pequeños reunidos, bien delimitadas, que son exudados duros. Y en la parte superior izquierda, además de hemorragias, se aprecian unas lesiones blancoamarillentas que están peor delimitadas, y que no tienen el granulado y el aspecto más redondeado de los exudados duros, estas son exudados algodonosos.



Figura 1.7: Algunas lesiones de la retinopatía diabética.

Por último, a modo de resumen, se muestran las lesiones propias de la retinopatía diabética (microaneurismas, exudados, neovasos y hemorragias) en un esquema teórico:

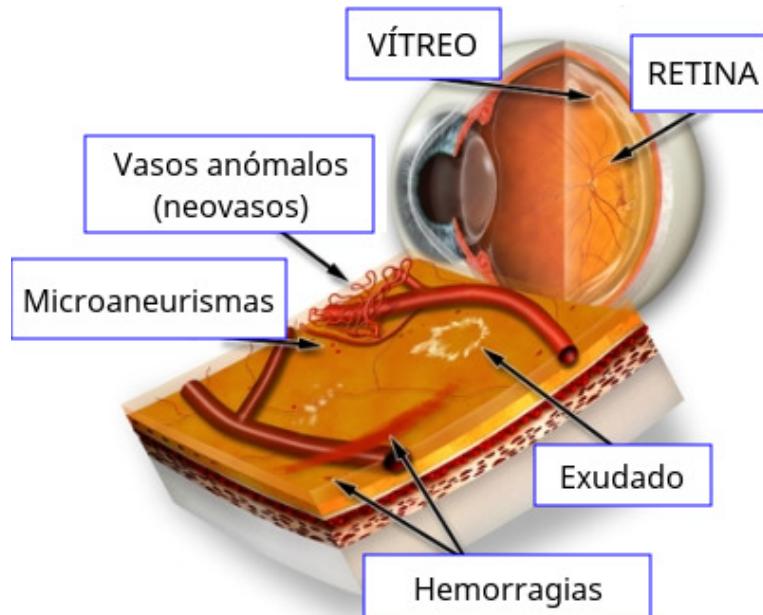


Figura 1.8: Esquema teórico de las lesiones de la retinopatía diabética.

Si no se trata, la retinopatía proliferativa puede causar una pérdida severa en la visión o incluso la ceguera, como anteriormente hemos explicado. Con este trabajo, se pretende detectar esta complicación lo más pronto posible mediante el análisis de una imagen tomada de la retina para poder empezar el tratamiento correspondiente y que haya más probabilidad de que este sea eficaz.

Para conseguir dicho propósito se ha realizado un algoritmo que diagnostica si es probable que una retinografía dada pertenezca a un paciente que padece retinopatía diabética o no.

Recursos

Simplemente basta con un ordenador de características avanzadas con un sistema software que permita la instalación del soporte software de MATLAB. Los precios para obtener una licencia para poder utilizar este depende de varios factores, por ejemplo, si eres estudiante cuesta 35€ ó 69€, mientras que la licencia individual general vale 800€ válida solo para un año ó 2000€

en caso de que quieras que sea perpetua¹. Actualmente MATLAB permite la instalación en las plataformas Windows, Linux y Macintosh.

Estructura de la memoria

Como hemos podido ver a lo largo de este capítulo, inicialmente se ha analizado en profundidad la retinopatía diabética, tanto sus lesiones como sus consecuencias, además de exponer los recursos que han sido necesarios para realizar este trabajo. En el segundo capítulo se definen los objetivos previstos y alcanzados de forma más detallada.

El propósito principal del trabajo no puede cumplirse sin tener en cuenta la fuerte base matemática que conlleva cada función empleada. Por ello, en el tercer capítulo, inicialmente se presentan y explican en profundidad los conceptos y herramientas matemáticas que han sido necesarias para el desarrollo del algoritmo propuesto, abarcando diversas áreas matemáticas como son la estadística y la topología, entre otras. Estas han sido extraídas principalmente del libro *Digital image processing*[6].

En el tercer capítulo también se describe los pasos empleados para conseguir la clasificación automática de retinas sanas o enfermas de forma detallada, basado principalmente en el propuesto en el artículo de *Detección Automática de Retinopatía Diabética basada en Técnicas de Procesamiento de Imágenes*[7].

El cuarto capítulo se encuentra dedicado a hablar de las funciones básicas empleadas, cuya principal fuente de consulta ha sido el soporte de ayuda de MATLAB[8]. A partir de estas se ha implementado las funciones correspondientes para la detección de los vasos sanguíneos, el disco óptico, el borde circular, los exudados duros y los microaneurismas, así como las funciones relacionadas con la extracción de características, el entrenamiento a partir del clasificador SVM o los árboles de decisión, y la validación.

Por último, se procederá a realizar una serie de pruebas para comprobar la robustez del algoritmo implementado a partir de la bases de datos de imágenes de la retina proporcionadas por el programa de investigación Messidor[9].

¹Para más información sobre los precios puede consultarse [10]

Capítulo 2

Objetivos del trabajo

El objetivo de este trabajo consistía en desarrollar un prototipo software en el que se usarán diferentes algoritmos de preprocesamiento, segmentación, extracción de características y clasificación sobre una base de datos de imágenes biomédicas (decidiendo posteriormente profundizar en el problema de la retinopatía diabética, centrándonos en las retinografías) para las cuales se analizaría el problema y buscaría soluciones desde el ámbito del procesamiento digital de imágenes.

Para todo ello se debía hacer un estudio del estado del arte y se desarrollarían las implementaciones más eficientes de los diferentes algoritmos.

Respecto al área matemática, se afirmaba que se haría un estudio sobre el soporte matemático correspondiente a cada uno de los algoritmos que se fuesen a emplear.

Dichos objetivos se han cumplido en su totalidad, pero se ha encontrado dificultades principalmente a la hora de decidir los distintos parámetros de algunas de las funciones relacionadas con las operaciones morfológicas, CLAHE... También se han probado otros algoritmos para la detección de las lesiones típicas de la retinopatía diabética, pero los resultados obtenidos no eran los esperados así que se ha decidido omitir su estudio en este proyecto.

El estado del arte es presentado en la introducción para poder entender desde un principio lo que es la retinopatía diabética y sus problemas asociados, facilitando así la comprensión del resto del documento.

En el tercer capítulo encontramos el soporte matemático que se necesita para poder entender mejor el algoritmo asociado a la detección de esta complicación, empleando herramientas relacionadas con el procesamiento digital de imágenes. Dicho algoritmo también se encuentra explicado en este capítulo.

El capítulo cuarto está dedicado a las funciones utilizadas e implementadas en MATLAB para conseguir desarrollar el prototipo software, mientras que en el resto de capítulos se procede a verificar la validez de este y a extraer conclusiones a partir de ellas.

Capítulo 3

Desarrollo del trabajo

Una vez explicado el objetivo de este trabajo, pasamos a analizar los detalles que debemos tener en cuenta para poder detectar la retinopatía diabética en los pacientes que la padecan a partir de imágenes de la retina, mediante el uso de técnicas matemáticas y computacionales. Pero antes de ello debemos explicar algunos conceptos y resultados previos para una mejor comprensión del algoritmo de detección automática de la retinopatía diabética que en este proyecto se ha decidido llevar a cabo.

3.1. Conceptos y herramientas matemáticas

Podemos definir una **imagen digital** como una matriz, o array bidimensional, de números donde cada celda corresponde a un píxel. Aunque una imagen también se puede interpretar como una superficie bidimensional, y por lo tanto, una imagen digital sería un muestreo discreto de la señal continua. En este proyecto, tomaremos la forma más común de visualizar una imagen, que es la primera definición que hemos expuesto.

La imagen en color está formada por tres canales RGB correspondientes a rojo, verde y azul. Y por tanto, cada píxel viene dado por tres valores numéricos que supondremos que están entre el rango de 0 (ausencia de un color) a 255 (máxima representación de ese color en dicho punto).

Pero debemos tener en cuenta la cercanía o proximidad en una imagen ya que los píxeles próximos tienen una “relación más estrecha” entre sí que los lejanos. Por tanto, tiene sentido definir la **vecindad** de un píxel y la **distancia** entre dos de ellos.

En las imágenes digitales podemos considerar dos tipos de vecindad, la conectividad a cuatro y la conectividad a ocho.

- **Vecindad de 4 píxeles, $N_4(p)$:** Un pixel $p = (x, y)$ tiene cuatro vecinos en dirección horizontal y vertical, cuyas coordenadas vienen dadas por: $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$ y $(x, y - 1)$

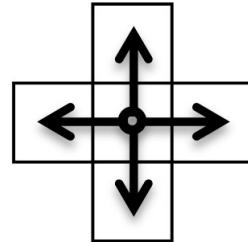


Figura 3.1: Conectividad a 4.

- **Vecindad de 8 píxeles, $N_8(p)$:** Un pixel $p = (x, y)$ tiene ocho vecinos en dirección horizontal, vertical y diagonal, cuyas coordenadas vienen dadas por las 4 anteriores y además: $(x + 1, y + 1)$, $(x + 1, y - 1)$, $(x - 1, y + 1)$ y $(x - 1, y - 1)$

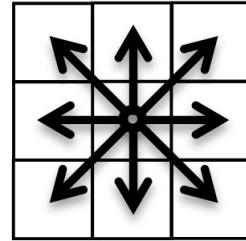


Figura 3.2: Conectividad a 8.

Por otra lado, para $p = (x, y)$, $q = (s, t)$ y $z = (v, w)$, decimos que D es una **distancia** si se verifica:

- $D(p, q) \geq 0$ [$D(p, q) = 0 \Leftrightarrow p = q$]
- $D(p, q) \geq D(q, p)$
- $D(p, z) \leq D(p, q) + D(q, z)$

Algunas de las distancias más utilizadas empleadas en la visión por computador son las siguientes:

- **Distancia euclídea** entre p y q definida como:

$$D_e(p, q) = \sqrt{(x - s)^2 + (y - t)^2}$$

- **Distancia Manhattan** entre p y q definida como:

$$D_4(p, q) = |x - s| + |y - t|$$

Nótese que los píxeles con $D_4 = 1$ son la vecindad de 4 píxeles de p .

- **Distancia de Chebyshov o máxima** entre p y q definida como:

$$D_8(p, q) = \max(|x - s|, |y - t|)$$

Los píxeles con $D_8 = 1$ son la vecindad de 8 píxeles de p .

Normalmente existe una dependencia espacial local debido a que los píxeles de alrededor a uno dado suelen ser parecidos o tener una fuerte relación entre ellos, es decir, el píxel y las regiones vecinas presentan dependencia entre sus valores.

Por tanto, las **características locales** se encargan de describir las propiedades de un píxel de la imagen con respecto a su entorno local o vecindad. Estas características pueden presentar formas específicas con una estructura propia, tales como puntos o contornos, o cualquier otra estructura que se adapte a formas habituales.

Una manera genérica de detectar características locales es usando la operación de la convolución con una forma local definida por un *kernel*. Llamamos **filtro**, **máscara** o **kernel** simplemente a una matriz de coeficientes que asigna una serie de valores a los píxeles vecinos de uno perteneciente a una imagen determinada, mientras que la **convolución** es el proceso de mover un filtro rotado 180° sobre la imagen y computar la suma de los productos en cada posición.

Formalmente, la **convolución** de un filtro $w(x, y)$ de tamaño $m \times n$ con una imagen $f(x, y)$ es una operación matemática que nos da el grado de solapamiento de w sobre f , la cual viene dada por la expresión

$$(w * f)(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t)f(x - s, y - t)$$

La convolución es una correlación con el filtro invertido. Por tanto, la **correlación** de un filtro $w(x, y)$ de tamaño $m \times n$ con una imagen $f(x, y)$ se define como

$$(w \cdot f)(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t)f(x + s, y + t)$$

En consecuencia, si el filtro es simétrico, en todas sus direcciones (diagonales, horizontales y verticales) convolución y correlación pasan a ser totalmente equivalentes.

Es claro que ambas cumplen las propiedades lineas, es decir:

- Distributivas sobre la suma

$$\begin{aligned} w * (f + g) &= (w * f) + (w * g) \\ w \cdot (f + g) &= (w \cdot f) + (w \cdot g) \end{aligned}$$

En consecuencia, si queremos aplicarle un filtro a una imagen resultante de la suma de dos, podemos aplicar el filtro a cada una de ellas por separado y luego sumarlas.

- Los escalares factorizan

$$\begin{aligned} kw * f &= w * kf = k(w * f) \\ kw \cdot f &= w \cdot kf = k(w \cdot f) \end{aligned}$$

Por lo tanto, si queremos aplicar un filtro multiplicado por un escalar k a una imagen, podemos multiplicar esa constante a la matriz imagen y aplicar el filtro, o aplicar el filtro a la imagen y luego hacer el producto de la imagen resultante y la constante k .

Además tanto la convolución como la correlación cruzada son conmutativas e invariantes frente a traslaciones, el operador se comporta igual en todas partes, es decir, el valor de la salida depende del patrón en el vecindario de la imagen, no de la posición de este.

Aparentemente cumplen las mismas propiedades pero esto no es así ya que mientras que la convolución cumple la asociatividad, la correlación, en general, no.

En consecuencia, si suposiesemos que f y g fuesen dos filtros y h una imagen, con la correlación no obtendríamos un núcleo que pudiesemos aplicar mientras que con la convolución sí.

Otro concepto muy relevante al procesar imágenes digitales es el de histograma de una imagen que provee información visual de la distribución de los niveles de intensidad dentro del rango $[0, L - 1]$. El **histograma** de una imagen digital con niveles de intensidad en el intervalo $[0, L - 1]$ es una función discreta $h(r_k) = n_k$, donde r_k es el k valor de intensidad y n_k es el número de píxeles en la imagen con intensidad r_k .

Normalmente se normaliza este dividiendo cada una de sus componentes por el número total de píxeles en la imagen, denotada por el producto

MN , donde M y N son el número de filas y columnas de la imagen. Por lo tanto, el histograma normalizado viene dado por $p_k = n_k/MN$, para $k = 0, 1, 2, \dots, L - 1$. En ese caso, cada valor obtenido representa la probabilidad de obtener un píxel con el valor de intensidad al que está asociada la frecuencia. Claramente, $\sum_{k=0}^{L-1} p_k = 1$.

El análisis del histograma es la base de varias técnicas de procesamiento en el dominio espacial, como mejoramiento de la imagen, compresión, filtrado, segmentación, etc.

3.1.1. De RGB a escala de intensidades

La intensidad de una imagen a color formada por los tres canales RGB se relaciona con la cantidad de luz emitida por ese punto en la escena. Pudiéndose estimar normalmente como la suma de los tres canales. Dicho valor se suele normalizar para que se mantenga en el rango de 0 a 255, siendo una de las normalizaciones más comunes la dada por la siguiente fórmula:

$$(R + G + B) \frac{255}{\max_{i \in I}(r_i + g_i + b_i)}$$

donde $R = (r_i)$, $G = (g_i)$, $B = (b_i)$ corresponden a las matrices relacionadas con el canal rojo, verde y azul, respectivamente.

En el caso de la función `rgb2gray` que nos proporciona MATLAB se realiza la transformación convirtiendo los valores RGB en escala de grises mediante una suma ponderada de las componentes R , G y B :

$$0.2989R + 0.5870G + 0.1140B$$

3.1.2. Expansión del histograma

Se utiliza la técnica de expansión del histograma para aumentar el contraste en imágenes, la cual consiste en expandir el rango de los niveles de intensidad de una imagen dada para ocupar un rango dinámico mayor.

El rango de intensidades se expande de acuerdo a la siguiente expresión:

$$s = (r - c) \frac{a - b}{c - d} + a$$

donde r y s son los niveles de intensidad de entrada y salida, respectivamente, las constantes a y b denotan los límites superior e inferior, respectivamente, del nuevo rango dinámico, mientras que las constantes c y d denotan los valores de intensidad máximo y mínimo, respectivamente, de la imagen original.

Para evitar la influencia de valores atípico que puede afectar al mejoramiento que deseamos obtener al aplicar esta técnica, se suele seleccionar c y d verificando que el 5 % de los píxeles sean mayores que c y el 5 % sean menores que d . Dichos valores se hallan a partir de la función de distribución acumulada discreta que describe la probabilidad de que el k -ésimo nivel de intensidad asuma un valor inferior o igual a k y viene dada por la siguiente fórmula:

$$c(k) = \sum_{i=0}^k p_i, \quad \forall k = 0, 1, \dots, L - 1$$

3.1.3. *Contrast-limited adaptive histogram equalization (CLAHE)*

La ecualización del histograma es una de las técnicas más utilizadas en el mejoramiento del contraste de la imagen, que redistribuye los píxeles sobre el rango $[0, L - 1]$, manteniendo el número de píxeles en la imagen.

Este algoritmo resulta de la realización de los siguientes pasos:

- 1) Se divide la imagen original en $n_x \times n_y$ regiones, ya que si se hiciese mediante el método de ventana deslizante¹ sería computacionalmente costoso a medida que se aumenta el tamaño de la ventana.
- 2) Se calcula la función de transformación de cada región, considerando así la información local que evita que exista bajo contraste en regiones pequeñas. Para ello, inicialmente se determina el histograma de dicha región, a continuación se recorta y se genera la función de transformación definida de la siguiente forma:

$$T(k) = (L - 1) \sum_{i=0}^k p_i, \quad \forall k = 0, 1, \dots, L - 1,$$

donde p_i es la probabilidad de ocurrencia del i -ésimo nivel de intensidad.

- 3) Por otra parte, se selecciona subimágenes de la original a partir de los puntos de referencia.
- 4) Se identifica a qué región pertenece la subimagen y se realiza la interpolación de cada píxel de cada subimagen, evitando así el efecto de “bloque” indeseado:

¹El método de ventana deslizante consiste en definir un vecindario de $N \times M$ píxeles alrededor de uno central, realizar las operaciones oportunas que se deseen y, a continuación, desplazar la región al píxel adyacente, repitiendo el proceso nuevamente.

- Si el píxel r pertenece a una región interna, entonces se interpola usando las cuatro funciones de transformación: adyacentes arriba izquierda, arriba derecha, abajo izquierda y abajo derecha:

$$s = (1 - y)[(1 - x)T_A(r) + xT_B(r)] + y[(1 - x)T_C(r) + xT_D(r)]$$

donde A, B, C y D son los puntos centrales de las regiones vecinas, cuyas funciones de transformación son $T_A(r)$, $T_B(r)$, $T_C(r)$ y $T_D(r)$, respectivamente.

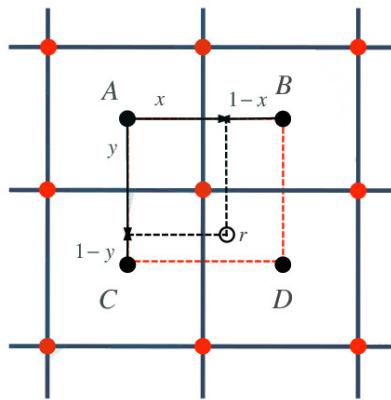


Figura 3.3: CLAHE

- Si el píxel pertenece a una región del borde, entonces interpolar usando las funciones de transformación adyacentes: izquierda y derecha o arriba y abajo.
- Si el píxel pertenece a una región de esquina, entonces interpolar usando la función de transformación que contiene al píxel.

3.1.4. Filtro de la mediana

Como ya sabemos, la **mediana** es un valor (o valores) que ocupan la posición central al ordenar los datos. Es decir, al menos la mitad de los datos son menores o iguales que la mediana y la otra mitad mayores o iguales que esta.

Como indica su nombre, el **filtro de la mediana** es un filtro no lineal que resulta de calcular el valor de la mediana de todos los píxeles de su alrededor. Por tanto, se debe seleccionar un tamaño de la máscara, es decir, la región que consideramos interesante para un determinado vecindario.

La principal función de este filtro es eliminar los picos de intensidad que aparecen aislados en el área de la máscara de filtro. Por tanto, se suele utilizar cuando se observa que una imagen tiene ruido de sal y pimienta.

3.1.5. Operaciones morfológicas

En el contexto del análisis de imágenes, se denota **morfología matemática** al conjunto de herramientas empleadas para extraer componentes de las imágenes que son útiles en la representación y descripción de la forma de los objetos.

Los operadores de morfología matemática son un conjunto de filtros sencillos, que se pueden combinar para obtener resultado más complejos.

Algunas de las operaciones en este apartado se desarrollará tanto en imágenes binarias como en imágenes en escala de grises. En imágenes binarias, las componentes son elementos del espacio de enteros bidimensional \mathbb{Z}^2 , donde cada elemento es un punto de coordenadas (x, y) de un píxel blanco (o negro, según el convenio) en la imagen. Mientras que en imágenes en escala de grises se puede representar como conjuntos cuyas componentes están en \mathbb{Z}^3 , donde dos componentes de cada elemento se refieren a las coordenadas de un píxel y la tercera corresponde al valor discreto de la intensidad.

A continuación se definen algunos conceptos básicos que serán útiles a lo largo de esta sección:

- La **reflexión** de un conjunto B se define como

$$\hat{B} = \{w | w = -b, b \in B\}$$

- La **traslación** de un conjunto B por el punto $z = (z_1, z_2)$ se define como

$$(B)_z = \{c | c = b + z, b \in B\}$$

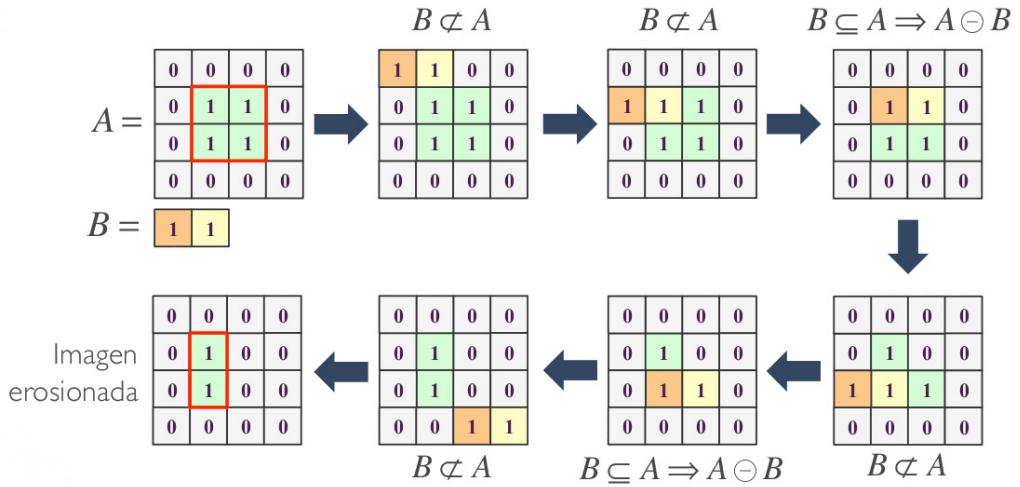
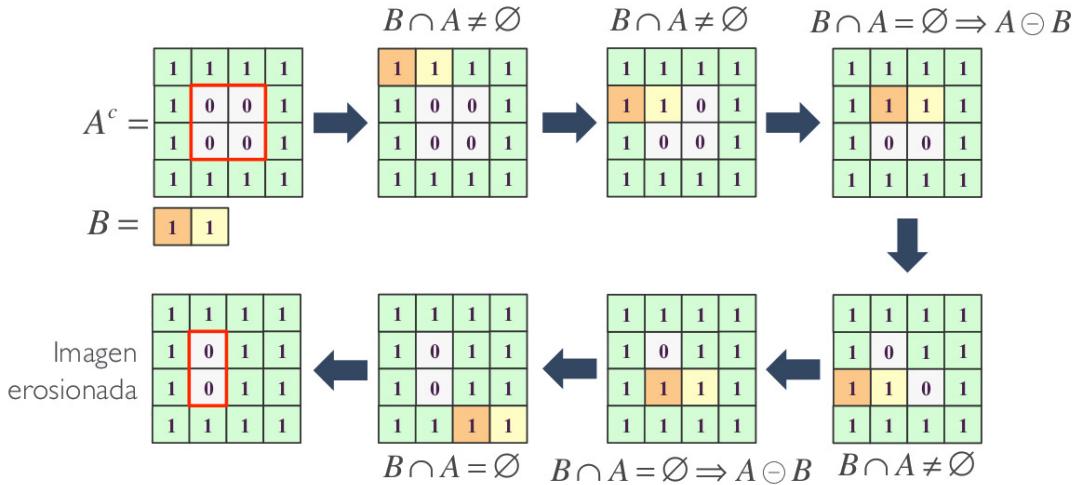
Los conjuntos de reflexión y traslación son empleados normalmente en morfología para formular operaciones basadas en los llamados **elementos estructurales**, que son pequeños conjuntos o subimágenes cuya distribución sirve de máscara para la aplicación de la especificación matemática del operador morfológico.

