Universidad San Pablo-CEU

Escuela Politécnica Superior
Ingeniería de Telecomunicación

Proyecto Final de Carrera

Detección Automática de Partículas en Micrografías

Realizado por
Enrique Recarte Lloréns

Dirigido por
Carlos Óscar Sánchez Sorzano

Madrid, Julio 2008

Resumen

Este proyecto se desarrolla en el ámbito de la reconstrucción tridimensional de partículas (Tomografía Tridimensional). Dicha técnica permite crear un modelo en 3D a partir de muchas proyecciones 2D de un tipo de partícula concreto. Estas proyecciones se realizan con un Microscopio Electrónico de Transmisión (TEM), el cual produce una imagen (Micrografía) a partir de radiaciones de electrones sobre la muestra. Estas radiaciones han de ser de muy baja intensidad para no dañar la muestra, lo cual provoca que la micrografía resultante contenga mucho ruido, teniendo que utilizar muchas proyecciones (alrededor de 100.000) para poder obtener un modelo tridimensional con la suficiente resolución (5 Å). Estas reconstrucciones tridimensionales son muy útiles para los biólogos, ya que ayudan a estudiar las formas de las moléculas, pudiendo así determinar cómo se relacionan unas con otras.

Uno de los pasos intermedios para esta reconstrucción tridimensional consiste en indicar para cada micrografía la posición exacta de las partículas de las que queremos hacer la reconstrucción. Debido al gran número de micrografías necesarias para poder realizar una reconstrucción con la suficiente resolución, este paso intermedio se ha convertido en uno de los principales cuellos de botella en el proceso global, puesto que se requiere a una persona física marcando todas las partículas para cada una de las micrografías.

En este proyecto se propone una aproximación a este problema mediante el uso de un modelo matemático basado en Redes Bayesianas, el cual ha demostrado ser un buen clasificador en varios ámbitos, así como simple computacionalmente hablando. Esta aproximación ha sido implementada dentro del conjunto de programas XMIPP [27], el cual dispone de una aplicación para marcar las partículas de forma manual. El objetivo será modificar este programa para que sea capaz de distinguir partículas en una micrografía y así ahorrar tiempo en el marcado de todas las partículas.

Al final de este documento, se exponen los resultados obtenidos, así como posibles mejoras futuras aplicables a este modelo.

Abstract

The project is developed in the Single Particle Analysis domain. This technique is used to make 3D model reconstructions from several 2D projections of some set of particles. In order to make this reconstruction, a Transmission Electron Microscopy (TEM) is used. This TEM creates an output image (Micrograph) by radiating the specimen with an electron beam. In order to avoid radiation damage, the dose of electrons has to be very low, which lead to very noisy images. Because of these noisy images, the number of projections in order to get good resolution 3D models have to be very large (around 100.000 projections for about 5 Å). These 3D models aid the biologists study some relevant information about the molecules like the shape which gives information about how the molecules interact with each other.

One of the intermediate steps of the reconstruction consists of manually picking all the particles present in a micrograph. Because of the large amount of micrographs needed for a good reconstruction, this step is a big bottleneck since it needs a person manually picking all these particles.

An approach to this problem is proposed by using a mathematical model which automatically detects the particles in a micrograph. This mathematical model is based on Bayesian Networks, which have proved to be good classifiers in many domains while not being computationally expensive. This approach has been implemented within the XMIPP [27] package. This package contains an application which lets the user mark manually all the particles present in a certain micrograph. The purpose of this project is to add a functionality to this application to automatically detect these particles and therefore make the whole reconstruction process faster and less tedious.

Throughout this document, the results and conclusions are exposed as well as some possible future steps.

Índice general

Índice de figuras

Índice de tablas

Estructura del Documento

Este documento se estructura de la siguiente manera:

- El Capítulo 1 es una breve **introducción** a la microscopía electrónica y a los elementos utilizados en este proyecto.
- En el Capítulo 2 se expone cómo se encuentra el **Estado del Arte** de esta materia, enumerando diversos trabajos relacionados con este y haciendo una comparativa entre ellos.
- Los Capítulos 3 y 4 explican el **modelo utilizado** para abordar el problema, y se entra en detalle en el concepto de las **Redes Bayesianas**, utilizadas para abordar la implementación.
- La **implementación** se explica en el Capítulo 5. Se explica cómo se ha implementado la solución, cuáles han sido las fases de desarrollo, los problemas encontrados y las soluciones para éstos.
- En el Capítulo 6 discutimos los **resultados obtenidos** y hacemos una comparativa con otros trabajos similares.
- Por último, en el Capítulo 7 comentamos las **conclusiones**, así como posibles **mejoras futuras** adicionales a nuestro proyecto para intentar conseguir mejores resultados.

Capítulo 1

Introducción

1.1. Introducción a la microscopía electrónica

Este proyecto se desarrolla en el marco de la biología molecular. Más concretamente, en la biología estructural, la cual estudia la estructura y disposición espacial de complejos moleculares. Dicha distribución espacial es de vital importancia, ya que en este campo se asume que existe una estrecha relación entre las estructuras terciaria (estructura tridimensional de una molécula) y cuaternaria (distribución espacial de un complejo molecular) y la función realizada por el complejo molecular. Esta relación cobra importancia en determinados campos que se dedican a la investigación, por ejemplo, de nuevos medicamentos, ya que cuanto más se sepa sobre cómo interactúan unas moléculas con otras, más fácil será hacer nuevos medicamentos más eficientes.

Para este estudio, existe una técnica conocida como Tomografía Tridimensional de Complejos Macromoleculares, la cual estudia modelos en 3 dimensiones a partir de datos en 2 dimensiones. Lo que la tomografía intenta conseguir es la construcción de un modelo tridimensional ontenido mediante medios computacionales a partir de varias muestras bidimensionales de una partícula (imágenes microscópicas). Estas muestras bidimensionales pueden ser obtenidas de diversas maneras (rayos X, resonancia magnética, microscopía electrónica, etc).