Erosión y dilatación

Binaria

La **erosión** de una conjunto A con un elemento estructurante B se define como

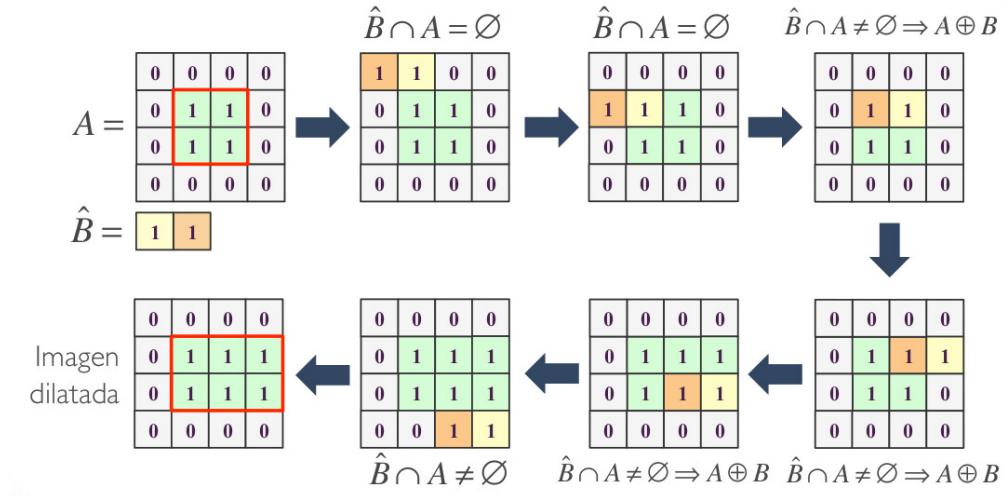
$$A \ominus B = \{z | (B)_z \subseteq A\} = \{z | (B)_z \cap A^c = \emptyset\}$$

Figura 3.4: Ejemplo erosión de una imagen binaria - $(B)_z \subseteq A$.Figura 3.5: Ejemplo erosión de una imagen binaria - $(B)_z \cap A^c = \emptyset$.²

La **dilatación** de una conjunto A con un elemento estructurante B se define como

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} = \{z \mid [(\hat{B})_z \cap A] \subseteq A\}$$

²Imágenes extraídas de [25].

Figura 3.6: Ejemplo dilatación de una imagen binaria.³

Teorema de dualidad: La erosión y la dilatación son duales una de la otra con respecto al conjunto complementario y la reflexión. Esto es,

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

$$(A \oplus B)^c = A^c \ominus \hat{B}$$

Demostración: Partiendo de la definición de erosión, tenemos que $(A \ominus B)^c = \{z \mid (B)_z \subseteq A\}^c$.

Si el conjunto $(B)_z$ está contenido en A , entonces $(B)_z \cap A^c = \emptyset$. Por lo tanto,

$$(A \ominus B)^c = \{z \mid (B)_z \cap A^c = \emptyset\}$$

Pero el complemento del conjunto de z 's que satisfacen $(B)_z \cap A^c = \emptyset$ es el conjunto de z 's tales que $(B)_z \cap A^c \neq \emptyset$. En consecuencia,

$$\begin{aligned} (A \ominus B)^c &= \{z \mid (B)_z \cap A^c \neq \emptyset\} \\ &= A^c \oplus \hat{B} \end{aligned}$$

Análogamente se prueba que $(A \oplus B)^c = A^c \ominus \hat{B}$.

■

La primera ecuación del teorema de dualidad nos indica que la erosión de A por B es el complemento de una dilatación de A^c por \hat{B} , y viceversa. La propiedad de la dualidad es útil particularmente cuando el elemento estructurante es simétrico con respecto de su origen. En este caso, podemos

³Imagen extraída de [25].

obtener la erosión de una imagen por B simplemente dilatando su fondo (por ejemplo, dilatando A^c) con el mismo elemento estructurante y aplicando el complemento al resultado. Dicho razonamiento se puede aplicar de manera similar a la segunda ecuación.

En escala de intensidades

La **erosión** de una imagen $f(x, y)$ en escala de grises con un elemento estructurante $b(x, y)$ se define como el valor mínimo de la imagen en la región coincidente con b cuando su origen está sobre el píxel (x, y) .

$$[f \ominus b](x, y) = \min_{(s,t) \in b} f(x + s, y + t)$$

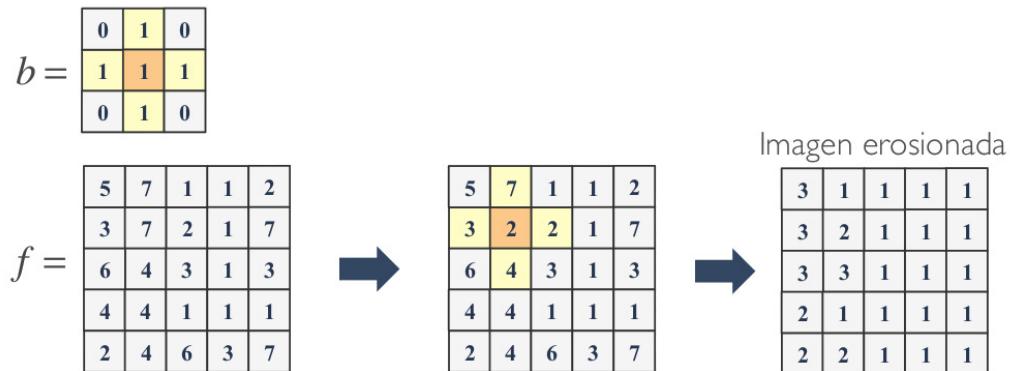


Figura 3.7: Ejemplo erosión de una imagen en escala de grises.⁴

La **dilatación** de una imagen $f(x, y)$ en escala de grises con un elemento estructurante $b(x, y)$ se define como el valor máximo de la región que cubre \hat{b} cuando su origen está sobre el píxel (x, y) .

$$[f \oplus b](x, y) = \max_{(s,t) \in b} f(x - s, y - t)$$

⁴Imagen extraída de [25].

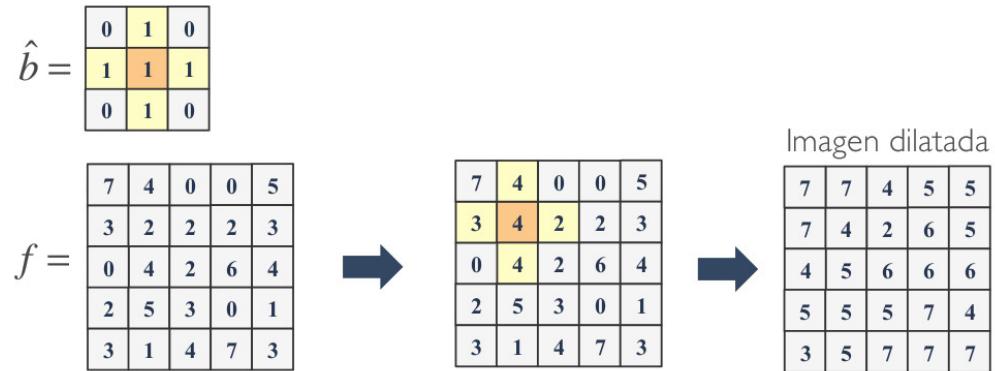


Figura 3.8: Ejemplo dilatación de una imagen en escala de grises.⁵

Como en el caso binario, la erosión y la dilatación son duales con respecto de la función de complemento y reflexión. Esto es,

$$(f \ominus b)^c(x, y) = (f^c \oplus \hat{b})(x, y)$$

donde $f^c = -f(x, y)$ y $\hat{b} = b(-x, -y)$. Por notación se suele omitir los argumentos de todas las funciones, es decir, se escribe la ecuación de la siguiente forma:

$$(f \ominus b)^c = (f^c \oplus \hat{b})$$

Análogamente,

$$(f \oplus b)^c = (f^c \ominus \hat{b})$$

Apertura y cierre

Binaria

La **apertura** de una conjunto A con un elemento estructurante B se define como

$$A \circ B = (A \ominus B) \oplus B$$

Es decir, es la erosión de A con B , seguido de una dilatación del resultado con B .

El **cierre** de una conjunto A con un elemento estructurante B se define como

$$A \bullet B = (A \oplus B) \ominus B$$

Es decir, es la dilatación de A con B , seguido de una erosión del resultado con B .

⁵Imagen extraída de [25].

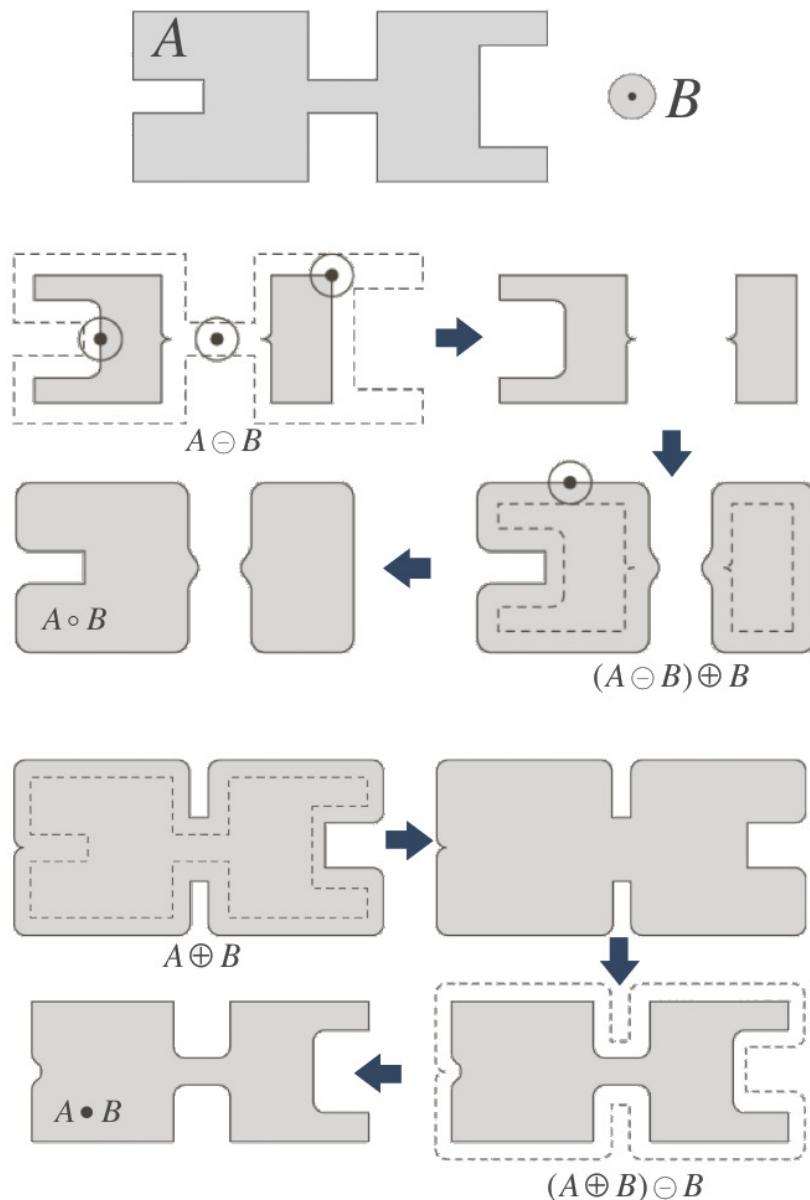


Figura 3.9: Ejemplo apertura y cierre de una imagen binaria.⁶

Las operaciones de apertura y cierre son duales una de la otra con respecto al conjunto complemento y reflexión. Estos es,

$$(A \circ B)^c = (A^c \bullet \hat{B})$$

$$(A \bullet B)^c = (A^c \circ \hat{B})$$

⁶Imagen extraída de [6].

En escala de intensidades

La **apertura** de una imagen f en escala de grises por un elemento estructurante b se define como

$$[f \circ b](x, y) = (f \ominus b) \oplus b$$

El **cierre** de una imagen f en escala de grises por un elemento estructurante b se define como

$$[f \bullet b](x, y) = (f \oplus b) \ominus b$$

La operación de apertura y cierre para escala de grises son duales con respecto al complemento y la reflexión del elemento estructurante:

$$(f \circ b)^c = f^c \bullet \hat{b}$$

$$(f \bullet b)^c = f^c \circ \hat{b}$$

Como $f^c = -f(x, y)$, podemos escribir las ecuaciones anteriores de la siguiente forma

$$-(f \circ b) = -f \bullet \hat{b}$$

$$-(f \bullet b) = -f \circ \hat{b}$$

Algoritmos

A partir de los conceptos dados anteriormente se definen algunos algoritmos que serán de utilidad.

- **Rellenado de huecos**

Sea A un conjunto cuyos elementos están 8-conectados encerrando la región que se desea llenar.

Inicialmente se crea una matriz, X_0 , de ceros del mismo tamaño que la contenida en A , excepto en la posición en X_0 correspondiente al punto dado en cada hueco, al cual se le asigna el valor 1. Entonces, el siguiente procedimiento rellena todas las regiones con unos:

$$X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, 3\dots$$

El algoritmo termina cuando $X_k = X_{k+1}$, contenido el conjunto X_k todos los huecos llenos. Por tanto, el conjunto solución es la unión de dichos huecos llenos y sus límites, $X_k \cup A$.

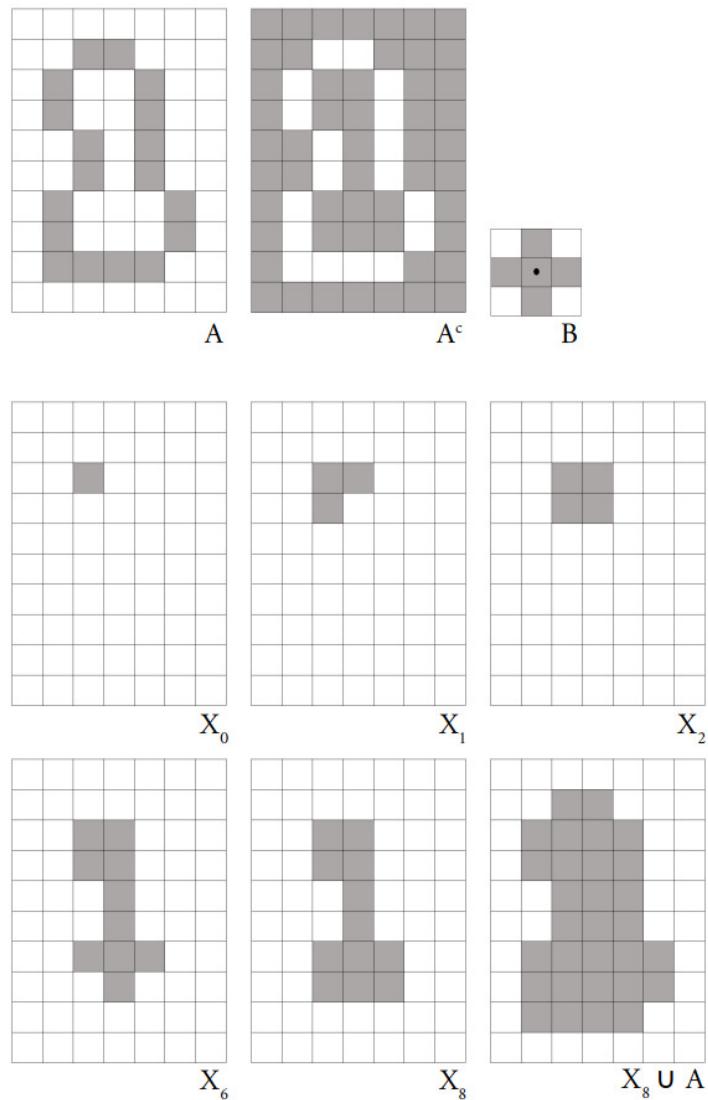


Figura 3.10: Ejemplo de relleno de huecos.⁷

■ Extracción de componentes conectadas

La extracción de componentes conexas, que son las regiones conectadas de píxeles, está relacionada con la localización de los objetos dentro de la imagen en binario.

Sea A un conjunto que contiene una o más componentes conectadas.

Inicialmente se crea una matriz, X_0 , de ceros del mismo tamaño que la

⁷Imagen construida a partir de [6].

contenida en A , excepto en la posición en X_0 correspondiente al punto dado en cada componente conectada en A , al cual se le asigna el valor 1. Entonces, el siguiente procedimiento iterativo intenta encontrar todas las componentes conexas:

$$X_k = (X_{k-1} \oplus B) \cup A \quad k = 1, 2, 3, \dots$$

El algoritmo termina cuando $X_k = X_{k-1}$, con X_k conteniendo todas las componentes conectadas.

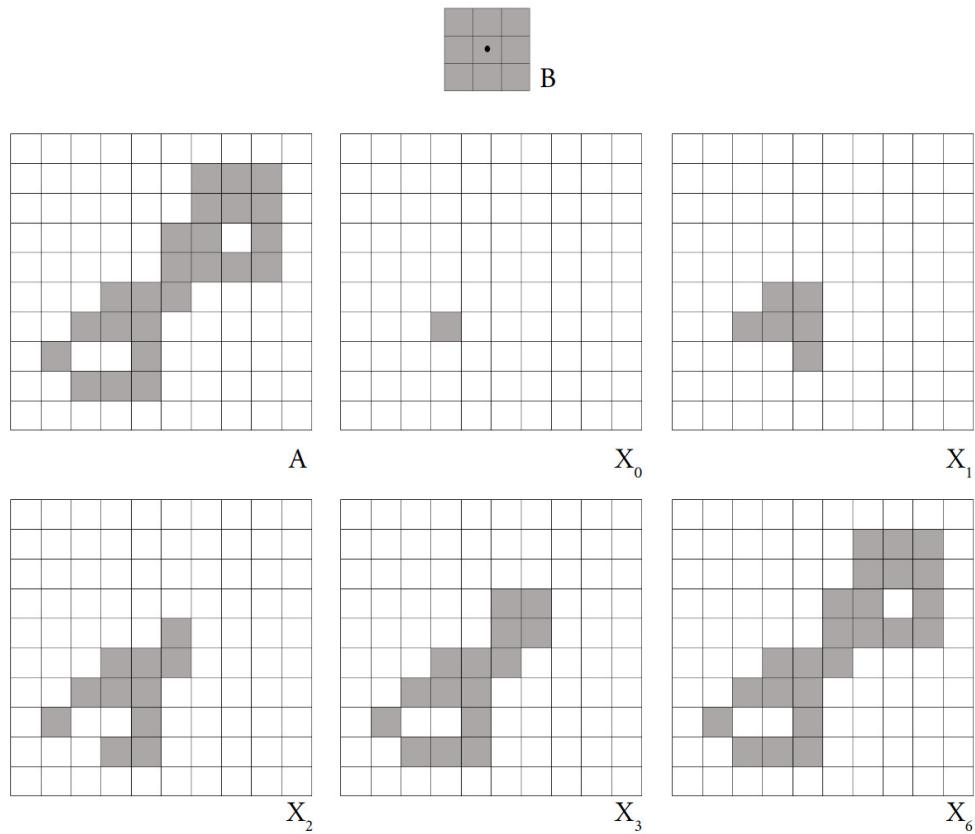


Figura 3.11: Ejemplo de extracción de componentes.⁸

■ Gradiente morfológico

La dilatación y la erosión pueden ser usadas de forma combinada mediante subtracción para obtener el **gradiente morfológico** de una imagen, el cual se define como

$$g = (f \oplus b) - (f \ominus b)$$

⁸Imagen construida a partir de [6].

El gradiente morfológico resalta las transiciones bruscas entre los niveles de grises de la imagen. Por lo tanto, se obtiene una imagen en la cual los bordes se resaltan y las áreas homogéneas se suprimen, produciendo un efecto gradiente.

A diferencia de algunos operadores de extracción de bordes como puede ser el de Sobel, los gradientes morfológicos utilizando elementos estructurales simétricos tienden a verse menos influenciados por la dirección de los bordes, pero se necesitan más requisitos computacionales.

■ Trasformación Top-Hat

La **transformación Top-Hat** de una imagen f en escala de grises se define como

$$h = f - (f \circ b)$$

Esta transformación es útil para el realzado de detalles ante la presencia de sombras y, además, corrige los efectos de la iluminación no uniforme.

3.1.6. Transformada de Hough circular

La transformada de Hough es un método de extracción de características para la detección de bordes en imágenes parametrizadas, es decir, los objetos que se desean detectar se representan por medio de una ecuación conocida.

Originalmente fue diseñada para la extracción de líneas rectas, sin embargo, se ha extendido para identificar otras formas como círculos o elipses. Pudiéndose aplicar a cualquier función de la forma $g(v, c) = 0$, donde v es un vector de coordenadas y c es un vector de coeficientes.

Si un círculo se describe como

$$(x - a)^2 + (y - b)^2 = r^2$$

donde (a, b) son las coordenadas del centro y r es su radio, entonces un punto arbitrario del borde (x_i, y_i) será transformado en un cono circular en el espacio de parámetros (a, b, r) .

El espacio de parámetros para el círculo pertenece a \mathbb{R}^3 pero se puede simplificar la representación del círculo considerando el radio constante. Por lo tanto, si consideramos que conocemos de antemano el radio del círculo que deseamos detectar, el espacio de parámetros se reduce a \mathbb{R}^2 .

La transformada de Hough circular no es un algoritmo rigurosamente especificado, sino que hay varios enfoques diferentes que pueden ser tomados

para su implementación. Sin embargo, en general, hay tres pasos esenciales que son comunes a todos:

1. Cálculo de la matriz acumuladora

Los píxeles del primer plano con un gradiente alto son elegidos para ser candidatos y se les permite emitir “votos” en la matriz acumuladora.

2. Estimación del centro

Los votos de los píxeles candidatos pertenecientes a un círculo tienden a acumularse en el intervalo de la matriz acumuladora correspondiente al centro del círculo. Por lo tanto, los centros de los círculos se estiman detectando los picos en la matriz acumuladora.

3. Estimación de los radios

Si la misma matriz acumuladora se utiliza para varios radios, los radios de los círculos detectados se estiman en un paso distinto.

En este caso, se ha utilizado el método denominado *two-stage*, en el cual el problema de búsqueda de círculos se descompone en dos etapas y los radios se calculan explícitamente utilizando los centros de los círculos estimados junto con la información de la imagen.

Two stage

Como el centro de un círculo debe estar en la normal de cada punto del círculo, el punto común de intersección de estas normales es en realidad el centro del círculo. Para acumular los votos a lo largo de la normal de cada punto del borde, se utiliza una matriz bidimensional. Por otro lado, para identificar el radio de los círculos, se calcula la distancia de cada punto desde un centro candidato y se produce un histograma del radio.

La detección de picos falsos en la etapa de búsqueda central puede conducir a un costo computacional significativo para la segunda etapa, especialmente si se usa un umbral bajo para detectar círculos pequeños. Sin embargo, este método es robusto en presencia de ruido, oclusión y variación de la iluminación.

3.1.7. Umbralización

Una vez se ha obtenido el valor umbral U mediante algún método determinando o fijándolo arbitrariamente, la imagen de entrada que denotaremos por $f(x, y)$ es segmentada de la siguiente forma:

$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) > U \\ 0 & \text{si } f(x, y) \leq U \end{cases}$$

para $x = 0, 1, 2, \dots, M - 1$ e $y = 0, 1, 2, \dots, N - 1$. Obteniendo la imagen binaria $g(x, y)$.

Método de Otsu

El método de Otsu es un método de umbralización global que permite diferenciar un objeto del fondo de la imagen mediante la binerización, empleando técnicas estadísticas. En concreto, se utiliza la varianza, que es una medida de la dispersión de valores.

Con este método se pretende calcular el valor umbral de manera que la dispersión dentro de cada clase sea lo más pequeña posible, pero al mismo tiempo lo más alta posible entre clases diferentes.

Partiendo de una imagen en escala de grises de tamaño $M \times N$ píxeles y L posibles niveles diferentes.

- 1) Se calcula el histograma normalizado de la imagen. Denotando a las componentes del histograma por p_i , $i = 0, 1, \dots, L - 1$, que es la probabilidad de que ocurra el nivel de intensidad i . Estas se obteniendo a partir de la siguiente fórmula:

$$p_i = \frac{n_i}{MN}$$

donde n_i es el número de píxeles con nivel de intensidad i .

- 2) Calcular las sumas acumulativas, $P_1(k)$ para $k = 0, 1, \dots, L - 1$:

$$P_1(k) = \sum_{i=0}^k p_i$$

Si se fijase un valor umbral T en el nivel de intensidad k , se dividiría el conjunto de píxeles de la imagen en dos clases:

- C_1 = píxeles con nivel de intensidad en el intervalo $[0, k]$.
- C_2 = píxeles con nivel de intensidad en $[k + 1, L - 1]$.

Por lo tanto, podemos decir que $P_1(k)$ indica la probabilidad de que un píxel pertenezca a la clase C_1 .

- 3) Se calcula las medias acumulativas, $m(k)$, para $k = 0, 1, \dots, L - 1$:

$$m(k) = \sum_{i=0}^k ip_i$$

- 4) Se calcula la media global, m_G :

$$m_G = \sum_{i=0}^{L-1} ip_i$$

- 5) Se calcula la varianza entre clases, $\sigma_B^2(k)$, para $k = 0, 1, \dots, L - 1$:

$$\sigma_B^2(k) = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}$$

El valor $\sigma_B^2(k)$ mide la dispersión entre los valores de intensidad de las clases C_1 y C_2 , definidas al tomar como valor umbral el valor de intensidad k .

- 6) Se obtiene el valor umbral, k^* :

$$\sigma_B^2(k^*) = \max_{0 \leq k \leq L-1} \sigma_B^2(k)$$

Si el máximo no es único, se obtiene k^* como la media de los valores de k correspondientes a los máximos detectados.

- 7) Se obtiene la medida de separabilidad, η^* , como el cociente entre la varianza de clases y la global:

$$\eta^* = \frac{\sigma_B^2(k^*)}{\sigma_G^2}$$

Método de entropía máxima

La entropía es una medida del grado de incertidumbre de una variable aleatoria.

Sea X una variable aleatoria discreta definida en un espacio de probabilidad $(\Omega, \mathcal{U}, \mathcal{P})$ y asumiendo valores x_1, x_2, \dots , con distribución de probabilidad $\{p_k : 1, 2, \dots\}$, donde $p_k = \mathcal{P}(X == x_k)$, entonces se define la **entropía** como

$$H(X) = - \sum_{k=1}^{\infty} p_k \log(p_k)$$

Se asume como regla general que $0 \log(0) = 0$. La base del logaritmo puede ser cualquier número positivo pero como regla general se utilizan los logaritmos en base 2 o el neperiano, los cuales corresponden a la elección de un bit o una unidad natural como unidad de medida.

Dada una imagen de tamaño $N \times M$, m niveles de gris y sea n_i la frecuencia de ocurrencia de cada nivel de intensidad. Entonces la probabilidad de ocurrencia de cada nivel de gris viene dada por:

$$p_i = \frac{n_i}{NM}$$

Supongamos que tenemos dos distribuciones de probabilidad A , del fondo, y B , del objeto. Y denotamos por t al umbral de binarización. entonces la función de entropía de la distribución A , $H(A)$, y de la distribución B , $H(B)$, están definida como:

$$H(A) = - \sum_{i=1}^t \frac{p_i}{p_t} \log \left(\frac{p_i}{p_t} \right)$$

$$H(B) = - \sum_{i=t+1}^m \frac{p_i}{p_m} \log \left(\frac{p_i}{p_m} \right)$$

donde:

$$p_t = \sum_{i=1}^t p_i$$

$$p_m = 1 - p_t$$

Por lo tanto, la entropía total de la imagen se define como la suma de las funciones de entropía de A y B. Y el umbral óptimo está dado por el valor t que maxima dicha suma.

3.1.8. *Gray Level Co-occurrence Matrix* (GLCM)

Textura significa repetir patrones de variación local de las intensidades de píxeles. Proporciona información sobre la disposición de los píxeles de la superficie y su relación con los píxeles circundantes. En este caso, el análisis estadístico de textura se basará en la matriz de co-ocurrencia de niveles de gris (GLCM).

Para un imagen bidimensional $f(x, y)$ con N niveles de gris, se define la **matriz de co-ocurrencia de niveles de gris** para cada d y ϕ como:

$$P(d, \phi) = \begin{pmatrix} p_{0,0} & p_{0,1} & \dots & p_{0,N-1} \\ p_{1,0} & p_{1,1} & \dots & p_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N-1,0} & p_{N-1,1} & \dots & p_{N-1,N-1} \end{pmatrix}$$

donde

$$p_{i,j} = \frac{\text{número de parejas de píxeles con intensidad } (i, j)}{\text{número total de parejas consideradas}}$$

$p_{i,j}$ se define como el número relativo de veces que la pareja de niveles de gris (i, j) ocurre cuando los píxeles separados por la distancia d a lo largo del ángulo ϕ son comparados.

Las características de textura comúnmente extraídas de GLCM son las siguientes:

- **Contraste:** Medida del contraste de la intensidad entre un píxel y su vecindario, y viene dado por:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (i - j)^2 p_{ij}$$

- **Homogeneidad:** Mide la cercanía de la distribución de los elementos en el GLCM a su diagonal y puede ser escrito matemáticamente como:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{p_{ij}}{1 + |i - j|}$$

- **Correlación:** Calcula la dependencia lineal de los valores de nivel de gris en la matriz de co-ocurrencia. Se representa matemáticamente como:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{(i - \mu_i)(j - \mu_j)p_{ij}}{\sigma_i \sigma_j}$$

donde μ_i , μ_j , σ_i y σ_j son las medias y las desviaciones típicas de p_i y p_j , respectivamente.

- **Energía:** Mide la uniformidad en el intervalo $[0, 1]$. Esta característica vale 1 en el caso de que los niveles de gris de la imagen sean constantes y se puede escribir matemáticamente como:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_{ij}^2$$

Generalmente, para reducir el tiempo de cómputo de la matriz de co-ocurrencia y evitar matrices dispersas, el número de niveles de cuantización de la imagen se reduce, por ejemplo de 256 a 8 niveles de intensidad.

3.1.9. *Support Vector Machines (SVM)*

Las máquinas de Vectores de Soporte (SVM) son sistemas de clasificación binarios, lo que nos permite distinguir entre dos clases. Además SVM es un clasificador lineal, es decir, la frontera de decisión es un hiperplano W .

Este hiperplano W se obtiene de manera que se maximice lo que se denomina el margén entre los ejemplos positivos y los ejemplos negativos, es decir, maximizamos la distancia entre aquellos ejemplos positivos y negativos que están más cercanos entre sí, denominados **vectores de soporte**. El hiperplano que se encuentra intenta separar al máximo estos ejemplos de la frontera. Por lo tanto, la frontera de decisión se centra en poder separar aquellos casos más cercanos entre sí, que son los más difíciles.

Si los datos no son linealmente separables, SVM nos permite aplicar una técnica que se conoce como *kernel trick*, la cual nos posibilita transformar el espacio de característica original en otro espacio de características diferente, y que es linealmente separable, de forma que podremos encontrar la frontera en este nuevo espacio pero sin necesidad de calcular esta transformación de forma explícita.

Dado un conjunto separable $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$, donde $x_i \in \mathbb{R}^d$ e $y_i \in \{+1, -1\}$, se puede definir el hiperplano de separación como una función lineal que es capaz de separar dicho conjunto sin error:

$$D(x) = (w_1x_1 + \dots + w_dx_d) + b = \langle w, x \rangle + b$$

donde w y b son coeficientes reales. Este hiperplano solución verificará las siguientes restricciones para todo x_i del conjunto:

$$\begin{aligned} \langle w, x_i \rangle + b &\geq 0 & \text{si } y_i = +1, & \quad (i = 1, \dots, n) \\ \langle w, x_i \rangle + b &\leq 0 & \text{si } y_i = -1, & \end{aligned}$$

De esta forma, la condición de clasificación va a implicar que cuando aplicamos los parámetros del hiperplano w a las muestras positivas, nos va a dar un valor mayor que $+1$; mientras que cuando lo aplicamos a las muestras negativas, dicho valor será inferior a -1 .

Estas dos condiciones se pueden expresar conjuntamente:

$$y_i D(x_i) = y_i (\langle w, x_i \rangle + b) \geq 0 \quad (i = 1, \dots, n)$$

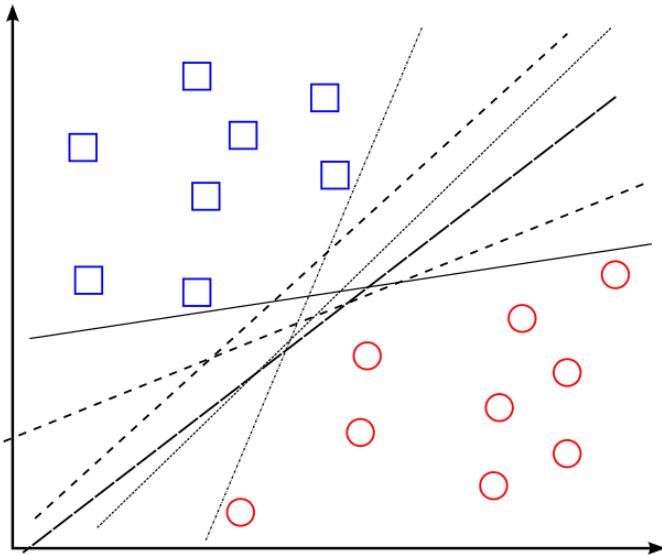


Figura 3.12: (SVM) Algunos hiperplanos solución ⁹

El hiperplano que permite separar los ejemplos no es único, existen infinitos hiperplanos separables, pero a nosotros nos interesa buscar uno que sea óptimo. Para ello debemos definir el concepto de **margen** de un hiperplano de separación, denotado por τ , como la mínima distancia entre dicho hiperplano y las muestras más cercanas de cualquiera de las dos clases. A partir de esta definición, un **hiperplano** de separación se denomina **óptimo** si su margen es de tamaño máximo.

Propiedad: *El hiperplano de separación óptimo equidista del ejemplo más cercano de cada clase.*

Demostración: Se realiza por reducción al absurdo. Supongamos que la distancia del hiperplano óptimo a la muestra más cercana de la clase +1 fuese menor que la correspondiente a la del ejemplo más cercano de la clase -1. Esto quiere decir que se puede alejar el hiperplano de la muestra de la clase +1 de forma que la distancia del hiperplano solución a dicho ejemplo sea mayor que antes y, a su vez, siga siendo menor que la distancia a la muestra más cercana de la clase -1. Contradicción ya que se puede aumentar el tamaño del margen cuando habíamos supuesto que este era máximo.

Análogamente se haría en el caso de que supusiesemos que la distancia del hiperplano óptimo al ejemplo más cercano de la clase -1 fuese menor que la correspondiente al ejemplo más cercano de la clase +1.

⁹Estas imágenes han sido extraídas de [20].

■

La distancia entre un hiperplano de separación $D(x)$ y un ejemplo x' viene dada por

$$\frac{|D(x')|}{\|w\|}$$

siendo w el vector, que junto con el parámetro b , define el hiperplano $D(x)$ y es perpendicular al hiperplano considerado. Por lo tanto, haciendo uso de las expresiones anteriores, obtenemos que todos los ejemplos de entrenamiento verifican:

$$\frac{y_i D(x_i)}{\|w\|} \geq \tau, \quad i = 1, \dots, n$$

Pudiéndose expresar también como:

$$y_i D(x_i) \geq \tau \|w\|, \quad i = 1, \dots, n$$

Se deduce por tanto que encontrar el hiperplano óptimo es equivalente a encontrar el valor de w que maximiza el margen, existiendo infinitas soluciones que difieren solo en la escala de w . Por lo tanto, para limitar el número de soluciones a una sola, la escala del producto de τ y la norma de w se fija, de forma arbitraria, a la unidad:

$$\tau \|w\| = 1$$

Y tenemos, por tanto, que:

$$\tau = \frac{1}{\|w\|}$$

En consecuencia, aumentar el margen equivale disminuir la norma de w .

Por tanto, un hiperplano de separación óptimo verificará:

$$y_i D(x_i) = y_i (\langle w, x_i \rangle + b) \leq 1, \quad i = 1, \dots, n$$

Los vectores de soporte serán aquellos ejemplos que verifiquen la igualdad en la restricción anterior, están situados a ambos lados del hiperplano óptico y definen el margen. Estos deberían ser los únicos ejemplos que se considerasen a la hora de construir dicho hiperplano ya que son los más cercanos al hiperplano de separación y, por tanto, los más difíciles de clasificar.

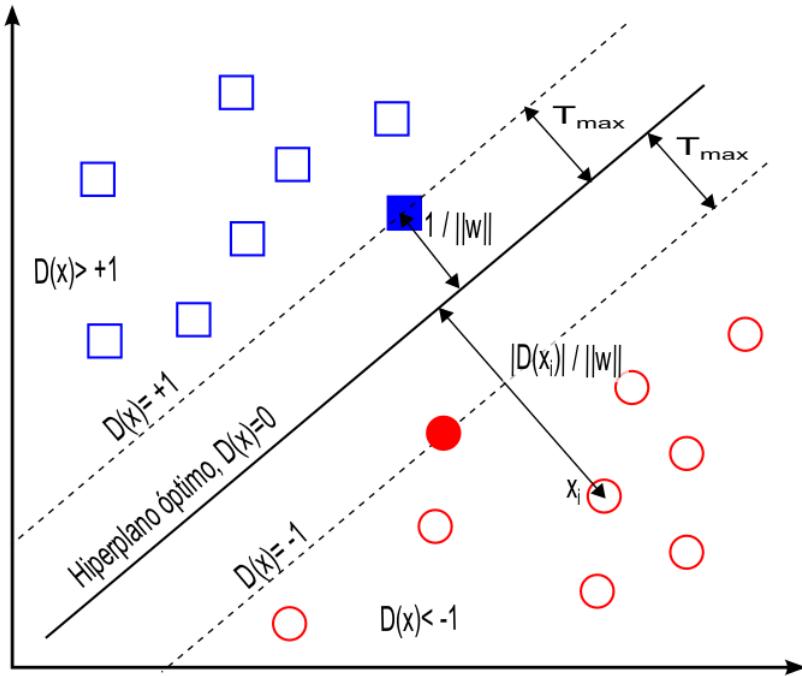


Figura 3.13: (SVM) Hiperplano de separación óptimo.

La búsqueda del hiperplano óptimo para el caso separable puede formalizarse como sigue:

$$\min_{w,b} f(w) = \|w\|$$

$$\text{s. a. } y_i(\langle w, x_i \rangle + b) - 1 \leq 0, \quad i = 1, \dots, n$$

Obsérvese que es equivalente minimizar $f(w) = \|w\|$ o el funcional $f(w) = \frac{1}{2}\|w\|^2$, pero este último permitirá simplificar la notación posterior, obteniendo expresiones más compactas. Por lo tanto, la búsqueda del hiperplano óptimo es equivalente al problema siguiente:

$$\min_{w,b} f(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2}\langle w, w \rangle$$

$$\text{s. a. } y_i(\langle w, x_i \rangle + b) - 1 \leq 0, \quad i = 1, \dots, n$$

Este problema de optimización con restricciones corresponde a un problema de programación cuadrático y es abordable mediante la teoría de la optimización¹⁰. Esta teoría establece que un problema de optimización,

¹⁰En el anexo primero se muestra, de forma reducida, los resultados de la teoría de la optimización necesarios para solucionar los diferentes problemas de optimización relacionados con los problemas de clasificación mediante SVM.

denominado primal, tiene una forma dual si la función a optimizar y las restricciones con funciones estrictamente convexas. Resolver el problema dual permite obtener la solución del problema primal.

Puede demostrarse que el problema de optimización anterior satisface el criterio de convexidad y, por tanto, tiene un dual.

Para la resolución del problema planteado nos ayudaremos del lagrangiano:

$$L(w, b, \alpha) = \frac{1}{2} \langle w, w \rangle - \sum_{i=1}^n \alpha_i [y_i (\langle w, x_i \rangle + b) - 1]$$

El signo menos del segundo sumando es debido a que las restricciones son del tipo $g(x) \geq 0$ en lugar de $g(x) \leq 0$.

Para la minimización de esta expresión se utilizan las condiciones de Karush-Kuhn-Tucker (KKT):

$$\begin{aligned} \frac{\partial L(w^*, b^*, \alpha)}{\partial w} &\equiv w^* - \sum_{i=1}^n \alpha_i y_i x_i = 0, \Rightarrow w^* = \sum_{i=1}^n \alpha_i y_i x_i, \quad i = 1, \dots, n \\ \frac{\partial L(w^*, b^*, \alpha)}{\partial b} &\equiv - \sum_{i=1}^n \alpha_i y_i = 0, \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0, \quad i = 1, \dots, n \\ \alpha_i [1 - y_i (\langle w^*, x_i \rangle + b^*)] &= 0, \quad i = 1, \dots, n \end{aligned}$$

A partir de la primera derivada parcial con respecto a w vamos a obtener el valor óptimo de w como combinación lineal de las muestras de entrenamiento. Bajo condiciones de convexidad podemos incluir esta solución de w en la fórmula del lagrangiano:

$$\begin{aligned} L(w, b, \alpha) &= \frac{1}{2} \left(\sum_i \alpha_i y_i x_i \right)^T \left(\sum_j \alpha_j y_j x_j \right) - \sum_i \alpha_i \left[y_i \left(\left(\sum_j \alpha_j y_j x_j \right)^T x_i + b \right) - 1 \right] \\ &= \frac{1}{2} \left(\sum_i \alpha_i y_i x_i \right)^T \left(\sum_j \alpha_j y_j x_j \right) - \sum_i \alpha_i y_i \left(\sum_j \alpha_j y_j x_j \right)^T x_i - b \sum_i \alpha_i y_i + \sum_i \alpha_i \\ &= \frac{1}{2} \left(\sum_i \alpha_i y_i x_i \right)^T \left(\sum_j \alpha_j y_j x_j \right) - \sum_i \alpha_i y_i \left(\sum_j \alpha_j y_j x_j \right)^T x_i + \sum_i \alpha_i \\ &= \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \\ &= -\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i \end{aligned}$$

Con esto hemos transformado el problema de minimización primal en su dual, consiguiendo una nueva función L cuya maximización sujeta a la condición que hemos obtenido, respecto al parámetro b , nos va a dar la solución final.

$$\begin{aligned} \max_{\alpha} L(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{s. a. } &\begin{cases} \sum_{i=1}^n \alpha_i y_i = 0, & i = 1, \dots, n \\ \alpha_i \geq 0, & \end{cases} \end{aligned}$$

Tanto el problema primal como este es abordable mediante técnicas estándar de programación cuadrática. Sin embargo, el coste computacional asociado a su resolución es factible incluso para problemas con un número muy alto de dimensiones ya que el problema de optimización dual escala con el número de muestras n , a diferencia del primal que lo hace con la dimensionalidad d .

La solución del problema dual permite obtener la solución del primal:

$$D(x) = \sum_{i=1}^n \alpha_i^* y_i \langle x, x_i \rangle + b^*$$

Tenemos además que si $\alpha_i > 0$ entonces el segundo factor de la parte izquierda de dicha expresión tendrá que ser cero y, por tanto

$$y_i (\langle w^*, x_i \rangle + b^*) = 1$$

es decir, el correspondiente ejemplo (x_i, y_i) , satisface la correspondiente restricción del problema primal, pero considerando el caso de igualdad que, por definición, estos son los vectores de soporte. En consecuencia, podemos afirmar que los ejemplos que tengan asociado un $\alpha_i > 0$ serán vectores de soporte. Se puede afirmar, además, que el hiperplano de separación $D(x)$ se construirá como una combinación lineal de sólo los vectores de soporte del conjunto de ejemplos, ya que el resto de muestras tendrán asociado un $\alpha_j = 0$.

A partir de la expresión anterior obtenemos el valor de b^* :

$$b^* = y_{vs} - \langle w^*, x_{vs} \rangle$$

donde (x_{vs}, y_{vs}) representa la tupla de cualquier vector de soporte junto con su valor de clase, es decir, la tupla de cualquier ejemplo que tenga asociado un α_i distinto de cero. La expresión anterior se transforma en:

$$b^* = \frac{1}{N_{vs}} \sum_{i=1}^{N_{vs}} (y_{vs} - \langle w^*, x_{vs} \rangle)$$

Finalmente, sustituyendo $w^* = \sum_{i=1}^n \alpha_i y_i x_i$ $i = 1, \dots, n$ en alguna de las expresiones anteriores, podemos calcular el valor de b^* en función de la solución del problema dual.

Cabe notar que la solución de SVM que hemos obtenido de esta forma es no paramétrica, lo que quiere decir que no vamos a tener necesidad de ajustar ningún parámetro durante el entrenamiento. Sin embargo, en el caso general cuando los datos no son linealmente separables, y por tanto tenemos que introducir tolerancia a errores de clasificación, sí que va a ser necesario ajustar algunos parámetros.