Históricamente, el TEM siempre había sido considerado como una herramienta poco efectiva comparada con los rayos X, debido la gran resolución que se podía llegar a alcanzar con esta última técnica, aunque requiere que la muestra estudiada forme cristal, lo cual no es fácil, especialmente si la muestra es muy grande. En cambio, el TEM no impone ningún tipo de restricción ni al tamaño ni al estado de la muestra. Sin embargo,

la resolución alcanzada por este instrumento es menor, ya que para no dañar la muestra, la dósis de elcectrones radiados sobre la muestra ha de ser muy baja, provocando una imagen resultante con una relación señal a ruido (SNR) muy baja. Es por esto que las técnicas de procesamiento de imágenes han cobrado una vital importancia en este campo, ya que se requiere mejorar esta relación de calidad para poder conseguir una reconstrucción tridimensional con una resolución aceptable. En los últimos años se han conseguido avances significativos consiguiendo resoluciones comprendidas entre los 6 Å y los 50 Å gracias a la aplicación de avanzados algoritmos de *denoising*.

1.1.1. Funcionamiento de un microscopio electrónico

La potencia amplificadora de un microscopio óptico está limitada por la longitud de onda de la luz visible. El microscopio electrónico utiliza electrones para iluminar un objeto. Dado que los electrones tienen una longitud de onda mucho menor que la de la luz pueden mostrar estructuras mucho más pequeñas. La longitud de onda más corta de la luz visible es de alrededor de 4.000 Å (1 Å es 0,0000000001 metros). La longitud de onda de los electrones que se utilizan en los microscopios electrónicos es de alrededor de 0,5 Å.

Todos los microscopios electrónicos cuentan con varios elementos básicos. Disponen de un cañón de electrones que emite los electrones que chocan contra el espécimen, creando una imagen aumentada. Se utilizan lentes magnéticas para crear campos que dirigen y enfocan el haz de electrones, ya que las lentes convencionales utilizadas en los microscopios ópticos no funcionan con los electrones. El sistema de vacío es una parte relevante del microscopio electrónico. Los electrones pueden ser desviados por las moléculas del aire, de forma que tiene que hacerse un vacío casi total en el interior de un microscopio de estas características. Por último, todos los microscopios electrónicos cuentan con un sistema que registra o muestra la imagen que producen los electrones.

1.1.2. Microscopios electrónicos existentes

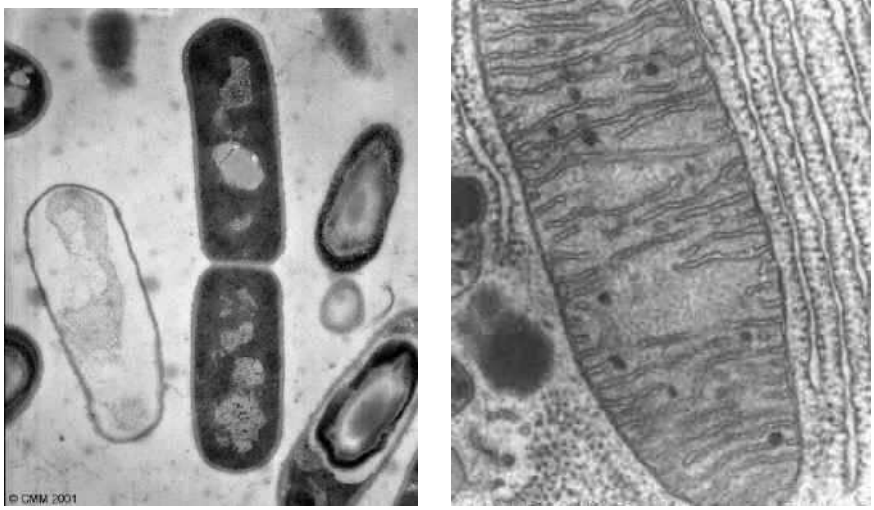
Existen dos tipos de microscopios electrónicos: el microscopio electrónico de transmisión (Transmission Electron Microscope, TEM) y el microscopio electrónico de barrido (Scanning Electron Microscope, SEM).

- Microscopio Electrónico de Transmisión (TEM): este tipo de microscopios permite observar una muestra en cortes ultrafinos. Para ello, dirige un haz de electrones

Figura 1.1: Microscopio Electrónico de Transmisión

hacia la muestra en cuestión, de manera que algunos electrones reboten, otros sean absorbidos por la muestra y otros lo atraviesen. Estos últimos son los que forman la imagen aumentada de la muestra, ya que debajo de la muestra se coloca una pantalla fluorescente para registrar la imagen aumentada gracias a estos electrones. Para poder utilizar un TEM, debe cortarse la muestra en capas muy finas, no mayores de unos pocos miles de Å. Los resultados obtenidos con este tipo de microscopio son de alrededor de un millón de aumentos. En la Figura 1.2 podemos ver dos ejemplos de imágenes obtenidas a través de un TEM.

- Microscopio Electrónico de Barrido (SEM): En este microscopio, en cambio, no hacen falta tantos preparativos sobre la muestra, ya que no hace falta cortarla en capas finas, ya que busca más bien representar la superficie de la muestra. Se explora la superficie de la muestra punto por punto (el TEM examina partes de la muestra cada vez) con un haz muy concentrado de electrones. Su funcionamiento es similar al de una televisión, en la cual existe un haz de electrones que barre la pantalla de televisión creando la imagen resultante. Este haz de electrones puede dispersarse sobre la muestra o bien provocar la aparición de nuevos elctrones secundarios. Todos estos electrones (perdidos y generados) son contados por un dispositivo electrónico situado a los lados de la muestra. Cada punto de la muestra se corresponde con un píxel de la pantalla. Cuanto mayor sea el número de electrones contado por el dispositivo electrónico, mayor será el brillo del píxel en la pantalla. El haz de electrones se va desplazando por la muestra, generando la imagen resultante del microscopio en la pantalla. Estos microscopios pueden lle-