Para abordar este nuevo problema se introduce un conjunto de variables reales positivas, denominadas **variables de holgura**, ξ_i , $i = 1, \dots, n$, que suaviza la condición del margen e introduce tolerancia a errores de clasificación. Estas variables permiten cuantificar el número de ejemplos no separables que se está dispuesto a admitir, es decir:

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i, \quad \xi \geq 0, \quad i = 1, \dots, n$$

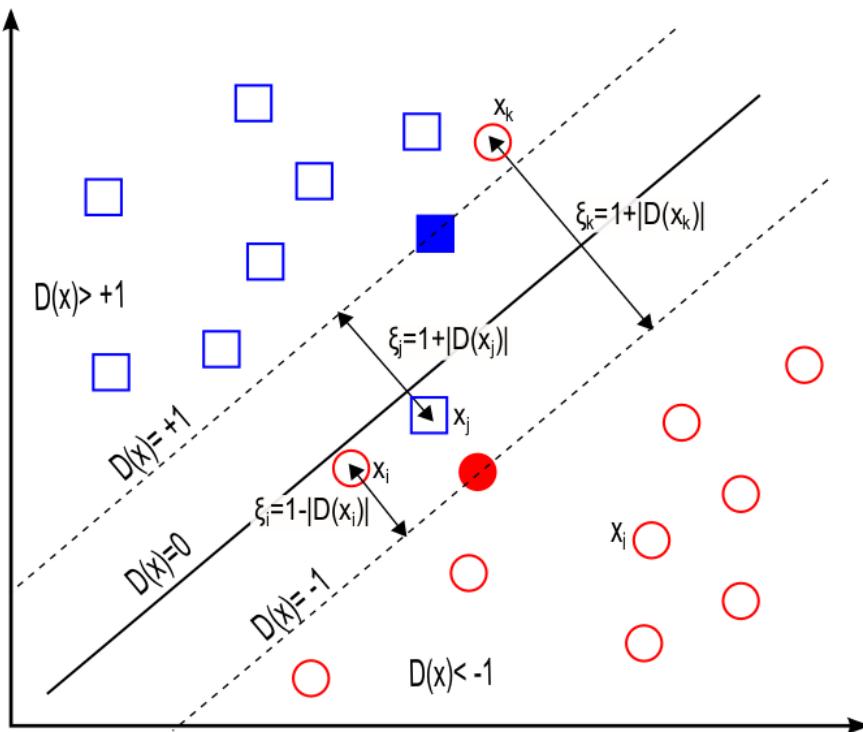


Figura 3.14: (SVM) Hiperplano solución de ejemplos cuasi-separables linealmente.

Por tanto, para un ejemplo (x_i, y_i) , su variable de holgura, ξ_i , representa la desviación del caso separables, medida desde el borde del margen que corresponde a la clase y_i . A partir de esta definición podemos distinguir varios casos:

- $\xi_i = 0$ si, y sólo si, (x_i, y_i) es separable.
- $\xi_i \geq 0$ si, y sólo si, (x_i, y_i) es no separable.
 - Si $\xi_i \geq 1$, entonces, además, el ejemplo está mal clasificado

Por lo tanto, la suma de todos las variables de holgura permite medir el coste asociado al número de ejemplos no separables. Claramente, cuanto mayor sea el valor de esta suma, mayor será el número de muestras no separables. Por lo que la función de optimización debe incluir los errores de clasificación cometiendo el hiperplano de separación, es decir:

$$f(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i$$

donde C es una constante, suficientemente grande y arbitraria. Por tanto, el impacto de las variables de holgura en la formulación queda modulado por el parámetro C , y este puede ser interpretado como un parámetro de regularización que nos permitirá controlar el equilibrio entre maximizar el margen y minimizar el error en las muestras de entrenamiento, es decir, nos permite regular el compromiso entre el grado de sobreajuste del clasificador final y la proporción del número de ejemplos no separables. Así, un valor de C muy grande permitiría valores de ξ_i muy pequeños.

En consecuencia, el nuevo problema de optimización viene dado por la siguiente formulación:

$$\begin{aligned} & \min \frac{1}{2} \langle w, w \rangle + C \sum_{i=1}^n \xi_i \\ \text{s. a. } & \begin{cases} y_i(\langle w, x_i \rangle + b) + \xi_i - 1 \geq 0, & i = 1, \dots, n \\ \xi_i \geq 0, & \end{cases} \end{aligned}$$

Este hiperplano recibe el nombre de **hiperplano de separación de margen blando**.

Como en el caso anterior, si el problema de optimización corresponde a un espacio de características de alta dimensionalidad, entonces puede ser transformado a su dual de forma análoga:

$$L(w, b\xi, \alpha, \beta) = \frac{1}{2} \langle w, w \rangle + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i [y_i(\langle w, x_i \rangle + b) + \xi_i - 1] - \sum_{i=1}^n \beta_i \xi_i$$

A partir de las derivadas parciales, aplicando las condiciones KKT, obtenemos:

$$\frac{\partial L}{\partial w} \equiv w^* - \sum_{i=1}^n \alpha_i y_i x_i = 0 \Rightarrow w^* = \sum_{i=1}^n \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} \equiv \sum_{i=1}^n \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} \equiv C - \alpha_i - \beta_i = 0 \Rightarrow C = \alpha_i + \beta_i$$

La primera nos permite establecer las relaciones entre las variables del problema primal (w, b, ξ) con las del problema dual (α, β), mientras que el resto establece restricciones adicionales de las variables duales. Además, obtenemos también las siguientes expresiones al aplicar las condiciones KKT:

$$\alpha_i [1 - y_i (\langle w^*, x_i \rangle + b^*) - \xi_i] = 0, \quad i = 1, \dots, n$$

$$\beta_i \cdot \xi_i = 0, \quad i = 1, \dots, n$$

Teniendo en cuenta que $C = \alpha_i + \beta_i$ y $\alpha_i, \beta_i \geq 0$, obtenemos que $0 \leq \alpha_i \leq C$.

Logrando finalmente la formalización buscada del problema dual.:

$$\max \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$\text{s. a. } \begin{cases} \sum_{i=1}^n \alpha_i y_i = 0, & i = 1, \dots, n \\ 0 \leq \alpha_i \leq C \end{cases}$$

Como en el caso anterior, la solución del problema dual nos permitirá expresar el hiperplano de separación óptima en términos de α^* :

$$D(x) = \sum_{i=1}^n \alpha_i^* y_i \langle x, x_i \rangle + b^*$$

Y es posible expresar b^* en términos de las variables duales:

$$b^* = y_i - \sum_{j=1}^n \alpha_i^* y_i \langle x_j, x_i \rangle \quad \forall \alpha_i \text{ t.q. } 0 < \alpha_i < C$$

donde los coeficientes α_i^* , $i = 1, \dots, n$ corresponden a la solución del problema dual.

Demostración: De la restricción $C = \alpha_i + \beta_i$, deducimos que si $\alpha_i = 0$, entonces $C = \beta_i$. Y como $\beta_i \cdot \xi_i = 0$, $\xi_i = 0$. Por tanto, se puede afirmar

que todos los ejemplos x_i cuyo α_i asociado sea igual a cero corresponde a ejemplos separables ($\xi_i = 0$).

Por otro lado, todo ejemplo no separable, x_i , se caracteriza por tener asociado un $\xi_i > 0$ y considerando que $\beta_i \cdot \xi_i = 0$, entonces $\beta_i = 0$. De la restricción $C = \alpha_i + \beta_i$, deducimos también que $C = \alpha_i$. Por tanto, se puede afirmar que todos los muestras x_i cuyo $\alpha_i = C$ corresponden a ejemplos no separables. Además, dado que, en este caso, $\alpha_i \neq 0$ y que $\alpha_i[1 - y_i(\langle w^*, x_i \rangle + b^*) - \xi_i] = 0$ se deduce que:

$$1 - y_i(\langle w^*, x_i \rangle + b^*) - \xi_i = 0$$

es decir,

$$1 - y_i D(x_i) = \xi_i$$

Distinguimos dos casos:

- Si el ejemplo x_i es no separable pero está bien clasificado, es decir, $y_i D(x_i) \geq 0$, entonces $\xi_i = 1 - |D(x_i)|$.
- Si el ejemplo x_i es no separable y, además, está mal clasificado, es decir, $y_i D(x_i) < 0$, entonces $\xi_i = 1 + |D(x_i)|$.

Por último nos queda estudiar el caso $0 < \alpha_i < C$. De $C = \alpha_i + \beta_i$ y $\beta_i \cdot \xi_i = 0$, obtenemos que $\beta_i \neq 0$ y $\xi_i = 0$. Además se deduce que:

$$1 - y_i(\langle w^*, x_i \rangle + b^*) = 0$$

Por tanto, podemos afirmar que un ejemplo, x_i , es vector de soporte si, y solo si, $0 < \alpha_i < C$.

De la expresión anterior obtenemos:

$$b^* = y_i - \langle w^*, x_i \rangle \quad \forall i \text{ t.q. } 0 < \alpha_i < C$$

Y sustituyendo $w^* = \sum_{i=1}^n \alpha_i y_i x_i$ en la expresión anterior obtenemos la deseada.

■

Hasta ahora se ha asumido que los ejemplos eran separables o quasi-separables y, por tanto, los hiperplanos se definían como funciones lineales. Sin embargo, a continuación, se va a estudiar el caso de clasificación binaria en que los ejemplos no sean separables linealmente. Para ellos, se describirá cómo usar de forma eficiente conjuntos de funciones base, no lineales, para definir espacios transformados de alta dimensionalidad y cómo buscar hiperplanos de separación óptimos en dichos espacios transformados. A cada

uno de estos espacios se le denomina **espacio de características**, para diferenciarlo del espacio de ejemplos de entrada.

Sea $\Phi : \mathbb{X} \rightarrow \mathcal{F}$ la función de transformación que hace corresponder cada vector x de entrada con un punto en el espacio de características \mathcal{F} , donde $\Phi(x) = [\phi_1(x), \dots, \phi_m(x)]$ y $\exists \phi_i(x)$, $i = 1, \dots, m$, tal que $\phi_i(x)$ es una función no lineal. La idea entonces es construir un hiperplano de separación lineal en este nuevo espacio. La frontera de decisión lineal obtenida en el espacio de características se transformará en una frontera de decisión no lineal en el espacio de entradas.

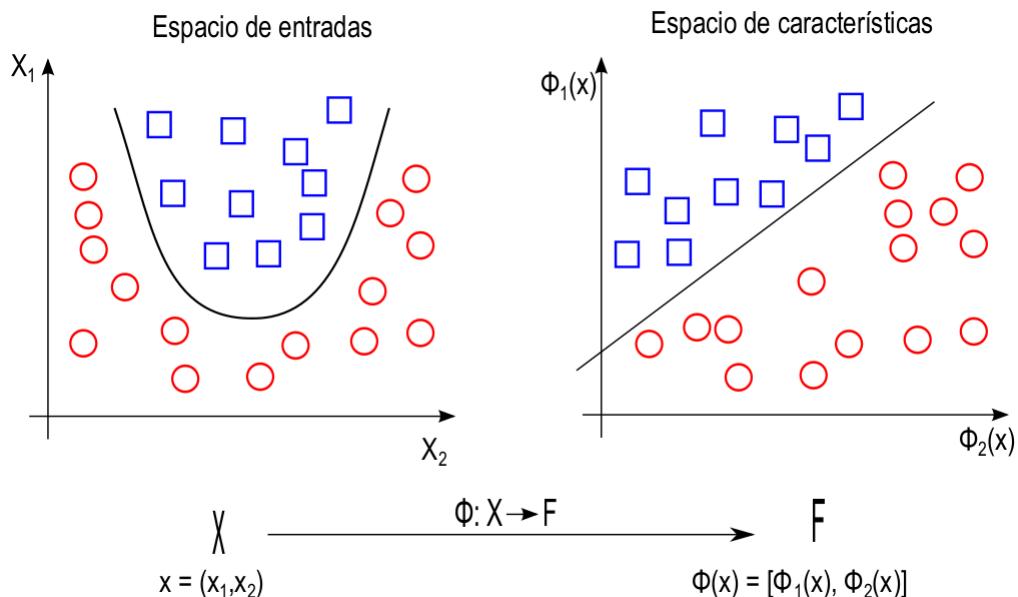


Figura 3.15: (SVM) Hiperplano solución de ejemplos no separables linealmente.

Se define entonces la función de decisión en el espacio de características como:

$$D(x) = (w_1\phi_1(x) + \dots + w_m\phi_m(x)) = \langle w, \Phi(x) \rangle$$

y, en su forma dual, la función de decisión se obtiene transformando la expresión de la frontera de decisión en:

$$D(x) = \sum_{i=1}^n \alpha_i^* y_i K(x, x_i)$$

donde $K(x, x')$ se denomina función kernel.

Una **función kernel** es una función $K : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ que asigna a cada par de elementos del espacio de entrada, \mathbb{X} , un valor real correspondiente al

productor escalar de las imágenes de dichos elementos en un nuevo espacio \mathcal{F} , es decir,

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle = (\phi_1(x)\phi_1(x') + \dots + \phi_m(x)\phi_m(x'))$$

Dado el conjunto de funciones base, $\Phi = \{\phi_1(x), \dots, \phi_m(x)\}$, el problema a resolver se expresa de la siguiente forma:

$$\begin{aligned} & \max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ & \text{s. a. } \begin{cases} \sum_{i=1}^n \alpha_i y_i = 0, & i = 1, \dots, n \\ 0 \leq \alpha_i \leq C, & \end{cases} \end{aligned}$$

Actualmente, no existe una forma teórica de encontrar el valor de C , simplemente la heurística de usar un valor grande (valiendo $C = \infty$ para el caso linealmente separable).

Teorema de Aronszajn: *Para cualquier función $K : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ que sea simétrica y semidefinida positiva, existe un espacio de Hilbert y una función $\Phi : \mathbb{X} \rightarrow \mathcal{F}$ tal que*

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad \forall x, x' \in \mathbb{X}$$

Una consecuencia importante de este teorema es que podemos construir una función kernel simplemente asegurándonos de que cumpla las dos condiciones del teorema, sin necesidad de conocer el conjunto de funciones base, $\Phi = \{\phi_1(x), \dots, \phi_m(x)\}$. Es decir, basta con conocer la forma funcional del kernel correspondiente, $K : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$, aún cuando este pudiese estar asociado a un conjunto infinito de funciones base.

Algunos ejemplos de funciones kernel se presentan a continuación:

- Kernel lineal:

$$K(x, x') = \langle x, x' \rangle$$

- Kernel polinómico de grado p :

$$K(x, x') = [\gamma \langle x, x' \rangle + \tau]^p$$

- Kernel gaussiano o de base radial:

$$K(x, x') = e^{-\gamma \|x-x'\|^2}, \quad \gamma > 0$$

- Kernel sigmoidal:

$$K(x, x') = \tanh(\gamma \langle x, x' \rangle - \tau)$$

- Multi-cuadrático inverso:

$$K(x, x') = (||x - x'||^{1/2} 2\sigma + c^2)^{-1}$$

- Kernel de intersección:

$$K(x, x') = \sum_{i=1}^n \min(x_i, x'_i)$$

A los parámetros γ , τ y p se les denomina parámetros del kernel, cuyo valor ha de ser fijado durante el proceso de entrenamiento, generalmente, mediante validación cruzada con algún conjunto de prueba.

3.1.10. Árboles de decisión: CART

Los **árboles de decisión** son una técnica que permite analizar decisiones secuenciales basadas en el uso de resultados y probabilidades asociadas. Existen muchos métodos de árboles de decisión, en este caso nos centraremos en los llamados árboles de clasificación y regresión, en inglés *Classification and Regression Trees* (CART), principalmente en los árboles de clasificación.

Partimos de una muestra de entrenamiento $(X_1, y_1), (X_2, y_2), \dots, (X_n, y_n)$ i.i.d. donde cada $X_i = (X_i^1, \dots, X_i^p)$ es un vector con p variables aleatorias e Y_i una variable unidimensional, discreta o continua. Y a partir de una muestra de entrenamiento construimos una estructura de tipo árbol.

Para construir el árbol maximal, partimos del nodo raíz con toda la muestra y vamos obteniendo los nodos interiores por particiones sucesivas, mediante una cierta pregunta o regla que involucra a uno de los p atributos. Se trata de árboles binarios, por lo tanto en función de la respuesta, cada nodo se divide en dos nodos hijos. Por último se elige algún criterio de parada, para saber cuando un nodo deja de dividirse y pasa a formar parte de un nodo terminal u hoja.

Una vez construido el árbol maximal, el predictor le asigna a cada región (hoja) un determinado valor:

$$E[y|(X^1, X^2)] = \sum_{j=1}^m f_j(X^1, X^2) \mathbf{1}_{\{(X^1, X^2) \in R_j\}}$$

donde, si y es continua (**árbol de regresión**) estimamos f_j por

$$\hat{f}_j = \frac{\sum_{\{i:(X_i^1, X_i^2) \in R_j\}} y_i}{\text{card}\{i : (X_i^1, X_i^2) \in R_j\}} \quad (\text{promedio de los } y_i \text{ en la región } R_j)$$

y si y es discreta (**árbol de clasificación**)

$$\hat{f}_j = \text{la clase más frecuente en } R_j$$

A continuación nos centramos en el caso de que y tome valores en $\{1, \dots, J\} \subset \mathbb{N}$.

Al construir una partición se intenta optimizar la homogeneidad de las regiones resultantes, por lo que en cada nodo del árbol, se intenta aumentar la pureza de los dos nodos obtenidos.

Las reglas de partición en un nodo dependen exclusivamente de los atributos. Supongamos que X^j toma valores en un conjunto finito $\{1, \dots, H\} \subset \mathbb{N}$, entonces las reglas son de la forma:

$$X^j \in C \text{ con } C \subset \{1, \dots, H\}$$

Entre todas las reglas posibles de una partición de un nodo, se debe seleccionar la que mejor contribuya al aumento de la homogeneidad de sus dos hijos. Para ello se define una medida de impureza sobre la variable de respuesta.

La **función de impureza** se define como una función ϕ definida sobre un conjunto de J -uplas $(p_1, \dots, p_J) \in \mathbb{R}^J$, tales que:

- $p_j \geq 0 \forall j = 1, \dots, J$
- $\sum_{j=1}^J p_j = 1$

verificando las siguientes propiedades:

- ϕ tiene un único máximo en $(\frac{1}{J}, \dots, \frac{1}{J})$
- ϕ tiene un mínimo 0 y solamente lo alcanza en los puntos de la forma $(1, 0, \dots, 0), (0, 1, \dots, 0), \dots, (0, 0, \dots, 1)$
- ϕ es una función simétrica de (p_1, \dots, p_J)

Dada una función de impureza ϕ , podemos determinar para cada nodo t de un árbol su medida de impureza como:

$$i(t) = \phi[p_1(t), \dots, p_J(t)]$$

donde p_j es la probabilidad condicional de que un elemento pertenezca a la clase j , dado que pertenece al nodo t y puede estimarse en la práctica, como la proporción de elementos de clase j en el nodo t .

Algunos de los criterios más utilizados en CART como medida de impureza de un nodo son:

- Medida de entropía:

$$i_{ent}(t) = - \sum_{j=1}^J p_j(t) \log p_j(t)$$

- Índice de Gini:

$$i_{Gini}(t) = \sum_{\substack{i,j=1 \\ i \neq j}}^J p_i(t)p_j(t) = 1 - \sum_{j=1}^J [p_j(t)]^2$$

Supongamos entonces que mediante una determinada regla hemos partido al nodo t en dos, t_I y t_D , y sea p_I y p_D la proporción de elementos del nodo t que caen sobre cada uno de los hijos respectivamente. A partir de dichos datos se establece una **medida de bondad** de una partición s , para un nodo t , de la siguiente manera:

$$\Delta i(s, t) = i(t) - p_I i(t_I) - p_D i(t_D) \geq 0$$

Es claro que el aumento de la bondad depende de la disminución de la impureza en los nodos hijos en relación al nodo padre. El criterio de selección de la mejor partición s^* en el nodo t , consiste en elegir aquella que proporciona la mayor bondad:

$$\Delta i(s^*, t) = \max_{s \in \psi} \{\Delta i(s, t)\}$$

donde ψ es el conjunto de todas las particiones posibles del nodo t .

Mediante las reglas de partición y comenzando desde el nodo raíz, se van particionando sucesivamente los nodos. Al terminar el proceso de partición se obtiene un árbol, sobre el cual se cuantifica conjuntamente la impureza de todas sus hojas. Para ello, definimos la **impureza del árbol** A de la siguiente manera:

$$I(A) = \sum_{t \in \tilde{A}} i(t)p(t)$$

donde \tilde{A} es el conjunto de hojas del árbol A y $p(t)$ la probabilidad de que un caso pertenezca al nodo t .

Una vez determinado que un nodo es terminal se debe asignar una clase. Normalmente esto se hace por el método del voto mayoritario, que consiste en asignarle al nodo t la clase j^* si $p_{j^*}(t) = \max_{j=1, \dots, J} p_j(t)$ y en caso de empate, se realiza un sorteo.

Existen varios criterios de parada para detener el proceso de partición para evitar que el árbol sea muy grande y se produzca un sobreajuste en el conjunto de entrenamiento. Un posible criterio de parada es declarar un umbral para las medidas de impureza. Pero si el umbral es muy bajo, se generan árboles muy grandes y si es muy alto, puede implicar que un nodo no se divide, cuando en realidad, una posterior partición de sus descendientes, sí está en condiciones de generar un buen decrecimiento de la impureza.

Otros criterios utilizados son, evitar partir un nodo si la cantidad de elementos es menor que un determinado umbral o si algunos de los dos nodos, que resultan de la partición óptima, no supera un umbral.

El tamaño del árbol es un parámetro de ajuste fundamental para el modelo, por lo que debería escogerse en función de los datos. Una solución posible sería primero construir el llamado árbol maximal, con la única condición de que no permitir nodos con muy pocos elementos, para luego aplicarle un proceso de poda, durante el cual se eliminan algunas de sus ramas o subárboles que obtengan beneficios muy pequeños, en lo que respecta a la disminución de la impureza.

Para la elección del mejor árbol podado necesitamos tener en cuenta el error de clasificación pero, como era de esperar, desconocemos el verdadero, por lo cual debemos trabajar con algún estimador de error a partir de los datos. A continuación definimos algunos de los estimadores más usados:

- **Estimador del error por resustitución:**

Se define el **estimador por restitución del error global de clasificación del árbol A** como:

$$R^{res}(A) = \sum_{t \in \tilde{A}} r(t)p(t)$$

donde $p(t)$ es la proporción de elementos del nodo t sobre el total de ejemplos considerados y $r(t) = 1 - p_{j^*}$ es el estimador por restitución del error de clasificación en el nodo t , es decir, la proporción de elementos mal clasificados.

Se puede demostrar que si se construye un árbol A' a partir de otro A partiendo de alguno de sus nodos, entonces:

$$R^{res}(A') \leq R^{res}(A)$$

o lo que es lo mismo:

$$R^{res}(t) \geq R^{res}(t_I) + R^{res}(t_D)$$

para todo nodo no terminal t y siendo $R^{res}(t) = r(t)p(t)$. Esto significa, que a medida que el árbol crece, el error por restitución disminuye. Sin embargo, en la práctica, la utilización este estimador tiene el inconveniente del mencionado problema de sobreajuste.

- **Estimador por muestra de prueba:**

Partimos la muestra inicial M en dos subconjunto: uno de entrenamiento, M_1 , y otro de prueba, M_2 , es decir, M_1 se utiliza para la construcción del árbol y M_2 para su evaluación.

Obtenemos el estimador de muestra de prueba repitiendo los cálculos del estimador anterior, pero tomando los datos del conjunto M_2 en lugar de la muestra completa.

Los tamaños de ambas muestras no tiene por qué ser iguales. Sin embargo, cuando el tamaño de la muestra no es lo suficientemente grande, este estimador no es el más adecuado.