(a) Bacilos en División (b) 60.000X MET, Mitocondria

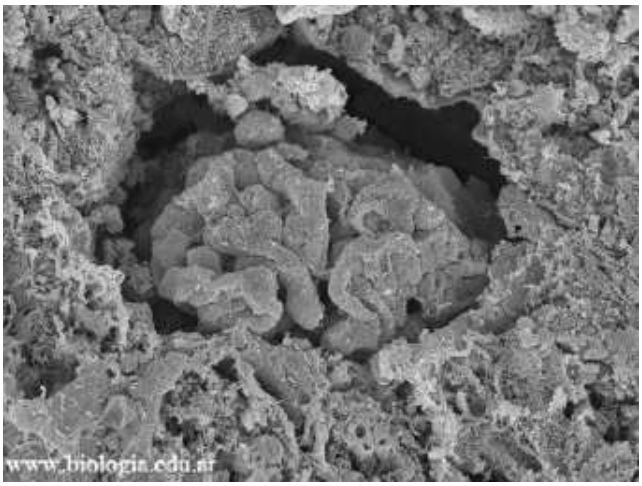
Figura 1.2: Ejemplos de micrografías obtenidas con un Microscopio Electrónico de Transmisión. Imágenes obtenidas de [1]

gar a generar 200.000 aumentos. Son útiles para estudiar superficies de muestras, ya que produce imágenes tridimensionales realistas.

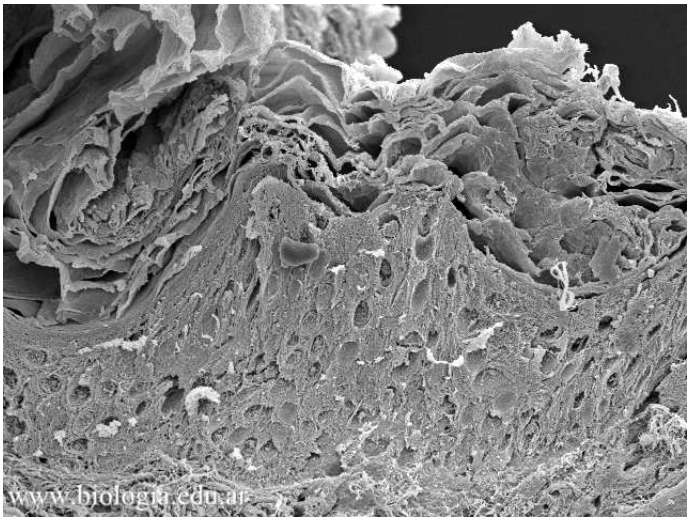
A parte de estos microscopios de los que hemos hablado, cabe destacar que existen otros intentos que básicamente consisten en combinaciones de los dos anteriores. Estos son:

- Microscopio Electrónico de Barrido y Transmisión, el cual intenta combinar las mejores características de los microscopios de transmisión y de barrido y es capaz de mostrar los átomos indivuduales de un objeto.

- Microanalizador de Sonda de Electrones, el cual cuenta con un analizador de espectro de rayos X, por lo que puede analizar los rayos X que emite la muestra al ser bombardeada con electrones. Además, puesto que a partir de los rayos X que emite una determinada muestra al ser bombardeada con electrones se puede obtener información acerca de la composición material, estos microscopios facilitan más información acerca de la muestra que los demás.



(a) Glomérulo renal humano 1200X



(b) Piel humana en corte, 600X

Figura 1.3: Ejemplos de micrografías obtenidas con un Microscopio Electrónico de Barrido. Imágenes obtenidas de [1]

Para este proyecto, el instrumento utilizado para obtener dichas muestras es un microscopio electrónico de transmisión (TEM). En la Figura 1.1 podemos ver una imagen de un TEM.

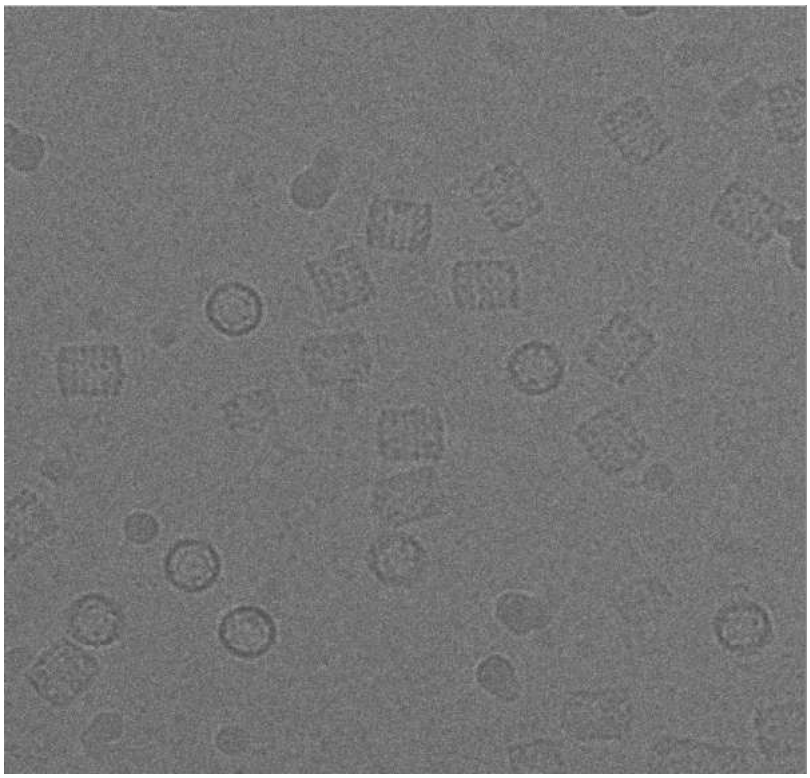
1.2. Tomografía tridimensional

La tomografía es una técnica matemática que trata de derivar los valores de una función real definida en un espacion n-dimensional (por ejemplo, la estructura tridimensional de una partícula) a partir de sus proyecciones en espacios de dimensión más reducida (por ejemplo, a partir de imágenes en 2D de esas partículas).

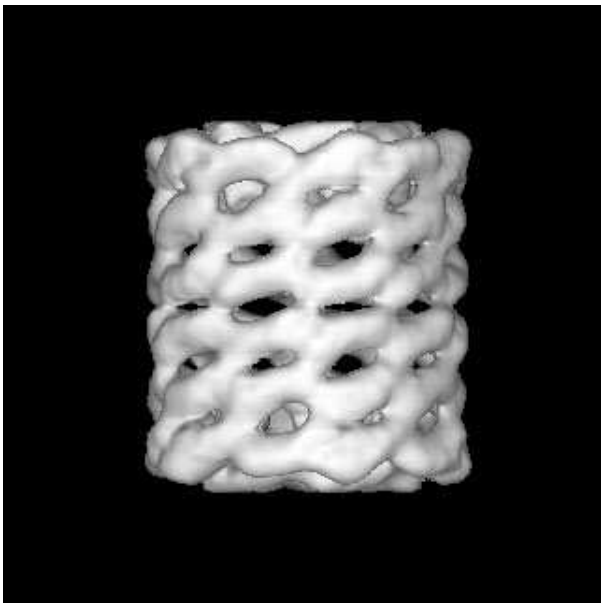
En el proceso de la Tomografía tridimensional, se utilizan muchas proyecciones bidimensionales como la de la Figura 1.4(a) (del orden de 100.000) para, mediante complejos modelos matemáticos, poder obtener un modelo tridimensional con la suficiente resolución como para poder estudiar la estructura terciaria (estructura tridimensional) de la partícula en cuestión. Estos complejos algoritmos, básicamente utilizan ciertos parámetros establecidos al tomar la micrografía, como la dirección de la proyección, la fase, etc. para así establecer el modelo tridimensional. El algoritmo más utilizado para esta reconstrucción es el *Weighted Back Projection* (WBP), aunque métodos basados en el espacio real como *Algebraic Reconstruction Techniques* (ART) han demostrado ventajas sobre WBP dependiendo de los casos.

Este proceso de reconstrucción consta de varios pasos y uno de ellos consiste en marcar la posición exacta de las partículas existentes para cada micrografía con el objetivo de aislar las partes de la micrografía que realmente son útiles para la reconstrucción.

Éste último paso mencionado requiere que alguna persona física recorra todas las micrografías y marque manualmente la posición de todas las partículas, con el considerable tiempo que esto conlleva. Es por eso que un mecanismo que agilice este proceso y lo haga tan automático como sea posible es un objetivo importante en el campo de la reconstrucción tridimensional de partículas, ya que permitiría a científicos utilizar el tiempo de marcado para otras investigaciones. Existen numerosos artículos y documentos que estudian este tema en profundidad con diversos resultados como [14], [11], [31], [21], [30], [22], [24], [23] entre otros.



(a) Micrografía de Ejemplo Obtenida con un TEM



(b) Ejemplo de una Reconstrucción Tomográfica

Figura 1.4: Ejemplo Simbólico de una Reconstrucción Tridimensional de una proteína Keyhole Limpet Hemocyanin.

1.3. Herramientas utilizadas

Para este proyecto, se ha utilizado un conjunto de programas de procesamiento de imágenes microscópicas desarrollado por el CSIC, conocido como XMIPP [27].

Este conjunto de programas de código abierto está orientado a la reconstrucción tridimensional de macromoléculas celulares, desde la adquisición de imágenes, hasta la reconstrucción tridimensional propiamente dicha. Para este proceso de reconstrucción, XMIPP cuenta con numerosos algoritmos de clasificación de imágenes, estimación de CTF(*Contrast Transfer Function*) y reconstrucción 3D entre otros. Además, XMIPP puede ser instalado en cualquier máquina con un compilador C++ GNU y librerías gráficas Qt. Esto significa que XMIPP se puede instalar en prácticamente cualquier máquina UNIX e incluso en Windows a través de Cygwin.

Para nuestro proyecto utilizaremos un programa para marcar partículas disponible en XMIPP llamado `xmipp_micrograph_mark` (Figura 1.5), cuya entrada es una micrografía y cuya salida es conjunto de coordenadas. Lo que se propondrá en este proyecto es un mecanismo para automatizar en la medida de lo posible este proceso, extendiendo y modificando dicho programa para que permita agilizar el proceso.

En el momento de empezar el desarrollo, el programa `xmipp_micrograph_mark` disponía de un mínimo desarrollo orientado a la detección automática de partículas. Algunos de estos elementos se han aprovechado para comenzar este proyecto como ya se explicará más detalladamente en el Capítulo 3.

Para comprobar los resultados obtenidos, disponemos de 80 micrografías de una proteína llamada *Keyhole Limpet Hemocyanin*, más conocida como KLH. Esta proteína se suele encontrar en moluscos y su estudio se centra principalmente en posibles remedios contra el cáncer. Las muestras utilizadas son proporcionadas por el grupo AMI del *The Scripps Research Institute, CA, USA* y están disponibles en la web `http://ami.scripps.edu/experiment/index.php`.