- **Estimador por validación cruzada:**

Inicialmente partimos la muestra M en n conjuntos del mismo tamaño aproximadamente, M_1, M_2, \dots, M_n . A continuación construimos un primer árbol A_1 tomando $M^1 = M - M_1$ como muestra de entrenamiento y M_1 como muestra de prueba para su evaluación, obteniendo el siguiente error:

$$R_1^{vc}(A_1) = \sum_{t \in \tilde{A}_1} r(t)p(t)$$

Y repetimos el mismo procedimiento, tomando como muestra de prueba M_2 y $M^2 = M - M_2$ de entrenamiento. Así, sucesivamente, calculamos $R_2^{vc}(A_2), \dots, R_n^{vc}(A_n)$. Finalmente, definimos el estimador por validación cruzada como:

$$R^{vc}(A) = \frac{1}{n} \sum_{v=1}^n R_v^{vc}(A_v)$$

Frecuentemente se utiliza $n = 10$, pero también se suelen tomar valores como 5 ó 20.

Como se dijo anteriormente, árboles muy grandes generan problemas de sobreajuste y, además, son difíciles de interpretar. Por esto, se suele construir primero una secuencia de árboles podados y, posteriormente, elegir el mejor de estos.

Inicialmente se construye el árbol maximal, A_{\max} , aplicando las reglas de partición y los criterios de parada mencionados anteriormente.

Llamamos **rama** de un árbol al conjunto formado por un nodo y todos sus descendientes, y decimos que A' es un **sub-árbol** de A ($A' \prec A$) si A' se obtiene podando A , es decir, eliminando alguna de sus ramas.

La idea es, a partir de A_{\max} , obtener por podas sucesivas una secuencia anidada de sub-árboles, decreciente según la cantidad de hoja:

$$A_K \prec A_{K-1} \prec \dots \prec A_1 = A_{\max}$$

Para esto tendremos en cuenta, que si bien al podar un árbol podemos perder bondad (aumentar el error), se puede ganar simplicidad. Por tanto, se puede tolerar un pequeño error para conseguir un árbol menos complejo.

Llamamos **complejidad** de un árbol A a su cantidad de hojas, $\text{card}(\tilde{A})$. La **medida de costo-complejidad** asociada al árbol A como:

$$R_\alpha(A) = R(A) + \alpha \times \text{card}(\tilde{A}), \quad \alpha \in \mathbb{R} \text{ y } \alpha \geq 0$$

donde α recibe el nombre de **parámetro de complejidad**.

Se pretende eliminar, mediante podas, algunas ramas del árbol de partida, reduciendo $R_\alpha(A)$.

Nótese que $R_\alpha(A)$ es una combinación lineal entre el error o costo del árbol y su complejidad (tamaño). Valores altos de α penalizan árboles con muchas hojas. Si $\alpha \rightarrow \infty$, el mejor árbol tendría sólo una hoja, aunque sea el que peor costo $R(A)$ tiene, mientras que si $\alpha = 0$ no tendría en cuenta el tamaño y se quedaría con el mejor costo $R(A)$, es decir, con el maximal. Por tanto, se trata de encontrar un compromiso entre bondad y complejidad.

El procedimiento entonces parte del árbol maximal, $A_1 = A_{\max}$, tomando $\alpha = 0$. Sabemos que para cualquier poda, se obtiene un sub-árbol podado AP que verifica:

$$R(A) < R(AP)$$

y por lo tanto

$$R_{\alpha=0}(A) < R_{\alpha=0}(AP)$$

A medida que α crece, pero permanece pequeño, se mantiene esa relación, pero podría pasar que se invirtiese la relación al crecer. La disminución en complejidad hace que el costo-complejidad mejore, aunque el error sea mayor en el árbol podada. Por lo tanto, debemos encontrar el menor valor de α para el cual existe un sub-árbol con mejor costo-complejidad, es decir,

el menor valor de α y la rama más débil para podarla. A ese nuevo subárbol lo denotaremos por A_2 y al valor del parámetro por α_2 . Repitiendo sucesivamente el procedimiento, encontramos la mencionada secuencia de árboles:

$$A_K \prec A_{K-1} \prec \dots \prec A_1 = A_{\max}$$

De todos los árboles de la secuencia, nos quedamos con el que tiene asociado el menor error estimado, es decir elegimos el A_{K_0} que verifique:

$$R(A_{K_0}) = \min_K \{R(A_K)\}$$

3.2. Algoritmo de detección de retinopatía diabética

En esta sección, se explicará una posible solución a la detección de la retinopatía diabética de forma detallada.

Siguiendo el artículo [7], inicialmente se segmentan los vasos sanguíneos, exudados duros y microaneurismas para cada imagen de fondo de ojo. Como paso siguiente, se extrae el área de los vasos sanguíneos, exudados duros y microaneurismas de manera que puedan ser utilizadas en el tercer paso para la clasificación mediante el clasificador SVM o los árboles de decisión.

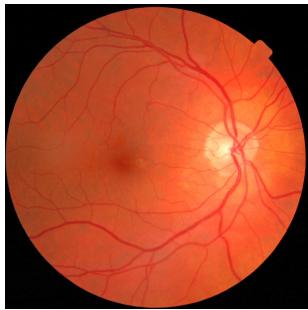
3.2.1. Detección y segmentación

a) Detección y segmentación de vasos sanguíneos:

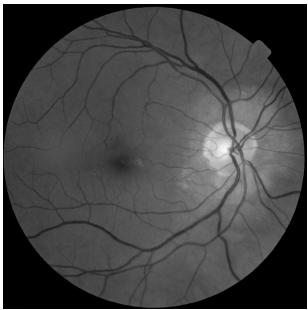
En etapas avanzadas de la retinopatía diabética suelen aparecer nuevos vasos anómalos y frágiles que no funcionan bien. Para detectar el árbol arterial principal y posibles neovasos generados se han realizado los siguientes pasos:

- 1) Se extrae el canal verde de la imagen de la retina ya que la sangre contiene características que aparecen más contrastadas en este canal. Este canal presenta mejor contraste entre las estructuras oculares y el fondo ya que la papila es intensa y los vasos sanguíneos y la mácula más oscuros que el fondo. Por lo tanto, suele ser el canal más empleado en el análisis de retinografías.
- 2) Sobre el canal verde se aplica el método de ecualización adaptativa del histograma de contraste limitado (CLAHE) para suavizar el fondo de la imagen.
- 3) Se normaliza la intensidad expandiéndola a través del rango de esta.
- 4) Se aplica el filtro de la mediana, y se realiza la diferencia entre esta y la obtenida en el punto anterior para poder obtener una imagen de los vasos sanguíneos resaltados. Con esto conseguimos eliminar de la imagen algunas de las estructuras que no forman parte de los vasos sanguíneos y que pueden provocar falsas detecciones.
- 5) Seguidamente se umbraliza por el método de Otsu, obteniendo los vasos sanguíneos en una imagen binaria.
- 6) Para acentuar los vasos se emplea un cierre con elemento estructurante con forma de línea.

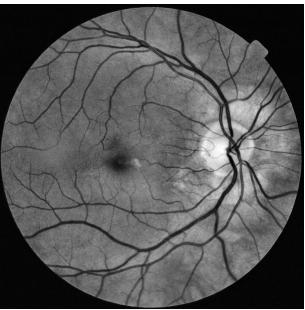
- 7) Por último, para remover los ruidos contenidos en la imagen binaria eliminamos los elementos conectados pequeños.



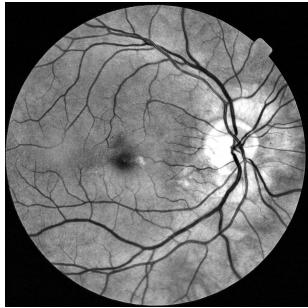
(a) Imagen de la retina



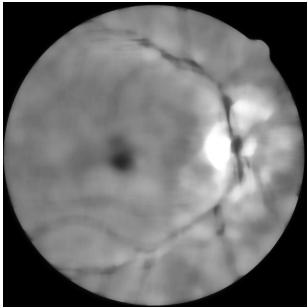
(b) Canal verde.



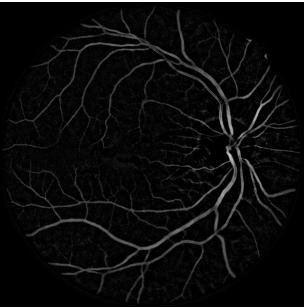
(c) Imagen ecualizada.



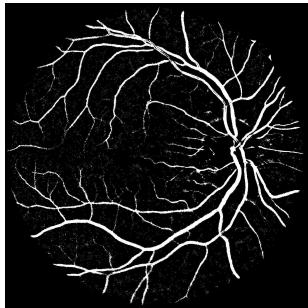
(d) Imagen con intensidad ajustada.



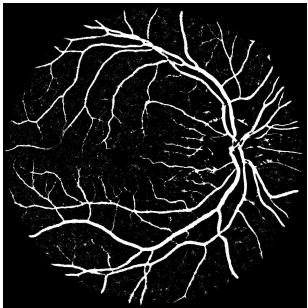
(e) Imagen generada por el filtro de la mediana.



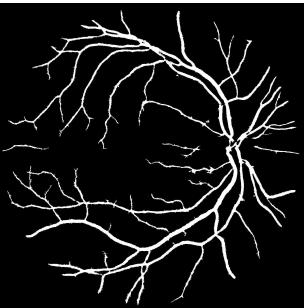
(f) Sustracción de imágenes.



(g) Imagen umbralizada.



(h) Cierre de la imagen.



(i) Imagen de vasos sanguíneos.

Figura 3.16: Secuencia de imágenes de detección y segmentación del vasos sanguíneos.

b) **Detección y segmentación de exudados duros:**

Se debe crear un borde circular que rodee la retina para detectar correctamente el área donde se encuentran los exudados duros, debido a que en algunas imágenes, los bordes circulares pueden ser confundidos por estos al ser más claros. Otro problema a la hora de detectar exudados duros es la similitud de coloración que los mismos poseen con el disco óptimo, por ello, realizamos también la detección de este.

■ *Detección del borde circular:*

- 1) Se extrae el canal rojo de la imagen de la retina.
- 2) Se umbraliza la imagen por el método de Otsu.
- 3) Se realiza la operación de gradiente morfológico con un elemento con forma de disco.

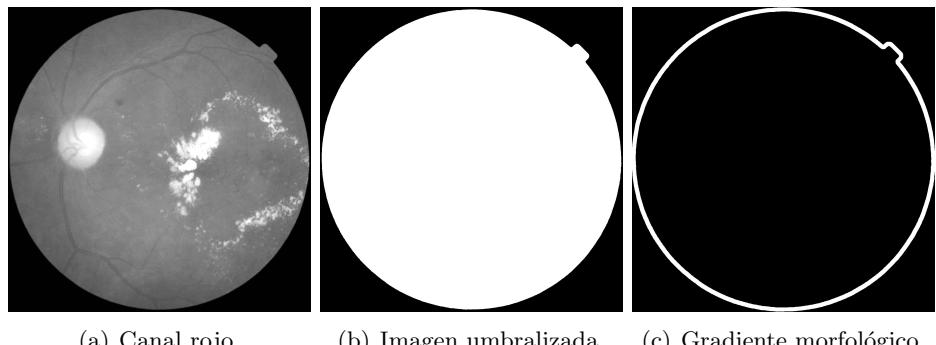


Figura 3.17: Secuencia de imágenes de detección y segmentación del borde circular.

■ *Detección de disco óptico:*

- 1) Se transforma la imagen de la retina a escala de grises.
- 2) Sobre dicha imagen se aplica el método de CLAHE.
- 3) Se realiza un ajuste de intensidad.
- 4) Se umbraliza a partir de un valor fijo.
- 5) Para acentuar la forma forma del disco óptico, se realiza las siguiente operaciones: erosión, cierre, relleno de huecos, otra vez cierre y dilatación.
- 6) Se selecciona el elemento que más puntos en común tenga con los vasos sanguíneos ya que estos emergen del disco óptico.
- 7) Se selecciona un punto central a partir del cual se dibuja el disco óptico con forma circular.

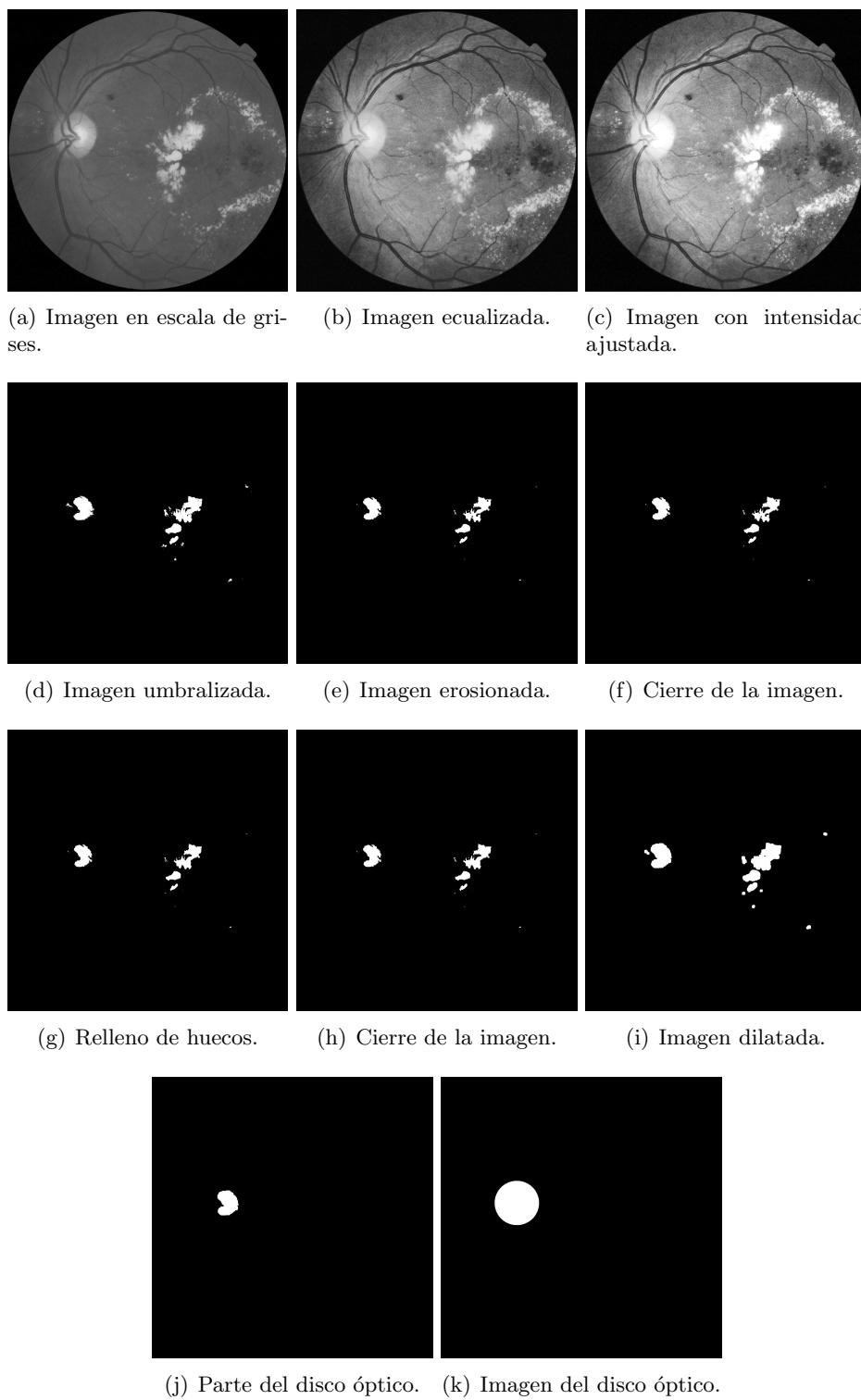
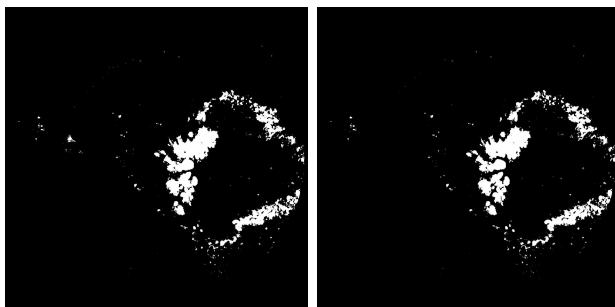
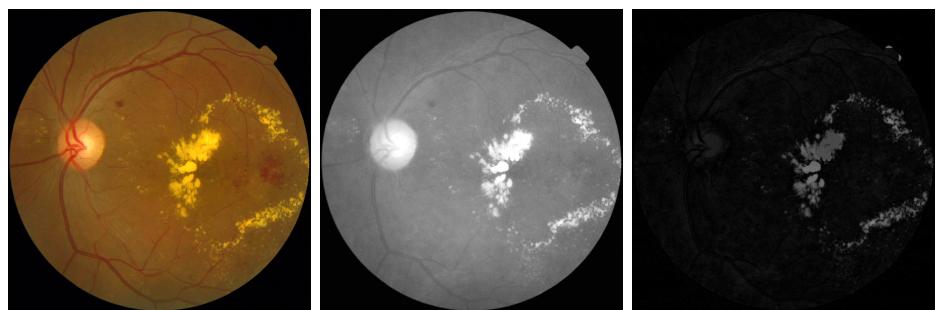


Figura 3.18: Secuencia de imágenes de detección y segmentación del disco óptico.

Debido a que los exudados duros son lesiones claras, estos se detectarán a partir del canal de intensidad de la imagen mediante los siguientes pasos:

- 1) Se extrae el canal rojo de la imagen original.
 - 2) Se aplica la transformada de Top-Hat con elemento estructurante en forma de disco para conseguir las componentes brillantes, corrigiendo los efectos de la iluminación no uniforme.
 - 3) Se elimina el borde circular.
 - 4) Se umbraliza por el método de entropía máxima.
 - 5) Se extrae la parte restante de los vasos sanguíneos y el disco óptico.



(d) Imagen umbralizada. (e) Imagen de exudados duros.

Figura 3.19: Secuencia de imágenes de detección y segmentación de los exudados duros.

c) Detección y segmentación de microaneurismas:

Los microaneurismas son uno de los primeros signos de retinopatía diabética. Se presentan como pequeños puntos rojos en las retino-

grafías, difíciles de detectar. En este caso, para su localización se ha realizado principalmente a partir de los pasos explicados en [18]:

- 1) Obtención del canal de intensidad de la imagen original a partir del canal rojo y verde.
- 2) Se aplica el filtro de la mediana y se calcula la diferencia de esta y la anterior, obteniendo una imagen con menos ruido.
- 3) Se realiza un ajuste de intensidad mediante el método de CLAHE.
- 4) Se segmenta la imagen aplicando el método de Otsu pero añadiéndole un pequeño error a partir de un valor determinado.
- 5) Se extrae las partes pertenecientes a los vasos sanguíneos, disco óptico, borde circular y exudados duros.
- 6) Se seleccionan los elementos conectados con forma circular cuyo radio esté en el intervalo [5, 50] ya que los microaneurismas tienen un tamaño de 10-100 micras de diámetro.

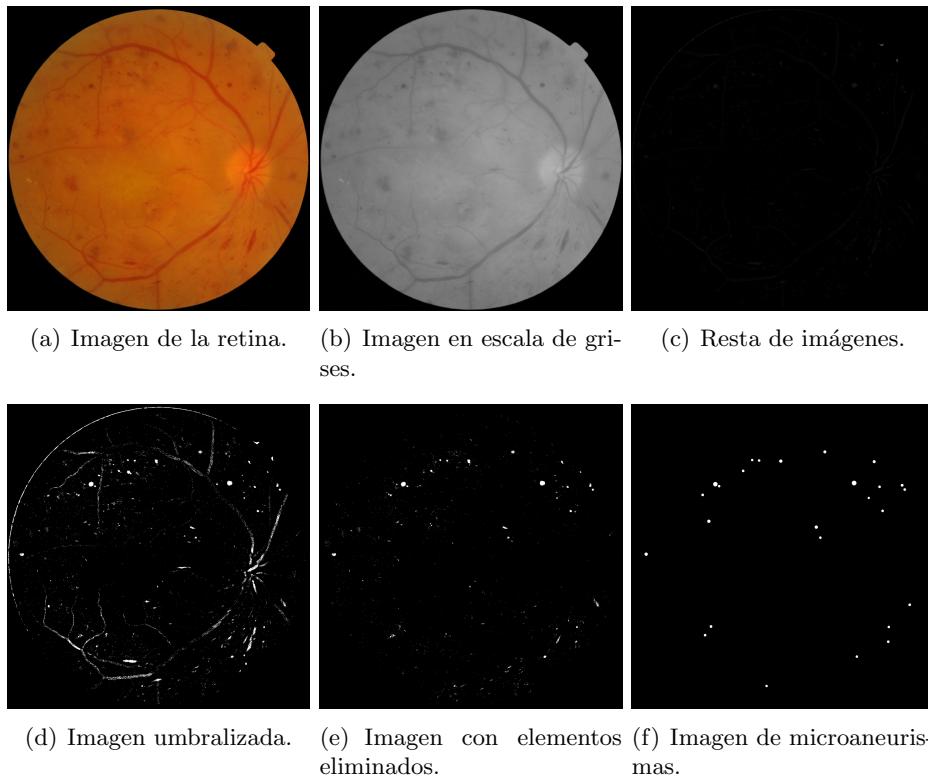


Figura 3.20: Secuencia de imágenes de detección y segmentación de los microaneurismas.

3.2.2. Extracción de características

Uno de los vectores usado en la clasificación almacena tres características de las imágenes segmentadas:

- a) Área de vasos sanguíneos.
- b) Área de exudados duros.
- c) Área microaneurismas.

Las cuales se obtienen al determinar el número total de píxeles en blanco en las imágenes correspondientes.

Motivados por la mejora conseguida en [17], se decidió realizar una comparativa entre sólo tener en cuenta estas tres características, y considerar además las medidas de contraste, homogeneidad, correlación y energía, obtenidas a partir de la matriz de co-ocurrencia, para formar el vector de características que será utilizado para la clasificación.

3.2.3. Clasificador

El objetivo de cualquier clasificador es encontrar una frontera que nos permita separar los ejemplos positivos (en este caso, retinas sanas o normales) de los negativos (retinas enfermas o anormales).

En busca de los mejores resultados, la clasificación es realizada tanto por el clasificador binario SVM con núcleos diversos como por los árboles de decisión. El clasificador recibe un conjunto de características de entrenamiento, cada una etiquetada con una de las siguientes categorías: 0 en caso de ser una retina sana y 1 en caso de ser la retina de un paciente con retinopatía diabética. A partir de estas características se construye un modelo que posteriormente será empleado para clasificar las imágenes de prueba.

Sin embargo, antes de realizar la clasificación, los datos se normalizan para asegurarnos de que las mediciones de distancia sean ponderadas de la misma forma para cada clase ya que si no se llevase a cabo esta normalización, se correría el riesgo de que variables con escalas grandes dominasen. En este caso, se ha realizado la normalización min-max en el intervalo [0, 1]:

$$x' = \frac{x - \min_x}{\max_x - \min_x}$$

donde x es una variable o característica, y \min_x y \max_x son los valores mínimo y máximo, respectivamente, de x , y x' la variable normalizada.

Capítulo 4

Diseño e implementación

Para el desarrollo de este trabajo se ha empleado la herramienta de software conocida como MATLAB, concretamente R2017a, ya que nos ofrece una amplia gama de funciones orientadas al procesamiento de imágenes y un gran soporte de ayuda para saber en todo momento cómo utilizarlas.