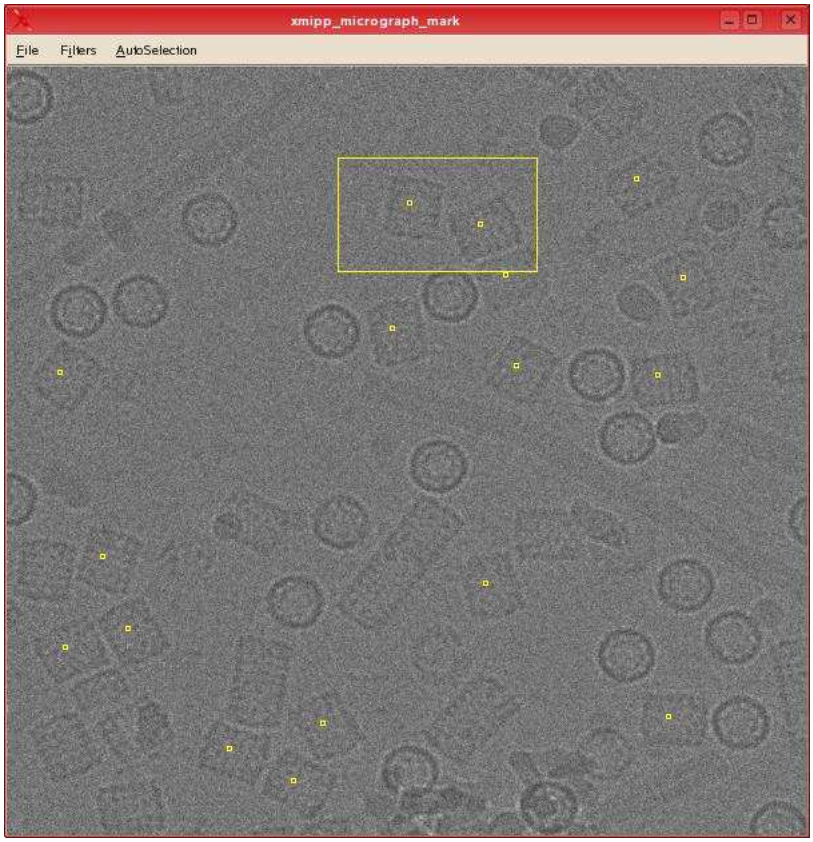


Figura 1.5: Ejemplo de una micrografía vista con XMIPP

Capítulo 2

Estado del arte

La detección automática de partículas no es un campo nuevo dentro de la reconstrucción tridimensional de partículas. Este problema ha sido estudiado extensamente por varios grupos de investigación anteriormente, ya que es evidente que es uno de los principales cuellos de botella existentes en este campo. Con el progreso de las técnicas de procesamiento de imágenes y su aplicación al análisis de partículas, la resolución alcanzada por esta técnica va en aumento, pero a su vez también aumenta el número de micrografías que hacen falta para hacer una reconstrucción a tan alta definición. Es por esto, que la detección automática de partículas empieza a ser un factor clave en dichas reconstrucciones.

En este capítulo repasaremos el estado actual de esta disciplina. Primero veremos un aspecto general, haciendo un resumen general sobre las diferentes técnicas utilizadas y posteriormente entraremos en detalle en algunos de los trabajos realizados.

2.1. Principales técnicas existentes en el campo

A continuación, pasamos a resumir algunas de las técnicas más utilizadas para abordar el problema de la detección.

2.1.1. Comparación de plantillas

Esta técnica conocida como *Template Matching* consiste en ir comparando muestras con una plantilla base considerada como partícula. Hay diferentes métodos de comparación y diferentes criterios de selección. El método de comparación más utilizado es el

de la Correlación Cruzada (*Cross-Correlation*), mientras que el método utilizado para conseguir la plantilla base normalmente consiste en coger una partícula y calcular diversas proyecciones para muchas rotaciones de esa misma imagen.

En general, la función utilizada es de la forma (2.1), donde $f(x, y)$ es la imagen y $g(x, y)$ es la plantilla de referencia.

$$c(x', y') = \sum_x \sum_y f(x, y) g(x + x', y + y') \tag{2.1}$$

Prácticamente todas las técnicas de comparación de plantillas utilizan la correlación cruzada para comparar ya sea como en (2.1) o con variantes de esta como la Correlación Cruzada Modificada, que añade algunas funciones y suposiciones adicionales.

La ventaja de utilizar la correlación cruzada es que podemos pasar la operación anteriormente descrita al dominio de Fourier (2.2), disminuyendo la cantidad de procesamiento necesaria.

$$c(x', y') = F^{-1}\{F\{f(x, y)\}F\{g(x, y)\}^*\} \tag{2.2}$$

Algunos ejemplos de esta técnica se describen en [22], [23].

2.1.2. Detección de ejes

Otra técnica utilizada para detectar partículas es la detección de ejes. Ésta coniste en aplicar unos filtros a la micrografía que resalta los contornos. Esto se podría conseguir aplicando un filtro paso alto, ya que los contornos son partes de la micrografía donde la frecuencia es más alta.

Una vez obtenidos los contornos, se pasa a una clasificación por similitud con unos determinados parámetros como pueden ser tamaño, localización, etc.

La principal ventaja de este método es que, gracias al filtrado paso alto, es insensible a variantes en la luminosidad de la micrografía, lo cual es muy habitual en las muestras obtenidas con un microscopio electrónico. Sin embargo, debido a la baja relación señal a ruido de este tipo de imágenes, muchas veces resulta muy complicado obtener contornos deseados, además de confundir partículas con posibles contaminantes u orientaciones indeseadas.

Ver [31] y [26].

2.1.3. Características de la imagen

Algunos métodos se centran en determinadas características de la imagen, tales como intensidad de píxeles, histogramas de anillos radiales, etc. Estos métodos son bastante eficientes detectando las partículas positivas, pero también tienen una tasa de falsos-positivos bastante alta, ya que cuesta bastante discriminar las impurezas y las orientaciones no deseadas.

Para más información en este tipo de métodos, consultar [19], [20]. Además, en el Capítulo 3 se entra más en detalle acerca de cómo funcionan este tipo de modelos.

2.1.4. Redes neuronales

Las Redes Neuronales son un tipo de sistema que consiste en unidades de decisión lógicas (neuronas) interconectadas entre sí. Se llaman así por su similitud estructural con las neuronas biológicas. Cada neurona tiene una posible salida (por ejemplo 1 ó 0) y una función de transferencia (relación entre la entrada y la salida) determinada dependiendo del modelo estudiado y la interconexión entre todas ellas constituye el resultado global. Pueden tener tantos niveles como se quiera. En (2.3) podemos ver la función de transferencia genérica de una Red Neuronal básica (llamada *perceptron*), donde n es el número de entradas de la red. En la Sección 3.3 se explican más detalladamente las redes neuronales.

$$y = f\{\sum_{i=1}^{n}(w_i xi + w_0)\} \tag{2.3}$$

Este tipo de sistemas normalmente tienen un coste de cómputo muy alto ya que requieren muchas operaciones para obtener una salida a partir de una entrada, pero suelen obtener resultados bastante óptimos en reconicimiento de patrones (*Pattern Recognition*), debido a su capacidad para modelar correctamente sistemas muy complejos.

No se suele utilizar este tipo de sistemas en el reconocimiento de partículas debido a su coste computacional, pero cabe destacar algunos trabajos relacionados, especialmente [14] y [19].