MATLAB combina computación numérica, gráficos y capacidades de lenguaje fáciles de usar. Su lenguaje está inherentemente orientado a las matrices, por lo que es ideal para el procesamiento de imágenes. Además existe un conjunto de funciones que amplía las capacidades para el desarrollo de aplicaciones y de nuevos algoritmos en el campo del procesado y análisis de imágenes, agrupadas en *Image Processing Toolbox*[11].

4.1. Funciones básicas

4.1.1. Función `rgb2gray`

La función `rgb2gray` nos permite transformar una imagen RGB a escala de grises mediante la siguiente sintaxis:

```
I = rgb2gray(RGB)
```

Convierte la imagen RGB en la imagen de escala de grises `I` eliminando la información de matiz y saturación mientras que conserva la luminancia.

4.1.2. Función `imadjust`

La función `imadjust` nos permite mostrar imágenes dentro de un rango de intensidad determinado o simplemente mejorar el contraste de una imagen en escala de grises mediante la expansión de su histograma.

Existen varias sintaxis para esta función, cuyas descripciones podemos encontrar en [29] o en [8], pero para este TFG se ha empleado la siguiente:

```
J = imadjust(I)
```

Ajusta los valores de intensidad de la imagen I en escala de grises. Por defecto, satura el 1% de los datos con mayor e igualmente hace con los de menor intensidad, usando los límites obtenidos automáticamente por la función `stretchlim`. Esta operación incrementa el contraste de la imagen de salida J.

4.1.3. Función adapthisteq

Con la función `adapthisteq` se mejora el contraste de la imagen de entrada I empleando el método de ecualización adaptativa del histograma con contraste limitado CLAHE, el cual es útil para potenciar los detalles de una imagen y mejorar el contraste tanto general como local.

La sintaxis de esta función es la siguiente:

```
J = adapthisteq(I, param1, val1, param2, val2...)
```

Se mejora el contraste de cada región de la imagen en escala de grises I para que el histograma concuerde manera aproximada con el histograma definido en el parámetro de '`Distribution`'. Seguidamente, se realiza la interpolación bilineal para eliminar los "bordes" indeseados, como se ha explicado anteriormente.

Algunos de los parámetros de entrada opcionales son los siguientes:

- '`NumTiles`' : Vector de dos componentes que determinan el número de regiones en las que se desea dividir la imagen. Nótese que cada componente debe valer al menos 2.

Por defecto vale [8 8], es decir, la imagen se divide en 64 regiones.

- '`ClipLimit`' : Número real perteneciente al intervalo [0, 1] que define el límite de mejora del contraste. Cuanto mayor es dicho número, mayor contraste se obtiene.

'`ClipLimit`' previene sobresaturación de la imagen, especialmente en zonas homogéneas. Estas zonas se caracterizan por tener un pico en el histograma debido a que existen demasiados píxeles con el mismo nivel de intensidad.

Por defecto vale 0.01.

- '`Distribution`' : Cadena que permite especificar la distribución deseada del histograma. Los distribuciones que se le puede asignar a este parámetro son las siguientes:

- '`uniform`' : Histograma plano.

- 'rayleigh': Histograma en forma de campana.
- 'exponential': Histograma curvado.

Por defecto su valor es 'uniform'.

4.1.4. Función medfilt2

La función `medfilt2` sirve para aplicar el filtro de la mediana de una imágen bidimensional. Dicha función se ha utilizado con la siguiente sintaxis:

```
B = medfilt2(A, [m n])
```

Aplica el filtro de la mediana a la imagen bidimensional `A`, obteniendo una imagen, también bidimensional, `B` donde cada píxel contiene el valor de la mediana a partir de los $m \times n$ vecinos alrededor del correspondiente píxel de la imagen de entrada.

4.1.5. Función strel

La función `strel` sirve para crear un elemento estructurante de una forma determinada, el cual es una parte esencial de operaciones morfológicas tales como la dilatación o la erosión.

Algunas de las posibles sintaxis para estos son las siguientes:

```
SE = strel('disk',r,n)
```

Crea un elemento estructurante en forma de disco, donde `r` es el radio y `n` especifica el número de líneas utilizadas para aproximar la forma del disco. Las operaciones morfológicas usando aproximaciones del disco funcionan mucho más rápido cuando los elementos estructurantes utilizan aproximaciones.

```
SE = strel('line',len,deg)
```

Crea un elemento estructurante lineal que es simétrico con respecto al centro del vecindario. El parámetro `deg` especifica el ángulo en grados de la línea desde el eje horizontal y en sentido antihorario, mientras que `len` es aproximadamente la distancia entre los centros de los elementos estructurantes en los extremos opuestos de la línea.

4.1.6. Función imerode

La función `imerode` nos permite erosionar la imagen a partir de uno o varios elementos estructurantes. En este caso, se ha empleado la sintaxis:

```
J = imerode(I, SE)
```

Erosina la imagen en escala de grises, binaria o binaria empaquetada I a partir del elemento estructurante o array de elementos estructurantes, SE , devueltos por la función `strel` o `offsetstrel`.

Si la imagen de entrada es lógica, `imerode` realiza la erosión binaria; en cualquier otro caso realiza una erosión en escala de grises. En dicho caso, el elemento estructurante debe ser plano.

Si SE es un array de elementos estructurantes, `imrode` realizará múltiples erosiones sucesivas de la imagen de entrada, aplicando progresivamente cada elemento estructurante perteneciente a este array.

4.1.7. Función `imdilate`

La función `imdilate` nos permite dilatar la imagen a partir de uno o varios elementos estructurantes. En este caso, se ha empleado la sintaxis:

$$J = \text{imdilate}(I, SE)$$

Dilata la imagen en escala de grises, binaria o binaria empaquetada I a partir del elemento estructurante o array de elementos estructurantes, SE , devueltos por la función `strel` o `offsetstrel`.

Análogamente a la función anterior tenemos que si la imagen de entrada es lógica, `imedilate` realiza la dilatación binaria; en cualquier otro caso realiza una dilatación en escala de grises. En dicho caso, el elemento estructurante debe ser plano.

Si SE es un array de elementos estructurantes, `imdilate` realizará múltiples dilataciones sucesivas de la imagen de entrada, aplicando progresivamente cada elemento estructurante perteneciente a este array.

4.1.8. Función `imopen`

La función `imopen` permite realizar la apertura morfológica de la imagen a partir de una elemento estructurante. Dicha función se ha empleado con la siguiente sintaxis:

$$J = \text{imopen}(I, SE)$$

Abre morfológicamente la imagen binaria o en escala de grises I a partir de un único elemento estructurante SE . La operación morfológica de apertura de una imagen consiste simplemente en una erosión seguida por una dilatación, empleando el mismo elemento estructurante en ambas operaciones.

4.1.9. Función imclose

La función `imclose` permite actuar cerrando morfológicamente la imagen a partir de una elemento estructurante. Dicha función se ha utilizado con la siguiente sintaxis:

```
J = imclose(I,SE)
```

Cierra morfológicamente la imagen binaria o en escala de grises `I` a partir de un único elemento estructurante `SE`. La operación morfológica de cierre de una imagen consiste simplemente en una dilatación seguida por una erosión, empleando el mismo elemento estructurante en ambas operaciones.

4.1.10. Función imfill

La función `imfill` rellena las regiones y huecos de una imagen, utilizando en la implementación con la siguiente sintaxis:

```
BW2 = imfill(BW,'holes')
```

Rellena los huecos de la imagen binaria `BW`. En esta sintaxis, un hueco es un conjunto de píxeles de fondo que no se pueden alcanzar llenando el fondo desde el borde de la imagen.

La función `imfill` utiliza por defecto una vecindad de 4 píxeles de fondo para entrada en 2-D. Dicho valor se puede anular a partir de la siguiente sintaxis:

```
BW2 = imfill(BW,conn,'holes')
```

donde `conn` puede valer 4 ó 8.

4.1.11. Función bwconncomp

La función `bwconncomp` sirve para encontrar las componentes conectadas en imágenes binarias. Esta función se puede usar con la siguiente sintaxis:

```
CC = bwconncomp(BW)
```

Devuelve las componentes conectadas encontradas en la imagen binaria `BW`. Dicha función usa por defecto la conectividad de 8 para dos dimensiones. Si queremos usar añadir usar la siguiente sintaxis especificando la conectividad deseada en `conn`.

```
CC = bwconncomp(BW, conn)
```

4.1.12. Función imtophat

La función `imtophat` permite aplicar el filtro Top-Hat a partir de un elemento estructurante. En este caso, se ha utilizado dicha función con la siguiente sintaxis:

```
J = imtophat(I,SE)
```

Realiza la transformación Top-Hat a la imagen en escala de grises o binaria `I` para devolver la imagen filtrada `J`. El filtro Top-Hat calcula la diferencia entre la imagen original y la resultante de realizar la apertura morfológica a partir del elemento estructurante `SE`, usando para ello la función `imopen`.

4.1.13. Función imfindcircles

La función `imfindcircles` sirve para encontrar círculos utilizando la transformada de Hough circular. En este caso, se ha empleado con la siguiente sintaxis:

```
[centers,radii] = imfindcircles(I,radiusRange)
```

Encuentra los círculos que puede haber en la imagen `I` cuyos radios en el rango especificado en `radiusRange`, obteniendo una matriz de dos columnas, `centers`, la cual contiene las coordenadas de los centros de los círculos en la imagen y un vector, `radii`, con los radios estimados correspondientes a cada círculo en `centers`.

Además, se puede especificar parámetros de entrada adicionales, como es el factor de sensibilidad para la transformada de Hough circular, '`Sensitivity`', que es un escalar no negativo contenido en el intervalo $[0, 1]$. Cuanto más alto es el valor de sensibilidad, más objetos circulares son detectados y, por tanto, mayor es el riesgo de cometer errores.

4.1.14. Función imbinarize

La función `imbinarize` se emplea para obtener una imagen binaria dada una en escala de grises mediante la umbralización de esta. Posibles sintaxis de esta función son las siguientes:

```
BW = imbinarize(I)
```

Crea una imagen binaria a partir de la imagen `I` reemplazando todos los valores por debajo del umbral hallado mediante el método de Otsu, con unos y el resto de valores con ceros.

```
BW = imbinarize(I,T)
```

Crea una imagen binaria a partir de la imagen I utilizando el valor del umbral T , dicho parámetro puede ser un umbral global de la imagen (valor entre 0 y 1) o un umbral adaptativo local, especificado como una matriz de valores del mismo tamaño que I .

4.1.15. Función `graythresh`

La función `graythresh` nos permite obtener el umbral global de una imagen mediante el método de Otsu, y presenta la siguiente sintaxis:

```
level = graythresh(I)
```

Calcula un umbral global normalizado, `level`, perteneciente al intervalo $[0, 1]$. Dicho umbral puede ser utilizado para convertir una imagen de intensidades a una imagen binaria, con la ayuda de la función `imbinarize`.

4.1.16. Función `immaxentropy`

La función `immaxentropy` binariza la imagen a partir del umbral calculado mediante el método de entropía máxima. La sintaxis de dicha función corresponde a la siguiente:

```
I1 = immaxentropy(I)
```

Binariza la imagen en escala de grises I a partir del método de entropía máxima. Esta función ha sido implementada en el fichero `immaxentropy.m` a partir de la explicación matemática dada anteriormente resultado de los siguientes pasos:

1. Obtenemos el histograma correspondiente a la imagen I con ayuda de la función `imhist`, el cual nos proporciona la frecuencia de ocurrencia de cada nivel de intensidad.
2. Calculamos la probabilidad de ocurrencia de cada nivel de gris.
3. Hallamos todas las funciones de entropía del fondo y del objeto probando con los distintos umbrales posibles, y elegimos aquel que maximice la suma de ambas funciones de entropía.

Nótese que hemos supuesto que $0\log(0) = 0$, por tanto, en la implementación se ha calculado inicialmente todos los valores de $\frac{p_i}{p_t}$ y $\frac{p_j}{p_m}$, y obviado aquellos que daban cero como resultado.

4.1.17. Función `graycomatrix`

La función `graycomatrix` sirve para generar la matriz de co-ocurrencia a partir de la siguiente sintaxis:

```
glcm = graycomatrix(I)
```

Crea una matriz de co-ocurrencia de niveles de gris (GLCM) de la imagen `I`. Cada elemento (i, j) en `glcm` especifica el número de veces que el píxel con valor i aparece horizontalmente adyacente al píxel con valor j .

Esta puede utilizarse con parámetros adicionales tales como:

- `'NumLevels'`: Número de niveles de gris, especificado como un entero. El número de niveles de intensidad determina el tamaño de la matriz de co-ocurrencia, y por defecto vale 2 para imágenes binarias y 8 para numéricas. En este último caso, se escalaría los valores en `I` y se obtendrían valores enteros entre 1 y 8.
 - `'GrayLimits'`: Rango usado para escalar la imagen de entrada en escala de grises, especificado como un vector formado por dos elementos `[low, high]`. Si N es el número de niveles de intensidad empleados para escalar, `[low, high]` se divide en N intervalos de igual tamaño y los valores pertenecientes a un intervalo se asignan a un único nivel de gris. Los valores de gris menores o iguales que `low` se escalan a 1, mientras que los que son mayores o iguales que `high` se escalan a `'NumLevels'`.
- Si `'GrayLimits'` corresponde a `[]`, `graycomatrix` utiliza el valor mínimo y máximo de los niveles de gris de `I` como límites, es decir, equivale a `[min(I(:)) max(I(:))]`.

4.1.18. Función `graycoprops`

La función `graycoprops` nos proporciona las propiedades deseadas de la matriz de co-ocurrencia y responde a la siguiente sintaxis:

```
stats = graycoprops(glcm,properties)
```

Calcula las características de la textura comúnmente extraídas de la matriz de co-ocurrencia `glcm` y especificadas en `properties`. En caso de que este último parámetro no se especifique, `graycoprops` devuelve un `struct` con las propiedades de contraste, correlación, energía y homogeneidad.

4.1.19. Función `fitcsvm`

La función `fitcsvm` nos permite realizar un entrenamiento binario empleando la técnica de máquinas de vectores soporte, normalmente conocida como SVM. La sintaxis de dicha función que se empleará corresponde a la siguiente:

```
Mdl = fitcsvm(X,Y)
```

Devuelve el modelo resultante de aplicar la técnica del SVM utilizando las muestras de entrenamiento de la matriz X y las etiquetas del vector Y.

Esta función puede ser llamada con el parámetro relacionado con una función *kernel* distinta de la lineal (que es la que usa por defecto para el aprendizaje de dos clases), 'KernelFunction':

- 'gaussian'/'rbf': Núcleo gaussiano o de base radial.
- 'linear': Núcleo lineal.
- 'polynomial': Núcleo polinómico. Se puede utilizar el parámetro 'PolynomialOrder' para especificar el orden de este polinomio.

4.1.20. Función fitctree

La función **fitctree** ajusta el árbol de decisión de la clasificación binaria. Dicha función será usada con la siguiente sintaxis:

```
tree = fitctree(X,Y)
```

Devuelve un árbol de decisión de clasificación binaria ajustado en función de las variables de entrada contenidas en la matriz X y la salida Y.

4.1.21. Función predict

La función **predict** predice las etiquetas utilizando el modelo de clasificación de análisis discriminante. La sintaxis de dicha función corresponde a la siguiente:

```
label = predict(Mdl,X)
```

Predice las etiquetas de los datos almacenados en la tabla o matriz X, basándose en el modelo de clasificación entrenado Mdl, y devuelve un vector con estas.

4.2. Funciones de detección

En esta sección se explica las diferentes funciones que se han implementado a partir de las funciones básicas para lograr detectar las áreas pertenecientes a los vasos sanguíneos, los exudados duros y los microaneurismas. Cada una de ellas devuelve una imagen en binario, cuyos píxeles valen 1 si se ha detectado que forma parte de la estructura correspondiente y 0 en caso contrario.

Cabe destacar que los parámetros utilizados en las diferentes funciones de erosión, dilatación, CLAHE, etc, se han elegido de manera experimental.

4.2.1. Funciones auxiliares

Como ayuda tanto en la detección de los exudados duros como de los microaneurismas, se han implementado las funciones de detección del borde circular, `detection_edge`, y del disco óptico, `detection_opticdisc`.

Función `detection_edge`

La función `detection_edge` nos permite detectar el borde circular a partir de la siguiente sintaxis:

```
edge = detection_edge(I)
```

Devuelve una imagen binaria `edge` con el borde circular que rodea a la retina en la imagen `I`.

Función `detection_opticdisc`

La función `detection_opticdisc` sirve para detectar el disco óptico en una imagen de fondo de ojo. La sintaxis de dicha función es:

```
od = detection_opticdisc(I, vessels, r)
```

Devuelve una imagen binaria `od` con la posición del disco óptico de la imagen `I`.

Nótese que se le puede pasar como parámetro los vasos sanguíneos, `vessels`, para determinar qué elemento de todos los detectados corresponde al disco óptico ya que desde este emergen los vasos sanguíneos, y el radio estimado, `r`, para dibujar el disco óptico. En caso de que el radio no sea especificado se le asignará por defecto 80, y si los vasos sanguíneos tampoco, estos se calcularán a partir de la función `detection_vessels`.

4.2.2. Función `detection_vessels`

La función `detection_vessels` nos permite detectar los vasos sanguíneos en una imagen de fondo de ojo. La sintaxis de dicha función se corresponde a la siguiente:

```
vessels = detection_vessels(I)
```

Devuelve una imagen binaria `vessels` en la cual podemos ver con valor 1 las partes detectadas pertenecientes a los vasos sanguíneos de la imagen `I`.

4.2.3. Función detection_hardexudates

La función `detection_hardExudates` nos permite detectar los exudados duros en una imagen de fondo de ojo. La sintaxis de dicha función se corresponde a la siguiente:

```
he = detection_hardExudates(I, vessels, od, edge)
```

Devuelve una imagen binaria `he` en la cual podemos ver con valor 1 las partes detectadas pertenecientes a los exudados duros de la imagen `I`.

Esta función realiza la extracción de los vasos sanguíneos, el disco óptico y el borde circular como paso final para eliminar falsos exudados duros, por lo que es posible llamar a la función especificando estos parámetros, pero no obligatorio. En caso de que no sean proporcionados, estos se calcularán a partir de las funciones correspondientes.

4.2.4. Función detection_microaneurysms

La función `detection_microaneurysms` nos permite detectar los microaneurismas en una imagen de fondo de ojo. La sintaxis de dicha función se corresponde a la siguiente:

```
ma = detection_microaneurysms(I, vessels, od, edge, he)
```

Devuelve una imagen binaria `ma` en la cual podemos ver con valor 1 las partes detectadas pertenecientes a los microaneurismas de la imagen `I`.

Análogamente a la función anterior, en esta se realiza la extracción de los vasos sanguíneos, el disco óptico, el borde circular y los exudados duros como paso final para eliminar falsos microaneurismas, por lo que es posible pasarle como parámetros estos. En caso de que no sean proporcionados, estos se calcularán a partir de las funciones correspondientes.

4.3. Funciones de extracción de características

El vector de características se crea a partir de las áreas de vasos sanguíneos, exudados duros y microaneurismas, además de las características relacionadas con la textura que son extraídas de la matriz de co-ocurrencia.

4.3.1. Función extraction

La función `extraction` sirve para crear el vector de características asociado a una retinografía determinada y sigue la siguiente sintaxis:

```
features=extraction(image, r)
```

Se crea el vector de característica de la imagen `imagen` (este parámetro corresponde al nombre y la ubicación de esta ya que la imagen se leerá dentro de esta función). Como se deben calcular las distintas áreas de las lesiones típicas de la retinopatía diabética, se puede especificar el radio `r` con que se dibujará el disco óptico. Por defecto, este se tomará como 80.

Nótese que esta función es auxiliar de `extractions`. Por lo tanto, se encuentra en el fichero correspondiente a dicha función.

4.3.2. Función `extractions`

Para poder realizar tantas pruebas como se desee, sin tener que estar calculando constantemente los vectores de características correspondientes a las distintas imágenes, se ha decidido crear un fichero que contendrá todos los vectores de características de una determinada carpeta. Esto se consigue llamando a la función con la siguiente sintaxis:

```
extractions(folder, filename, r)
```

Crea un fichero llamado `filename` a partir de las imágenes especificadas en un fichero que se debe denominar `datos.txt`, el cual debe encontrarse en la carpeta `folder`. Dicho fichero debe contener una columna con los nombres de las imágenes, contenidas en `folder`, que se desean tener en cuenta para extraer sus vectores de características.

1 Ejemplo de fichero `datos.txt`

- 1: Image name Retinopathy grade
 - 2: 20051020_43808_0100_PP.tif 0
 - 3: 20051020_43906_0100_PP.tif 3
 - 4: 20051020_44261_0100_PP.tif 0
 - 5: 20051020_44284_0100_PP.tif 0
 - 6: 20051020_44714_0100_PP.tif 0
 - 7: 20051020_44843_0100_PP.tif 3
 - 8: 20051020_44901_0100_PP.tif 2
 - 9: 20051020_44923_0100_PP.tif 2
 - 10: 20051020_62014_0100_PP.tif 1
 - 11: ...
-

A partir de estas se creerá el fichero `filename` que contendrá inicialmente el tamaño de la muestra (es decir, el número de imágenes que se tendrá en cuenta para las extracciones), las etiquetas correspondientes a las distintas muestras y posteriormente el vector de características de cada una de ellas.

2 Ejemplo de fichero `filename`

```
210
0
1
0
1
...
136881 2882 0 0.030748 0.984629 0.960851 0.464212
140104 26 458 0.063427 0.968397 0.975890 0.317310
200872 12 115 0.105690 0.947184 0.904150 0.286196
299388 49869 2904 0.076632 0.961728 0.943899 0.277359
...
```

Cabe notar que el fichero `filename` no se guardará en la carpeta `folder`. Si simplemente se especifique el nombre del fichero este se creará o sobre-escribirá, en caso de que ya existiese, en la carpeta en la que se encuentra trabajando MATLAB, si se desea que tenga una ruta diferente se deberá especificar en el parámetro correspondiente.

Por otro lado, también se puede especificar el radio del disco óptico, pero en este caso tampoco es necesario, siendo 80 su valor por defecto.

4.4. Funciones de entrenamiento y validación

Persiguiendo nuestro objetivo, pasamos a dividir un conjunto de muestras de nuestra base de datos de imágenes de la retina para entrenar un subconjunto y comprobar la calidad del modelo obtenido a partir de estas, pudiendo así estimar cómo de bueno es el algoritmo.

4.4.1. Función auxiliar

Función `read`

La función `read` sirve para almacenar los vectores de características y sus etiquetas asociadas. Esta se puede usar a partir de la siguiente sintaxis:

```
[X, y]=read(filename)
```

Crea la matriz formada a partir los vectores de características `X` y el vector `y` que contiene las etiquetas de las imágenes asociadas del fichero `filename`. La matriz `X` es normalizada por columnas para obtener mejores resultados en la clasificación.

4.4.2. Función train

La función **train** permite entrenar un subconjunto de muestras a partir del clasificador SVM. Dicha función puede ser usada con la siguiente sintaxis:

```
model=train(features, grade, indices, classif)
```

Se extrae un subconjunto, especificado en **indices**, de la matriz formada por los vectores de características, **features**, con sus correspondientes etiquetas extraídas del vector **labels** para poder obtener un modelo entrenado a partir del parámetro **classif**:

- '1': SVM con núcleo lineal.
- 'g': SVM con núcleo gaussiano.
- 'p': SVM con núcleo polinómico de orden 4.
- **En otro caso:** Árbol de decisión.

En caso de que este último parámetro no se especifique se entrenará con el clasificador SVM polinómico.

4.4.3. Función test

La función **test** nos proporciona los distintos porcentajes de acierto de un subconjunto de muestras. Para ello, se emplea con la siguiente sintaxis:

```
[SP, SN, A] = test(features, grade, model, indices)
```

Se estiman las etiquetas del subconjunto de muestras extraido de la matriz **features** a partir de los **indices** especificados. Y a continuación, se comprueba si las etiquetas correspondientes, proporcionadas por el vector **grade**, coinciden con las estimadas. Llevando un control de los aciertos y fallos, se calcula la sensibilidad **SP**, especificidad **SN** y exactitud **A**.

4.4.4. Función validation

La función **validation** permite realizar la técnica de validación cruzada conocida como *5-fold cross validation*, llamando a dicha función con la siguiente sintaxis.

```
validation(filenameIN, filenameOUT, n, classif, num)
```

Se extraen los vectores de características y etiquetas asociadas del fichero **filenameIN**. A continuación se crean las 5 particiones para posteriormente realizar las **num** pruebas tomando **n** (dicho valor debe ser un entero contenido en el intervalo [1, 7]) características, cuyos resultados serán almacenados en

filenameOUT. Los datos correspondientes serán entrenados con el clasificador especificado en **classif**, cuyos valores permitidos son los explicados en la función anterior.

3 Ejemplo de fichero filenameOUT

Sensibilidad Especificidad Exactitud

0.961538	1.000000	0.976190
0.925926	1.000000	0.952381
1.000000	1.000000	1.000000
1.000000	1.000000	1.000000
1.000000	0.941176	0.976190
<hr/>		
0.977493	0.988235	0.980952

Los parámetros que no tienen que ser especificados son **n**, cuyo valor por defecto es 7 (en el caso de que se llame a la función con un valor erróneo, también se asignará este), **classif**, cuyo valor por defecto es 'p', y **num**, siendo su valor por defecto 5.

Como podemos observar, en el fichero **filenameOUT** se muestran los resultados de las 5 particiones realizadas y además la media correspondiente a cada medida.

Capítulo 5

Análisis de resultados

La validación cruzada es un procedimiento de remuestreo utilizado para evaluar modelos de aprendizaje automático en una muestra de datos limitada, este se usa comúnmente en aprendizaje automático para comparar y seleccionar un modelo para un problema de modelado predictivo dado porque es fácil de entender, fácil de implementar y nos permite predecir el ajuste de un modelo a un hipotético conjunto de datos de prueba cuando no disponemos del conjunto explícito de este.

En particular, para comprobar la fiabilidad de los resultados que podemos obtener, se ha aplicado la técnica de validación conocida como *5-fold cross-validation*, consistente en dividir el conjunto de datos en 5 particiones disjuntas al 20 %.

El modelo se obtiene de entrenar hasta un total del 80 % de los datos disponibles, es decir, se consigue a partir de 4 de las 5 particiones obtenidas, y realizamos la validación con el 20 % restante. Dichas particiones se van alternando como conjunto de prueba consiguiendo hasta un total de cinco valores de porcentaje de clasificación en dicho conjunto, uno para cada partición que ha sido parte del conjunto de validación.

Los parámetros utilizados para evaluar el rendimiento de la clasificación son los siguientes:

- **Sensibilidad:** Porcentaje de retinas anormales clasificadas correctamente.

$$SP = \frac{TP}{TP + FN}$$

donde TP es el número de imágenes anormales clasificadas como enfermas, y FN es el número de imágenes normales encontradas como anormales, es decir, falsos negativos.

- **Especificidad:** Porcentaje de retinas sanas clasificadas correctamente.

$$SN = \frac{TN}{TN + FP}$$

donde TN es el número de imágenes normales clasificadas como sanas, y FP es el número de imágenes anormales encontradas como normal, son los llamados falsos positivos.

- **Exactitud:** Porcentajes de imágenes diagnosticadas correctamente de la cantidad total de imágenes.

$$A = \frac{TN + TP}{N}$$

siendo $N = TN + FN + TN + FP$ el número total de imágenes sanas y enfermas.

Claramente, cuanto mayor sean los valores de sensibilidad y especificidad, mejor será el diagnóstico obtenido.

Para estos experimentos, se han utilizado imágenes de la base de datos MESSIDOR¹, compuesta por 1200 imágenes capturadas usando 8 bits por plano de color de 1440×960 , 2240×1488 ó 2304×1536 píxeles, y proporcionadas por el departamento oftalmológico de *Service Ophtalmologie Lariboisière, LaTIM - CHU de BREST* y *CHU de St Etienne*. Estas imágenes han sido recortadas dependiendo de su tamaño y base de datos con ayuda de la función `imcrop`, eliminando así algunos de los píxeles innecesarios para reducir el tiempo computacional de cálculo.

Dicha bases de datos además proporciona dos diagnósticos para cada imagen proporcionados por expertos, el grado de retinopatía y el riesgo de edema macular. En este trabajo sólo nos interesará el primer parámetro, que se simplifica cambiando los números 1, 2 y 3 por 1 (imágenes anormales) y 0 (imágenes normales) en caso contrario.

Las muestras de entrenamiento han sido entrenadas a partir del clasificador SVM probando los distintos *kernels* que nos proporciona MATLAB, así como a partir de los árboles de decisión, y considerando o sólo las 3 características relacionadas con las áreas de las distintas lesiones típicas de la retinopatía diabética detectadas ó 7, añadiendo las asociadas a la textura, obtenidas a partir de la matriz de co-ocurrencia.

¹Los datos incluidos en esta base de datos pueden ser usados, de forma gratuita, con fines de investigación y educación. [9]

Para poder observar y comparar los datos² mejor se muestran dos tablas resumen con las medidas de sensibilidad, especificidad y exactitud aproximadamente obtenidas a partir de los distintos clasificadores y vectores de características para los cuales se ha realizado 5 pruebas considerando 200 imágenes aproximadamente con subconjuntos distintos a las que se ha aplicado la técnica de validación cruzada antes explicada. Dichos conjuntos de imágenes contenían aproximadamente el mismo número de retinografías de pacientes sanos y enfermos.

Base de datos primera

	Sensibilidad	Especificidad	Exactitud
SVM lineal	45.66 %	97.68 %	72.19 %
SVM gaussiano	55.77 %	95.25 %	76.29 %
SVM polinómico	62.18 %	92 %	77.14 %
Árboles de decisión	76.65 %	79.66 %	78.38 %

Tabla 5.1: (B1) Porcentajes obtenidos con el vector de características de tamaño 3.

	Sensibilidad	Especificidad	Exactitud
SVM lineal	45.66 %	98.37 %	72.95
SVM gaussiano	50.89 %	96.12 %	74.29 %
SVM polinómico	71.84 %	89.79 %	80.95 %
Árboles de decisión	75.77 %	78.07 %	76.95 %

Tabla 5.2: (B1) Porcentajes obtenidos con el vector de características de tamaño 7.

Los porcentajes de especificidad oscilan entre el 45 % y el 77 % dependiendo del clasificador y vector empleado, mientras que la especificidad oscila entre el 78 % y el 99 %, logrando una de exactitud de entre el 72 % y el 81 %. Producíendose mayoritariamente pequeñas mejoras al emplear los vectores que consideran las 7 características.

Se puede observar que existe una relación de costo entre la sensibilidad y la especificidad del detector, las mismas son inversamente proporcionales, lo que significa que a medida que aumenta la sensibilidad, la especificidad

²Las tablas completas con los resultados obtenidos tras realizar la validación cruzada se encuentran en el anexo 2, al final de este documento.

disminuye y viceversa. Por lo tanto, en este caso, no existe ningún clasificador que sobresalga notablemente sobre el resto sino que depende de la medida que se tenga en cuenta. Por ejemplo, el árbol de decisión con vector de 3 características obtiene mayor sensibilidad y el SVM con *kernel* lineal y vector de 7 características consigue la mejor tasa de especificidad, mientras el clasificador SVM con *kernel* polinómico obtiene mayor exactitud que el resto.

Estos errores en la clasificación se ven influenciados además por los producidos en los algoritmos de detección de las distintas lesiones de la retinopatía diabética, lo que provoca que el vector de características se vea afectado.

Base de datos segunda

	Sensibilidad	Especificidad	Exactitud
SVM lineal	56.12 %	90.96 %	69.62 %
SVM gaussiano	45.60 %	99.68 %	69.05 %
SVM polinómico	49.70 %	100 %	71.62 %
Árboles de decisión	99.25 %	98.92 %	99.14 %

Tabla 5.3: **(B2)** Porcentajes obtenidos con el vector de características de tamaño 3.

	Sensibilidad	Especificidad	Exactitud
SVM lineal	73.38 %	79.52 %	75.24 %
SVM gaussiano	60.35 %	93.90 %	74.76 %
SVM polinómico	73.09 %	98.84 %	84.29 %
Árboles de decisión	98.51 %	98.92 %	98.67 %

Tabla 5.4: **(B2)** Porcentajes obtenidos con el vector de características de tamaño 7.

Se podría decir que en un entorno clínico es interesante tener en cuenta la sensibilidad, que nos proporciona el porcentaje de imágenes enfermas correctamente detectadas. Las consecuencias de evaluar una retina sana incorrectamente implica simplemente repetir la prueba. Sin embargo, las consecuencias de evaluar una retina enferma incorrectamente pueden ser peligrosas. Pero también se debe considerar que repetir la prueba puede ser costoso, por lo tanto, sería deseable que se obtuviesen mejores resultados también en

sensibilidad.

En este caso, es claro que el mejor clasificador obtenido con este conjunto de prueba es el árbol de decisión, con aproximadamente un 99 % tanto en especificidad como en sensibilidad y, por tanto, en exactitud.

Capítulo 6

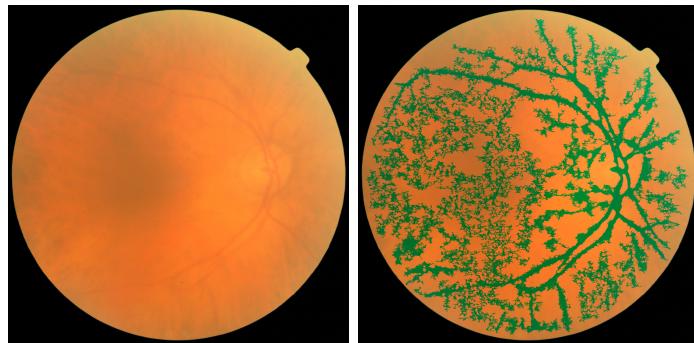
Conclusiones y trabajos futuros

En este trabajo se ha implementando un algoritmo cuyo proceso de detección y segmentación se ha realizado mediante el uso de técnicas de procesamiento de imágenes, aislando algunas de las lesiones típicas de la retinopatía diabética como son los exudados duros y los microaneurismas, y extrayendo además los vasos sanguíneos debido a que pueden haberse creado neovasos. Posteriormente ha sido entrenado el clasificador a partir de las características extraídas, incluyendo en algunos casos las relacionadas con la textura, permitiendo la clasificación de las imágenes de retina en sanas o enfermas.

Los datos fueron validados a partir de la técnica conocida como *5-fold cross-validation*, obteniendo buenos porcentajes principalmente en especificidad. Y durante este proceso se comprobó que a medida que aumenta la sensibilidad, la especificidad disminuye y viceversa.

Además se observó que el porcentaje de error de la clasificación se ve influenciado por los errores producidos al detectar los vasos sanguíneos, exudados duros y microaneurismas ya que estos afectan a los vectores de características necesarios para la clasificación, y estos a su vez depende de muchos parámetros que pueden ser convenientes para unas imágenes pero no para otras.

Algunos de estos errores se pueden ver a continuación:



(a) Imagen de la retina. (b) Imagen con los supuestos vasos sanguíneos resaltados.

Figura 6.1: Ejemplo detección incorrecta de los vasos sanguíneos.



(a) Imagen de la retina. (b) Imagen con los supuestos exudados duros resaltados.

Figura 6.2: Ejemplo detección incorrecta de los exudados duros.



(a) Imagen de la retina. (b) Imagen con los supuestos microaneurismas.

Figura 6.3: Ejemplo detección incorrecta de los microaneurismas.

Vías futuras

Existen una gran cantidad de técnicas de procesamiento de imágenes que podrían ser utilizadas para comprobar si se produce una mejora tanto en la clasificación de las distintas imágenes a partir de los vectores de características creados como en la detección de las lesiones típicas utilizadas en este caso, o incluso plantearse detectar otras como pueden ser los exudados blandos o algonosos.

En la mayoría de artículos leídos normalmente se utilizaba como clasificador el SVM pero podría probarse a entrenar los vectores de características usando otros más complejos, como son las redes neuronales convolucionadas o las redes bayesianas.

Por otro lado, también existen muchos trabajos relacionados con la detección de la retinopatía diabética, como es el propuesto por *Selvathi et al.*[17], así como de la detección de los vasos sanguíneos, exudados duros o microaneurismas. Un artículo para la detección de los vasos sanguíneos que me llamó la atención fue el propuesto por *Chaudhuri et al.*[30] que detectaba estos aplicando filtros sucesivamente, pero a causa del tiempo no se pudo comprobar la eficacia de este.

Además de mejorar los métodos a partir de otras propuestas ya realizadas por otros investigadores sería conveniente que este fuese supervisado por un experto en esta materia ya que permitiría incluso crear otro distinto en el cual se obtuviesen buenos resultados tanto en tiempo como en porcentaje de aciertos, proporcionando datos relacionados con las diferentes retinografías sanas y enfermas que pueden ser relevantes para una correcta clasificación.

Bibliografía

- [1] National Eye Institute (NEI). *Facts About Diabetic Eye Disease*. (18 de febrero de 2018)
<https://nei.nih.gov/health/diabetic/retinopathy>
- [2] Clínica Baviera. *Retinopatía diabética*. (30 de agosto de 2018)
<https://www.clinicabaviera.com/retinopatia-diabetica>
- [3] Piniés, José A. *Retinografía con cámara no midriática*. Seminarios de diabetes. Técnicas diagnósticas en diabetes (I), 2005.
- [4] National Eye Institute. *Retinopatía diabética*. (18 de febrero de 2018)
<https://nei.nih.gov/health/espanol/retinopatia>
- [5] Pascual, Rubén. *Retinopatía diabética: Proyecto divulgativo sobre la visión*. (10 de septiembre de 2008)
<https://ocularis.es/retinopatia-diabetica-i-conceptos/>
- [6] González, Rafael C. *Digital image processing*. 3rd ed. Harlow: Pearson Prentice Hall, 2008.
- [7] Chamorro Torres, Humberto; Encina Melgadero, José Santiago. *Detección Automática de Retinopatía Diabética basada en Técnicas de Procesamiento de Imágenes*. Facultad Politécnica Ingeniería en Informática - Universidad Nacional de Asunción, 2014.
- [8] Soporte de MATLAB. *Funciones*, 2018.
<https://es.mathworks.com/help/>
- [9] *Base de datos MESSIDOR*.
<http://www.adcis.net/en/Download-Third-Party/Messidor.html>
- [10] *Precios licencia de MATLAB*.
<https://es.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=comm>
- [11] *Image Processing Toolbox*.
<https://es.mathworks.com/products/image.html>

- [12] Fernández Valdivia, Joaquín. *Apuntes de la asignatura Procesado digital de imágenes*. Universidad de Granada.
- [13] Pérez de la Blanca Capilla, Nicolás. *Apuntes de la asignatura Visión por Computador*. Universidad de Granada, 2017/2018.
- [14] Maldonado Jurado, Juan Antonio; Román Román, Patricia. *Apuntes de la asignatura Estadística descriptiva e introducción a la probabilidad*. Universidad de Granada, 2013/2014.
- [15] Herrera Triguero, Francisco *Apuntes de la asignatura Metaheurística*. Universidad de Granada, 2016/2017.
- [16] Brownlee, Jason. *Introduction to k-fold Cross-Validation*. (23 de mayo de 2018)
[https://machinelearningmastery.com/
k-fold-cross-validation/](https://machinelearningmastery.com/k-fold-cross-validation/)
- [17] Selvathi D; Prakash, N.B.; Balagopal, Neethi. *Automated Detection of Diabetic Retinopathy for Early Diagnosis using Feature Extraction and Support Vector Machine*. International Journal of Emerging Technology and Advanced Engineering, noviembre de 2012.
- [18] SujithKumar S B*; Singh, Vipula *Automatic Detection of Diabetic Retinopathy in Non-dilated RGB Retinal Fundus Images*. International Journal of Computer Applications, junio de 2012.
- [19] López Peña, Antonio; Valveny, Ernest y Vanrell, Maria. *Curso sobre detección de objetos*. Universitat Autònoma de Barcelona, 2015.
- [20] Carmona Suárez, Enrique J. *Tutorial sobre Máquinas de Vectores Soporte (SVM)*. Universidad Nacional de Educación a Distancia, julio de 2014.
- [21] Pajares Martinsanz, Gonzalo; Cruz García, Jesús María de la. *Visión por computador imágenes digitales y aplicaciones*. Madrid RA-MA, 2001.
- [22] Gómez Flores, Wilfrido. *Análisis de imágenes digitales. Mejoramiento de la imagen: procesamiento del histograma*. (7 de junio de 2018)
[https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/
Clase11.pdf](https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase11.pdf)
- [23] Yuen, H.K; Princen, J.; Illingworth, J.; Kittler, J. *Image and Vision Computing (Comparative study of Hough transform methods for circle finding)*. Elsevier, 1990.
- [24] Encyclopedia of Mathematics. *Entropy*. (14 de agosto de 2018)
<https://www.encyclopediaofmath.org/index.php/Entropy>

- [25] Gómez Flores, Wilfrido. *Análisis de imágenes digitales. Filtrado de la imagen: Operadores morfológicos.* (7 de junio de 2018)
<https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase17.pdf>
- [26] Gómez Flores, Wilfrido. *Representación y descripción: Clasificadores supervisados.* (7 de junio de 2018)
<https://www.tamps.cinvestav.mx/~wgomez/diapositivas/AID/Clase35.pdf>
- [27] González, Rafael C; Woods, Richard E.; Eddins; Steven L. *Digital image processing using MATLAB.* Pearson Prentice Hall, 2004.
- [28] Roche, Ariel. *Árboles de decisión y Series de tiempo.* Facultad de Ingeniería - UDELAR, diciembre de 2009.
- [29] Laorden Fiter, Eduardo. *Descripción, comparación y ejemplos de uso de las funciones de la toolbox de procesado digital de imágenes de MATLAB.* Escuela Universitaria de Ingeniería Técnica de Telecomunicación - Universidad Politécnica de Madrid, 2012.
- [30] Chaudhuri, Subhasis; Chattenjee; Shankar; Katz, Norman; Nelson, Mark; Goldbaum, Michael. *Detection of Blood Vessels in Retinal Images Using Two-Dimensional Matched Filters.* IEEE transactions on medical imaging, septiembre de 1989.

Anexo 1

En este anexo se hace un resumen de las principales ideas de la teoría de optimización, orientado a la resolución de problemas asociados con el uso del clasificador SVM.

Definición 1 Se denomina **problema primal** al siguiente problema de optimización:

$$\begin{aligned} & \min f(x), \quad x \in \Omega \\ & s.a. \quad g_i(x) \leq 0, \quad i = 1, \dots, n \end{aligned}$$

Si todas las funciones f y g_i fueran lineales estaríamos ante un problema de programación lineal, mientras que si la función a optimizar es cuadrática pero las restricciones lineales, nos encontramos ante un problema de optimización cuadrática. La solución del problema primal, x^* , cumplirá que $g_i(x^*) \leq 0$ y $f(x^*) \leq f(x)$, $\forall x$ t.q. $g_i(x) \leq 0$, donde $i = 1, \dots, n$.

Definición 2 Se define la **función de Lagrange o lagrangiano** como:

$$L(x, \alpha) = f(x) + \sum_{i=1}^n \alpha_i g_i(x)$$

donde los coeficientes $\alpha \geq 0$ reciben el nombre de **multiplicadores de Lagrange**.

Los multiplicadores de Lagrange indican la dificultad de cumplir cada restricción, es decir, a mayor valor de α_i , más difícil será de cumplir la restricción asociada g_i . La función lagrangiana tiene la particularidad de incorporar la función objetivo y las funciones restricción en una única función.

Definición 3 A partir de la función de Lagrange se puede definir el **problema dual** como:

$$\begin{aligned} & \min \varphi(x) = \inf_{x \in \Omega} L(x, \alpha) \\ & s.a. \quad \alpha_i(x) \geq 0, \quad i = 1, \dots, n \end{aligned}$$

Normalmente el problema dual es más fácil que el primal y, bajo determinadas condiciones, al resolver el primero, podemos obtener también la solución del problema primal asociado.

Teorema 1 *Sean x y α vectores tales que satisfacen las restricciones respectivamente del problema primal y dual, entonces $\varphi(\alpha) \leq f(x)$.*

Corolario 1 *El problema dual está acotado superiormente por el problema primal.*

Corolario 2 *El problema dual está acotado superiormente por el problema primal.*

Corolario 3 *Si $\varphi(\alpha) = f(x)$, entonces α y x son soluciones, respectivamente, del problema dual y primal.*

Este teorema es de principal interés práctico ya que permite establecer una heurística para resolver, simultáneamente, el problema primal y dual. Así, estaremos más cerca de la solución, a medida que la diferencia $|\varphi(\alpha) - f(x)|$ sea pequeña, alcanzando la solución cuando la diferencia es nula.

El teorema de Karush-Kuhn-Tucke, que a continuación se enuncia, nos permite establecer las condiciones suficientes para que un punto x^* sea solución del problema primal.

Teorema 2 (Karush-Kuhn-Tucke) *Si en el problema primal las funciones $f : \mathbb{R}^d \rightarrow \mathbb{R}$ y $g_i : \mathbb{R}^d \rightarrow \mathbb{R}$, $i = 1, \dots, n$ son funciones convexas y, además, existen constantes $\alpha_i \geq 0$, $i = 1, \dots, n$ tales que:*

$$\frac{\partial f(x^*)}{\partial x_j} + \sum_{i=1}^n \alpha_i \frac{\partial g_i(x^*)}{\partial x_j} = 0, \quad j = 1, \dots, d$$
$$\alpha_i g_i(x^*) = 0, \quad i = 1, \dots, n$$

entonces el punto x^ es un mínimo global del problema primal.*

El interés de este teorema es que establece las condiciones que han de cumplirse para poder resolver el problema primal gracias al dual. Así, partiendo del problema primal, se construye el lagrangiano. Seguidamente, se aplica la primera condición del teorema de KKT a dicha función para obtener un conjunto de relaciones que, al ser sustituidas en la función lagrangiana, harán desaparecer todas las variables primales de dicha función. La función así obtenida sólo dependerá de los multiplicadores de Lagrange, y es posible que surjan restricciones adicionales para los multiplicadores de Lagrange. A partir de la solución del problema dual se puede resolver el primal también, bastará sustituir la solución dual en las relaciones anteriormente obtenidas al aplicar la primera condición KKT a la función lagrangiana.

Anexo 2

Esta parte del trabajo está reservada para los distintos resultados obtenidos en las diversas bases de datos que contienen imágenes de diversos tamaños. Resultados obtenidos a partir de los distintos clasificadores y vectores de características para los cuales se ha realizado 5 pruebas con subconjuntos distintos a los que se ha aplicado la técnica de validación cruzada.

Por simplicidad se emplean las siguientes siglas:

- **B_n:** Base de datos n , con $n = 1, 2$.
- **FVm:** vector de características de tamaño m , con $m = 3, 7$.