2.2. Trabajos similares más destacados

En esta sección resumimos algunos trabajos destacados en el campo de la detección automática de partículas. Existen numerosos trabajos estrechamente relacionados que

no se comentan en esta sección. Se puede ver un resumen más amplio y exhaustivo en [33] y en [18].

2.2.1. Detecting particles in cryo-EM micrographs using learned features [14]

Resumen

En este trabajo, se describe un modelo basado en el algoritmo de aprendizaje *Adaboost*, el cual ya había sido catalogado como un buen clasificador en el dominio de **reconocimiento de caras**.

Éste es un algoritmo de aprendizaje supervisado (*supervised learning*). Este tipo de algoritmos se caracterizan por tener una fase de entrenamiento a partir de un conjunto de muestras aisladas para dicha fase. A partir de estas muestras, se obtienen una serie de *features*, que posteriormente nos servirán para comparar en la fase de clasificación, y determinar así si lo que estamos escaneando es partícula o no partícula.

Esta primera fase de entrenamiento, se crea un clasificador de dos categorías (partícula/no partícula). Este tipo de clasificadores se llaman Clasificadores Discriminativos. Para ello, obtiene dos conjuntos de muestras (subimágenes de 50×50 píxeles), uno para trozos de micrografía con partícula y otro para trozos sin partícula. Posteriormente, en el conjunto de muestras positivas se coge cada muestra y se rota para que la partícula tenga una orientación determinada (por ejemplo, se alinean las partículas con el borde de la ventana). Con esto, se consigue discriminar más fácilmente orientaciones no deseadas de partículas. Por otra parte, las muestras que contienen las no-partículas debería tener una serie de elementos de la micrografía que pudieran ser confusos para el clasificador, como por ejemplo orientaciones no deseadas, filamentos, etc. de forma que luego en la fase de clasificación le sea más fácil determinar si lo que está escaneando es una impureza o es una partícula.

Una vez obtenidas estas subimágenes, el modelo se compone de varios clasificadores en cascada. Cada uno de estos clasificadores son llamados Clasificadores débiles, para entre todos ellos formar un Clasificador fuerte. En la ecuación (2.4) podemos ver la salida de este Clasificador fuerte expresada de forma matemática, donde α_k se corresponde con el peso de cada clasificador para K clasificadores, mientras que τ es un parámetro de umbral que se ajusta para determinar la relación entre falsos-positivos (no-partículas que el clasificador marca erróneamente) y los falsos negativos (partículas que el clasifi-

cador no marca). $f_k(t)$ es la función de salida de cada Clasificador débil, que puede ser 1 ó -1, según si considera que hay partícula o no.

$$F(t) = \begin{cases} 1 & , \sum_{k=1}^{K} \alpha_k f_k(t) \geq \tau \sum_{k=1}^{K} \alpha_k \\ -1 & \text{otherwise} \end{cases} \tag{2.4}$$

Las características consideradas para este trabajo fueron las imágenes integrales, las cuales se obtienen mediante la integral en dos dimensiones de la imagen original, dada por (2.5). La ventaja de elegir esta característica es que su cálculo es muy rápido.

$$I_{int} = \sum_{x'<x, y'<y} I(x', y') \tag{2.5}$$

Una vez obtenidos estos clasificadores intermedios y estas características, se pasa a un proceso de ponderación, que consiste en:

- por una parte, se define la salida de cada clasificador débil, mediante una técnica llamada *Discriminant Analysis*. Puesto que en este caso el clasificador tiene 2 categorías, habrá que definir 2 funciones discriminativas, una para el conjunto de muestras de partículas y otra para las no-partículas.

  Estas funciones tienen la forma:

  $$P_p^f(t) = -\frac{1}{2} log(\sigma_p^{f2}) - \frac{(\upsilon_f(t) - \mu_p)^2 + log(\lambda)}{2\sigma_p^{f2}} \tag{2.6}$$

  , donde σ_p^f y μ_p^f son la varianza y la media ponderadas de la característica f sobre el conjunto de partículas I_p y $\lambda = (\sum_i w_p(i)/(\sum_i w_p(i) + \sum_j w_{np}(j)))$ es un parámetro de ponderación. Para la función discriminativa sobre el conjunto de no-partículas I_{np}, la fórmula es análoga, excepto que se sustituye λ por $\lambda - 1$.

  Por último, la función del clasificador "débil"queda:

  $$f(t) = \begin{cases} 1 & , P_p^f(t) \geq P_{np}^f(t) \\ -1 & , P_p^f(t) < P_{np}^f(t) \end{cases} \tag{2.7}$$

- por otra parte, discriminar las características menos interesantes. Esto se hace para reducir el gran número de características obtenido previamente, ya que a la hora de clasificar, probablemente habrá un gran número de características que no sean importantes por no aportar la suficiente información. Esta discriminación

se hace mediante el algoritmo *Adaboost*. Este algoritmo discrimina características según el error obtenido al clasificar únicamente con esa característica sobre un conjunto de entrenamiento, discriminando así las que más error provoquen.

El objetivo de tener varios clasificadores en cascada es que cada clasificador vaya rechazando las no-partículas según su dificultad.

Resultados

Los resultados obtenidos mediante este algoritmo son bastante interesantes, ya que se consigue una tasa de *True-Positives* rondando el 90 %, mientras que la tasa de *False-Positives* está alrededor del 20 %. Esta relación *True-Positives*/*False-Positives* es la que realmente determina la calidad del modelo, y en este caso es bastante alta.

Comentarios

Cabe resaltar de este trabajo la cascada de clasificadores. No tanto la cascada en sí, sino la distribución de pesos entre los clasificadores según la varianza de cada uno de ellos. Esto es importante, ya que a mayor varianza, mayor información aportará ese clasificador, es decir, más diferencia habrá entre partícula/no partícula, y por tanto más puede influir en la decisión final.

2.2.2. Automatic particle pickup method using a neural network has high accuracy by applying an initial weight derived from eigenimages: a new reference free method for single-particle analysis [20]

Resumen

En este documento se expone un nuevo método para la detección, el cual se vale de las Redes Neuronales. En la Sección 3.3 se entra un poco más en detalle acerca de este tipo de modelo utilizado como clasificador.

El modelo empleado se basa en una Red Neuronal de 3 capas con distribución piramidal, donde la capa de entrada tiene 1600 nodos de entrada, la capa oculta tiene 81 y la capa de salida únicamente tiene 1 nodo, el cual determina si hay partícula o no.

Este modelo precisa de una fase de aprendizaje, en la cual se marcan 200 partículas de forma manual. A partir de estas partículas, se inicializa la red neuronal con unos

pesos aleatorios, para mediante un algoritmo de propagación regresiva, ir actualizando los pesos de la red. Este algoritmo consiste en obtener la función de error en la salida, y en función de este error, propagarse por la red regresivamente (desde la salida a la entrada) e ir actualizando los pesos con alguna relación matemática.

Además, cabe destacar que en el proceso de aprendizaje, a cada una de las 200 partículas se le aplica una rotación de 360°a intervalos de X grados, para así obtener más muestras de aprendizaje con varias orientaciones de las partículas, lo cual es importante, puesto que las partículas rara vez tendrán la misma orientación en una misma micrografía. Se ha demostrado que existe una gran relación entre el coeficiente de rotación X y la precisión de este modelo, ya que cuanto menor sea X, más muestras de aprendizaje habrá y mejor conocerá el modelo qué características tiene una partícula.

Posteriormente en este trabajo, se propone una mejora para este modelo, la cual consiste en inicializar la red neuronal con los pesos obtenidos a partir de las autoimágenes obtenidas mediante PCA (*Principal Component Anañysis*). Esta es una técnica que permite, gracias a una transformada espacial, poder disminuir las dimensiones de un determinado conjunto de datos, lo que significa que a partir de la imagen de la partícula, puede aislar los datos que realmente resultan interesantes con poca pérdida de información. Esto lo consigue mediante funciones como la matriz de covarianza, descomposición de autovalores entre otros.

Se demuestra que al inicializar de esta forma la red neuronal, el tiempo necesario para todo el proceso se reduce en aproximadamente el 20 %, aumentando ligeramente la precisión de la red neuronal.

Resultados

La detección de partículas con este modelo llega a alcanzar una tasa de aciertos de aproximadamente 95 %, resultado que se encuentra entre los mejores en este aspecto. Sin embargo, no menciona nada acerca de los falsos positivos, que son la mayor fuente de problemas en este campo.

Comentarios

Este método parece ser mejor que el método tradicional basado en la correlación-cruzada, sin embargo, aún así sigue existiendo un paso posterior a la detección para eliminar los falsos-positivos que hayan podido aparecer.

Se echan de menos algunos comentarios sobre cómo funciona exactamente el modelo

de detección, ya que no entra en detalle sobre cómo clasifica realmente la red neuronal. Existen varios tipos de redes neuronales muy utilizadas, las cuales tienen una función de transferencia conocida para cada nodo. En el documento descrito se detalla muy poco qué función ha sido utilizada en los nodos.

Por otra parte, en los resultado obtenidos habla de la tasa de aciertos, pero no menciona la de Falsos-positivos, los cuales suelen ser el mayor problema encontrado en el campo de la detección automática, ya que resulta realmente difícil aislar estos elementos.

Por último, este proceso es realmente pesado computacionalmente hablando, lo cual se demuestra en el tiempo que lleva realizar toda la fase de aprendizaje (alrededor de 1 hora).

2.2.3. FindEM–a fast, efficient program for automatic selection of particles from electron micrographs [23]

Resumen

Para esta investigación, se ha utilizado un programa llamado FindEM, el cual se basa en la comparación con plantillas utilizando la correlación en el espacio real. Este tipo de algoritmos suelen ser bastante lentos, pero si se pasa al campo de Fourier, su rendimiento aumenta considerablemente, ya que las operaciones en Fourier siempre suelen ser menos costosas. Este algoritmo en el campo de Fourier es comúnmente conocido como *Fast Local Correlation Function* (FLCF). Los resultados expuestos son a partir de muestras de la proteína *KLH*.

Mediante esta comparación de plantillas, se crean unos mapas de correlaciones, con unos picos en los puntos donde haya partículas, para posteriormente pasarlos por unos filtros que se encargan de rechazar elementos con menor correlación (por debajo de un umbral), así como de rechazar partículas superpuestas e impurezas del estilo.

En este documento se describen dos modelos diferentes, aunque estrechamente relacionados entre sí. El primero consiste en generar 2 plantillas diferentes, una para detectar las orientaciones laterales y otra para detectar las orientaciones verticales de la proteina. Esto requiere conocer *a priori* determinados parámetros relacionados con la estructura tridimensional, pero por otra parte, su capacidad de detección mejora, ya que conoce mejor lo que está buscando. El segundo método consiste en tener una única plantilla para detectar todas las orientaciones de la partícula. Este método es más débil,

ya que es menos específico.

Método 1

1. Se crean las plantillas a partir de una micrografía base. Se marcan 20 partículas de esta micrografía, se calcula la correlación de la orientación y la traslación, para posteriormente hacer un promedio y obtener unas plantillas con una alta calidad señal/ruido.

2. Postiormente se correla la micrografía con las plantillas y se extraen los picos, creando dos listas, una para cada caso.

3. Para intentar evitar en la medida de lo posible los falsos-positivos, primero se desechan los picos que estén por debajo de un cierto umbral definido para cada lista.

4. Un segundo filtrado consiste en refinar los picos obtenidos, desechando los que estén por debajo de un valor definido por el usuario.

5. Se comparan las listas de picos obtenidas, buscando ocurrencias comunes, y desechando la que tenga un valor más bajo.

6. Por último, se desechan las partículas que estén muy cerca unas de otras, mediante una distancia definida por el usuario.

Método 2

1. La plantilla se construye exactamente igual que en el Método 1, con la diferencia de que únicamente se construye una.

2. Se correla la plantilla con la micrografía que estemos analizando y se obtienen los picos de la correlación.

3. El umbral a partir del cual hacemos el primer filtrado ha de ser más bajo, ya que tenemos tipos diferentes de partículas y por tanto las correlaciones disminuirán.