Base de datos primera

Imágenes de tamaño 2240×1488 originalmente recortadas, obteniendo una resolución de 1391×1396 .

SVM lineal

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.450000	0.526316	0.500000	0.777778	0.280000
Partición 2	0.500000	0.428571	0.333333	0.363636	0.500000
Partición 3	0.550000	0.294118	0.521739	0.333333	0.545455
Partición 4	0.300000	0.555556	0.473684	0.363636	0.470588
Partición 5	0.476190	0.346154	0.428571	0.444444	0.533333
Media	0.455238	0.430143	0.451466	456566	0.465875

Tabla 8.1: (B1) Sensibilidad - SVM lineal y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.352941	0.347826	0.400000	0.523810	0.437500
Partición 2	0.409091	0.529412	0.391304	0.411765	0.384615
Partición 3	0.523810	0.312500	0.631579	0.550000	0.500000
Partición 4	0.550000	0.521739	0.526316	0.437500	0.555556
Partición 5	0.428571	0.500000	0.400000	0.407407	0.380952
Media	0.452883	0.442295	0.469840	0.466096	0.451725

Tabla 8.2: (B1) Sensibilidad - SVM lineal y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	0.956522	1.000000	1.000000	0.941176
Partición 2	1.000000	1.000000	0.958333	1.000000	0.950000
Partición 3	1.000000	0.960000	1.000000	1.000000	1.000000
Partición 4	0.954545	0.916667	0.913043	1.000000	1.000000
Partición 5	0.952381	1.000000	1.000000	0.916667	1.000000
Media	0.981385	0.966638	0.974275	0.983333	0.978235

Tabla 8.3: (B1) Especificidad - SVM lineal y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	1.000000	0.923077
Partición 2	0.950000	0.960000	0.947368	1.000000	1.000000
Partición 3	1.000000	1.000000	0.956522	0.954545	1.000000
Partición 4	1.000000	0.947368	1.000000	1.000000	1.000000
Partición 5	1.000000	1.000000	1.000000	1.000000	0.952381
Media	0.990000	0.981474	0.980778	0.990909	0.975092

Tabla 8.4: (B1) Especificidad - SVM lineal y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.738095	0.761905	0.761905	0.904762	0.547619
Partición 2	0.761905	0.714286	0.690476	0.666667	0.714286
Partición 3	0.785714	0.690476	0.738095	0.666667	0.761905
Partición 4	0.642857	0.761905	0.714286	0.666667	0.785714
Partición 5	0.714286	0.595238	0.714286	0.714286	0.833333
Media	0.728571	0.704762	0.723810	0.723810	0.728571

Tabla 8.5: (B1) Exactitud - SVM lineal y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.738095	0.642857	0.714286	0.761905	0.738095
Partición 2	0.666667	0.785714	0.642857	0.761905	0.619048
Partición 3	0.761905	0.738095	0.809524	0.761905	0.761905
Partición 4	0.785714	0.714286	0.785714	0.785714	0.809524
Partición 5	0.714286	0.738095	0.714286	0.619048	0.666667
Media	0.733333	0.723810	0.733333	0.738095	0.719048

Tabla 8.6: (B1) Exactitud - SVM lineal y FV7.

SVM gaussiano

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.619048	0.444444	0.541667	0.687500	0.684211
Partición 2	0.470588	0.666667	0.562500	0.608696	0.529412
Partición 3	0.800000	0.500000	0.619048	0.391304	0.550000
Partición 4	0.333333	0.473684	0.550000	0.526316	0.608696
Partición 5	0.560000	0.611111	0.500000	0.650000	0.454545
Media	0.556594	0.539181	0.554643	0.572763	0.565373

Tabla 8.7: (B1) Sensibilidad - SVM gaussiano y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.550000	0.700000	0.631579	0.428571	0.636364
Partición 2	0.523810	0.380952	0.476190	0.500000	0.444444
Partición 3	0.409091	0.444444	0.434783	0.347826	0.478261
Partición 4	0.466667	0.409091	0.500000	0.590909	0.318182
Partición 5	0.695652	0.600000	0.500000	0.631579	0.625000
Media	0.529044	0.506898	0.508510	0.499777	0.500450

Tabla 8.8: (B1) Sensibilidad - SVM gaussiano y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.952381	0.958333	1.000000	0.923077	0.956522
Partición 2	0.960000	0.944444	0.961538	1.000000	1.000000
Partición 3	1.000000	1.000000	0.952381	0.947368	1.000000
Partición 4	0.875000	0.956522	0.954545	0.956522	0.842105
Partición 5	0.941176	0.916667	0.909091	0.954545	0.950000
Media	0.945711	0.955193	0.955511	0.956303	0.949725

Tabla 8.9: (B1) Especificidad - SVM gaussiano y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.909091	1.000000	0.956522	1.000000	0.900000
Partición 2	1.000000	0.952381	0.857143	0.961538	1.000000
Partición 3	1.000000	0.958333	1.000000	0.842105	0.947368
Partición 4	0.925926	1.000000	1.000000	1.000000	0.950000
Partición 5	1.000000	0.909091	1.000000	1.000000	0.961538
Media	0.967003	0.963961	0.962733	0.960729	0.951781

Tabla 8.10: (B1) Especificidad - SVM gaussiano y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.785714	0.738095	0.738095	0.833333	0.833333
Partición 2	0.761905	0.785714	0.809524	0.785714	0.809524
Partición 3	0.904762	0.738095	0.785714	0.642857	0.785714
Partición 4	0.642857	0.738095	0.761905	0.761905	0.714286
Partición 5	0.714286	0.785714	0.714286	0.809524	0.690476
Media	0.761905	0.757143	0.761905	0.766667	0.766667

Tabla 8.11: (B1) Exactitud - SVM gaussiano y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.738095	0.857143	0.809524	0.714286	0.761905
Partición 2	0.761905	0.666667	0.666667	0.785714	0.761905
Partición 3	0.690476	0.738095	0.690476	0.571429	0.690476
Partición 4	0.761905	0.690476	0.785714	0.785714	0.619048
Partición 5	0.833333	0.761905	0.761905	0.833333	0.833333
Media	0.757143	0.742857	0.742857	0.738095	0.733333

Tabla 8.12: (B1) Exactitud - SVM gaussiano y FV7.

SVM polinómico de grado 4

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.388889	0.736842	0.700000	0.461538	0.714286
Partición 2	0.666667	0.714286	0.666667	0.476190	0.600000
Partición 3	0.640000	0.636364	0.647059	0.789474	0.476190
Partición 4	0.687500	0.478261	0.681818	0.650000	0.625000
Partición 5	0.666667	0.521739	0.518519	0.800000	0.600000
Media	0.609944	0.617498	0.642812	0.635441	0.603095

Tabla 8.13: (B1) Sensibilidad - SVM polinómico y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.705882	0.947368	0.789474	0.666667	0.652174
Partición 2	0.722222	0.650000	0.619048	0.653846	0.750000
Partición 3	0.640000	0.772727	0.777778	0.650000	0.625000
Partición 4	0.916667	0.800000	0.666667	0.800000	0.714286
Partición 5	0.588235	0.450000	0.636364	0.823529	0.941176
Media	0.714601	0.724019	0.697866	0.718808	0.736527

Tabla 8.14: **(B1)** Sensibilidad - SVM polinómico y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	0.913043	0.954545	1.000000	0.952381
Partición 2	0.904762	0.892857	0.925926	0.952381	0.888889
Partición 3	0.941176	0.900000	0.880000	0.782609	0.904762
Partición 4	0.923077	0.894737	0.850000	0.909091	0.944444
Partición 5	0.857143	0.947368	1.000000	0.925926	0.954545
Media	0.925232	0.909601	0.922094	0.914001	0.929004

Tabla 8.15: **(B1)** Especificidad - SVM polinómico y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.800000	0.956522	0.956522	0.791667	0.894737
Partición 2	0.958333	1.000000	0.809524	0.875000	0.923077
Partición 3	0.941176	0.850000	0.958333	0.863636	0.944444
Partición 4	0.888889	0.863636	0.857143	1.000000	0.952381
Partición 5	0.840000	0.863636	0.900000	0.920000	0.840000
Media	0.885680	0.906759	0.896304	0.890061	0.910928

Tabla 8.16: **(B1)** Especificidad - SVM polinómico y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.738095	0.833333	0.833333	0.666667	0.833333
Partición 2	0.785714	0.833333	0.833333	0.714286	0.785714
Partición 3	0.761905	0.761905	0.785714	0.785714	0.690476
Partición 4	0.833333	0.666667	0.761905	0.785714	0.761905
Partición 5	0.761905	0.714286	0.690476	0.880952	0.785714
Media	0.776190	0.761905	0.780952	0.766667	0.771429

Tabla 8.17: **(B1)** Exactitud - SVM polinómico y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.761905	0.952381	0.880952	0.738095	0.761905
Partición 2	0.857143	0.833333	0.714286	0.738095	0.857143
Partición 3	0.761905	0.809524	0.880952	0.761905	0.761905
Partición 4	0.904762	0.833333	0.761905	0.904762	0.833333
Partición 5	0.738095	0.666667	0.761905	0.880952	0.880952
Media	0.804762	0.819048	0.800000	0.804762	0.819048

Tabla 8.18: **(B1)** Exactitud - SVM polinómico y FV7.

Árbol de decisión

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.705882	0.750000	0.700000	0.730769	0.750000
Partición 2	0.708333	0.823529	0.590909	0.857143	0.789474
Partición 3	0.789474	0.684211	0.687500	0.700000	0.809524
Partición 4	0.700000	0.761905	0.789474	0.826087	0.650000
Partición 5	1.000000	0.833333	0.833333	0.833333	0.857143
Media	0.780738	0.770596	0.720243	0.789466	0.771228

Tabla 8.19: **(B1)** Sensibilidad - Árbol de decisión y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.608696	0.842105	0.739130	0.650000	0.750000
Partición 2	0.652174	0.772727	0.785714	0.764706	0.708333
Partición 3	0.823529	0.818182	0.818182	0.761905	0.909091
Partición 4	0.750000	0.850000	0.692308	0.842105	0.722222
Partición 5	0.777778	0.722222	0.625000	0.791667	0.764706
Media	0.722435	0.801047	0.732067	0.762077	0.770870

Tabla 8.20: (B1) Sensibilidad - Árbol de decisión y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.720000	0.818182	0.818182	0.750000	0.818182
Partición 2	0.888889	0.880000	0.750000	0.857143	0.739130
Partición 3	0.782609	0.869565	0.884615	0.818182	0.809524
Partición 4	0.681818	0.904762	0.913043	0.894737	0.863636
Partición 5	0.809524	0.444444	0.722222	0.666667	0.809524
Media	0.776568	0.783391	0.817613	0.797346	0.807999

Tabla 8.21: (B1) Especificidad - Árbol de decisión y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.789474	0.826087	0.894737	0.818182	0.681818
Partición 2	0.894737	0.900000	0.714286	0.800000	0.777778
Partición 3	0.800000	0.650000	0.750000	0.761905	0.500000
Partición 4	0.818182	0.681818	0.750000	0.739130	0.833333
Partición 5	0.750000	0.791667	0.961538	0.833333	0.800000
Media	0.810478	0.769914	0.814112	0.790510	0.718586

Tabla 8.22: (B1) Especificidad - Árbol de decisión y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.714286	0.785714	0.761905	0.738095	0.785714
Partición 2	0.785714	0.857143	0.666667	0.857143	0.761905
Partición 3	0.785714	0.785714	0.809524	0.761905	0.809524
Partición 4	0.690476	0.833333	0.857143	0.857143	0.761905
Partición 5	0.904762	0.666667	0.785714	0.738095	0.833333
Media	0.776190	0.785714	0.776190	0.790476	0.790476

Tabla 8.23: **(B1)** Exactitud - Árbol de decisión y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.690476	0.833333	0.809524	0.738095	0.714286
Partición 2	0.761905	0.833333	0.738095	0.785714	0.738095
Partición 3	0.809524	0.738095	0.785714	0.761905	0.714286
Partición 4	0.785714	0.761905	0.714286	0.785714	0.785714
Partición 5	0.761905	0.761905	0.833333	0.809524	0.785714
Media	0.761905	0.785714	0.776190	0.776190	0.747619

Tabla 8.24: **(B1)** Exactitud - Árbol de decisión y FV7.

Base de datos segunda

Imágenes de tamaño 1440×960 originalmente recortadas, obteniendo una resolución de 926×921 .

SVM lineal

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.521739	0.423077	0.304348	0.652174	0.190476
Partición 2	0.333333	0.478261	1.000000	0.550000	0.809524
Partición 3	0.625000	0.521739	0.500000	0.357143	0.576923
Partición 4	1.000000	0.681818	0.478261	1.000000	0.392857
Partición 5	0.360000	0.652174	0.521739	0.384615	0.714286
Media	0.568014	0.551414	0.560870	0.588786	0.536813

Tabla 8.25: **(B2)** Sensibilidad - SVM lineal y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.500000	0.684211	0.772727	0.608696	0.655172
Partición 2	0.760000	0.818182	0.739130	0.851852	0.769231
Partición 3	0.944444	0.565217	0.782609	0.772727	0.909091
Partición 4	0.750000	0.769231	0.739130	0.904762	0.800000
Partición 5	0.807692	0.777778	0.653846	0.458333	0.550000
Media	0.752427	0.722924	0.737489	0.719274	0.736699

Tabla 8.26: (B2) Sensibilidad - SVM lineal y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	0.947368	1.000000
Partición 2	1.000000	1.000000	0.038462	1.000000	0.904762
Partición 3	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 4	0.714286	1.000000	1.000000	0.136364	1.000000
Partición 5	1.000000	1.000000	1.000000	1.000000	1.000000
Media	0.942857	1.000000	0.807692	0.816746	0.980952

Tabla 8.27: (B2) Especificidad - SVM lineal y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	0.521739	0.550000	0.947368	0.923077
Partición 2	0.588235	0.850000	0.842105	1.000000	1.000000
Partición 3	0.500000	1.000000	0.947368	0.550000	0.600000
Partición 4	0.888889	0.937500	0.789474	0.523810	0.545455
Partición 5	0.875000	0.800000	0.937500	0.944444	0.818182
Media	0.770425	0.821848	0.813289	0.793124	0.777343

Tabla 8.28: (B2) Especificidad - SVM lineal y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.738095	0.642857	0.619048	0.785714	0.595238
Partición 2	0.619048	0.714286	0.404762	0.785714	0.857143
Partición 3	0.785714	0.738095	0.619048	0.571429	0.738095
Partición 4	0.857143	0.833333	0.714286	0.547619	0.595238
Partición 5	0.619048	0.809524	0.738095	0.619048	0.857143
Media	0.723810	0.747619	0.619048	0.661905	0.728571

Tabla 8.29: **(B2)** Exactitud - SVM lineal y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.714286	0.595238	0.666667	0.761905	0.738095
Partición 2	0.690476	0.833333	0.785714	0.904762	0.857143
Partición 3	0.690476	0.761905	0.857143	0.666667	0.761905
Partición 4	0.809524	0.833333	0.761905	0.714286	0.666667
Partición 5	0.833333	0.785714	0.761905	0.666667	0.690476
Media	0.747619	0.761905	0.766667	0.742857	0.742857

Tabla 8.30: **(B2)** Exactitud - SVM lineal y FV7.

SVM gaussiano

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.571429	0.428571	0.565217	0.681818	0.392857
Partición 2	0.333333	0.500000	0.550000	0.407407	0.428571
Partición 3	0.550000	0.448276	0.291667	0.423077	0.391304
Partición 4	0.423077	0.611111	0.333333	0.240000	0.500000
Partición 5	0.384615	0.320000	0.500000	0.647059	0.476190
Media	0.452491	0.461592	0.448043	0.479872	0.437785

Tabla 8.31: **(B2)** Sensibilidad - SVM gaussiano y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.583333	0.454545	0.500000	0.571429	0.692308
Partición 2	0.565217	0.500000	0.666667	0.708333	0.520000
Partición 3	0.560000	0.772727	0.653846	0.578947	0.894737
Partición 4	0.842105	0.521739	0.416667	0.761905	0.400000
Partición 5	0.576923	0.730769	0.625000	0.400000	0.590909
Media	0.625516	0.595956	0.572436	0.604123	0.619591

Tabla 8.32: **(B2)** Sensibilidad - SVM gaussiano y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 2	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 3	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 4	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 5	1.000000	1.000000	1.000000	1.000000	1.000000
Media	1.000000	1.000000	1.000000	0.984000	1.000000

Tabla 8.33: **(B2)** Especificidad - SVM gaussiano y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	0.950000	0.950000	1.000000	0.875000
Partición 2	1.000000	1.000000	0.904762	0.944444	1.000000
Partición 3	1.000000	1.000000	0.937500	0.913043	0.782609
Partición 4	0.869565	0.894737	1.000000	0.904762	1.000000
Partición 5	1.000000	0.937500	0.777778	0.882353	0.950000
Media	0.973913	0.956447	0.914008	0.928921	0.921522

Tabla 8.34: **(B2)** Especificidad - SVM gaussiano y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.785714	0.714286	0.761905	0.833333	0.595238
Partición 2	0.619048	0.714286	0.785714	0.619048	0.714286
Partición 3	0.785714	0.619048	0.595238	0.642857	0.666667
Partición 4	0.642857	0.833333	0.619048	0.547619	0.714286
Partición 5	0.619048	0.595238	0.690476	0.809524	0.738095
Media	0.690476	0.695238	0.690476	0.690476	0.685714

Tabla 8.35: (B2) Exactitud - SVM gaussiano y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.761905	0.690476	0.714286	0.714286	0.761905
Partición 2	0.761905	0.714286	0.785714	0.809524	0.714286
Partición 3	0.738095	0.880952	0.761905	0.761905	0.833333
Partición 4	0.857143	0.690476	0.666667	0.833333	0.642857
Partición 5	0.738095	0.809524	0.690476	0.595238	0.761905
Media	0.771429	0.757143	0.723810	0.742857	0.742857

Tabla 8.36: (B2) Exactitud - SVM gaussiano y FV7.

SVM polinómico de grado 4

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.423077	0.583333	0.619048	0.608696	0.480000
Partición 2	0.500000	0.260870	0.625000	0.304348	0.344828
Partición 3	0.625000	0.520000	0.360000	0.500000	0.500000
Partición 4	0.363636	0.480000	0.346154	0.458333	0.583333
Partición 5	0.571429	0.550000	0.523810	0.560000	0.733333
Media	0.496628	0.478841	0.494802	0.486275	0.528299

Tabla 8.37: (B2) Sensibilidad - SVM polinómico y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.727273	0.681818	0.750000	0.620690	0.800000
Partición 2	0.739130	0.681818	0.750000	0.680000	0.695652
Partición 3	0.750000	0.655172	0.761905	0.800000	0.800000
Partición 4	0.736842	0.750000	0.730769	0.772727	0.727273
Partición 5	0.680000	0.900000	0.727273	0.687500	0.666667
Media	0.726649	0.733762	0.743989	0.712183	0.737918

Tabla 8.38: (B2) Sensibilidad - SVM polinómico y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 2	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 3	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 4	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 5	1.000000	1.000000	1.000000	1.000000	1.000000
Media	1.000000	1.000000	1.000000	1.000000	1.000000

Tabla 8.39: (B2) Especificidad - SVM polinómico y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.950000	1.000000	1.000000	1.000000	1.000000
Partición 2	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 3	1.000000	1.000000	0.952381	1.000000	1.000000
Partición 4	0.956522	1.000000	1.000000	0.900000	0.950000
Partición 5	1.000000	1.000000	1.000000	1.000000	1.000000
Media	0.981304	1.000000	0.990476	0.980000	0.990000

Tabla 8.40: (B2) Especificidad - SVM polinómico y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.642857	0.761905	0.809524	0.785714	0.690476
Partición 2	0.714286	0.595238	0.785714	0.619048	0.547619
Partición 3	0.785714	0.714286	0.619048	0.738095	0.714286
Partición 4	0.666667	0.690476	0.595238	0.690476	0.761905
Partición 5	0.785714	0.785714	0.761905	0.738095	0.904762
Media	0.719048	0.709524	0.714286	0.714286	0.723810

Tabla 8.41: **(B2)** Exactitud - SVM polinómico y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.833333	0.833333	0.833333	0.738095	0.904762
Partición 2	0.857143	0.833333	0.880952	0.809524	0.833333
Partición 3	0.833333	0.761905	0.857143	0.880952	0.880952
Partición 4	0.857143	0.857143	0.833333	0.833333	0.833333
Partición 5	0.809524	0.952381	0.857143	0.880952	0.785714
Media	0.838095	0.847619	0.852381	0.828571	0.847619

Tabla 8.42: **(B2)** Exactitud - SVM polinómico y FV7.

Árbol de decisión

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	0.960000	1.000000
Partición 2	0.954545	0.947368	1.000000	1.000000	1.000000
Partición 3	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 4	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 5	1.000000	1.000000	0.950000	1.000000	1.000000
Media	0.990909	0.989474	0.990000	0.992000	1.000000

Tabla 8.43: **(B2)** Sensibilidad - Árbol de decisión y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.961538	1.000000	1.000000	0.964286	1.000000
Partición 2	1.000000	1.000000	0.952381	1.000000	1.000000
Partición 3	0.952381	0.947368	1.000000	0.960000	1.000000
Partición 4	1.000000	1.000000	1.000000	0.961538	1.000000
Partición 5	0.965517	0.961538	1.000000	1.000000	1.000000
Media	0.975887	0.981781	0.990476	0.977165	1.000000

Tabla 8.44: (B2) Sensibilidad - Árbol de decisión y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	1.000000	1.000000
Partición 2	1.000000	1.000000	1.000000	0.944444	0.952381
Partición 3	0.937500	0.952381	1.000000	1.000000	1.000000
Partición 4	1.000000	1.000000	0.944444	1.000000	1.000000
Partición 5	1.000000	1.000000	1.000000	1.000000	1.000000
Media	0.987500	0.990476	0.988889	0.988889	0.990476

Tabla 8.45: (B2) Especificidad - Árbol de decisión y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	1.000000	0.944444
Partición 2	0.937500	1.000000	1.000000	0.956522	1.000000
Partición 3	1.000000	1.000000	0.941176	1.000000	1.000000
Partición 4	1.000000	0.950000	1.000000	1.000000	1.000000
Partición 5	1.000000	1.000000	1.000000	1.000000	1.000000
Media	0.987500	0.990000	0.988235	0.991304	0.988889

Tabla 8.46: (B2) Especificidad - Árbol de decisión y FV7.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	1.000000	1.000000	1.000000	0.976190	1.000000
Partición 2	0.976190	0.976190	1.000000	0.976190	0.976190
Partición 3	0.976190	0.976190	1.000000	1.000000	1.000000
Partición 4	1.000000	1.000000	0.976190	1.000000	1.000000
Partición 5	1.000000	1.000000	0.976190	1.000000	1.000000
Media	0.990476	0.990476	0.990476	0.990476	0.995238

Tabla 8.47: **(B2)** Exactitud - Árbol de decisión y FV3.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
Partición 1	0.976190	1.000000	1.000000	0.976190	0.976190
Partición 2	0.976190	1.000000	0.976190	0.976190	1.000000
Partición 3	0.976190	0.976190	0.976190	0.976190	1.000000
Partición 4	1.000000	0.976190	1.000000	0.976190	1.000000
Partición 5	0.976190	0.976190	1.000000	1.000000	1.000000
Media	0.980952	0.985714	0.990476	0.980952	0.995238

Tabla 8.48: **(B2)** Exactitud - Árbol de decisión y FV7.