4. Posteriormente, los picos obtenidos son procesados mediante una técnica de clasificación llamada *Multivariate Statistical Analysis*, la cual separa directamente las clases existentes en la muestra que recibe.

Resultados

Los resultados obtenidos para este modelo son interesantes, alcanzando un tasa de acierto de 97 % y una tasa de falsos-positivos del 20 % para el Método 1. Para el Método en cambio, la tasa de aciertos es del 80 % mientras que no se comenta cuál es la de los falsos-positivos, pero por la constitución del modelo, lo lógico es que fuera mayor que en el Método 1.

Comentarios

Modelo bastante interesante por su simplicidad, pero parece demasiado estático. Requiere saber demasiada información acerca del tipo de partículas que estás buscando. Además, los resultados obtenidos son suponiendo que estás buscando cualquier orientación de la partícula (tanto lateral como vertical), suposición que no suele ser la adoptada. Con esta suposición suaviza drásticamente el efecto de los falsos-positivos.

En el documento se proponen mejoras interesantes, sobre todo para mejorar la velocidad, ya que para detectar esas alrededor de 1000 partículas en las 82 micrografías se tarda alrededor de 56 minutos.

2.2.4. An approach to automatic particle picking from electron micrographs based on reduced representation templates [30]

Resumen

Presentación de un nuevo algoritmo, el cual consta de 5 partes bien definidas:

1. **Contrucción de las plantillas reducidas:** Se construyen plantillas reducidas a partir de otros modelos o bien de los datos directamente. Estas plantillas serán utilizadas posteriormente para la comparación de patrones en la fase de clasificación. En principio es mejor obtener estas plantillas directamente de los datos en vez de otros modelos, ya que así evitamos posibles limitaciones del modelo, así como posibles diferencias entre modelos de contraste, intensidad, etc.

   Para crear estas plantillas, se marcaron 75 partículas *KLH* manualmente de 7 micrografías. Una vez marcadas estas partículas, se alinean y se hace un promedio de todas ellas. Posteriormente, para fabricar la plantilla reducida, se cogen una

serie de píxeles representativos para agilizar el proceso. Para este experimento, se hicieron dos plantillas diferentes, una con 40 puntos y otra con 74. El resultado no tuvo variaciones importantes entre una plantilla y otra, corroborando la robustez del modelo. La fase de reconocimiento es bastante rápida gracias a esta reducción importante de datos.

2. **Selección de candidatos:** A la hora de comparar con la plantilla en la fase de detección, se utilizó una función bastante simple del tipo:

$$S = \sum I_{bright} - \sum I_{dark} \tag{2.8}$$

donde I_{bright} y I_{dark} son las intensidades en los puntos claros y oscuros respectivamente.

El resultado de esta función será alto cuando los puntos asignados a I_{bright} tengan intensidades altas y simultáneamente los puntos asignados a I_{dark} tengan intensidades bajas. En principio, esto debería ocurrir cuando lo que estemos analizando sea una partícula.

3. **Filtrado por resultado en la comparación:** El siguiente paso consite en seleccionar los mejores resultados para la función de referencia (2.8). Este proceso además se vale del hecho de que los picos de partículas positivas serán más altos que los de las orientaciones indeseadas o las impurezas. Este proceso tiene dos fases. En la primera se define un primer umbral para desechar los peores resultados (impurezas y orientaciones indeseadas), mientras que la segunda fase se define otro umbral para filtrar mejor estas selecciones.

Este proceso tiene una serie de parámetros ajustables, los cuales deberán ser reajustados para cada tipo de modelo que estemos estudiando.

4. **Filtrado por distancia:** Posteriormente se hace un filtrado por distancia física de los picos, para así desechar partículas que estén demasiado próximas unas de otras. Esta distancia mínima se define a partir de la distancia mínima de las partículas de la plantilla reducida.

5. **Filtrado de datos anómalos:** Por último, se hace un filtrado de *outliers* ó datos anómalos. Estos salientes son elementos dentro del conjunto elegido que son un tanto diferentes a la media. Para obtener estas anomalías, se alinean todas las

partículas y se calcula la media de todas ellas y posteriormente la correlación cruzada entre la media y cada una de las partículas candidatas. Cualquiera de estas candidatas que tenga una varianza en la correlación de más de 3 órdenes de magnitud es desechada. Este paso es interesante para quitar posibles orientaciones indeseadas que no hayan podido ser excluidas previamente.

Resultados

Los resultados obtenidos con este método tienen una parte muy buena y otra no tan buena. La tasa de falsos-positivos es increiblemente buena, llegando a obtener aproximadamente un 6 %. Sin embargo, este gran resultado en los falsos-positivos tiene su contrapartida en cuanto a la tasa de aciertos, que se sitúa en un 59 % en el peor de los casos y en un 73 % en el mejor de ellos.

Comentarios

Método muy interesante por su simplicidad, sin embargo, no obtiene muy buenos resultados en cuanto a la tasa de aciertos, si bien cabe destacar que la tasa de falsos-positivos es de las mejores vistas hasta la fecha.

Un punto negativo a destacar es el poco dinamismo de este algoritmo, debido al método de construcción de la plantilla reducida con la cual se hace la comparación en la fase de clasificación, que se basa en promedio de intensidades, el cual es muy sensible a cambios muy habituales como pueden ser los cambios de contraste y de intensidad.

Capítulo 3

Modelo de clasificación

En esta sección, se explica detalladamente el modelo utilizado para abordar la detección automática. Se hará una breve comparativa acerca de los diferentes clasificadores más utilizados en diferentes problemas de clasificación y reconocimiento de patrones. Más adelante, una vez hayamos elegido nuestro clasificador, entraremos más en detalle acerca de qué parámetros significativos de las partículas vistas en las micrografías, de manera que consigamos caracterizarlas lo más posible y que así resulte más fácil la detección.

Antes de todo, explicaremos en qué estado se encontraba el programa que vamos a modificar, el cual ya disponía de su propio modelo inacabado de clasificación. Veremos qué se puede aprovechar de éste, qué cosas se pueden cambiar y qué otras cosas deben ser desechadas.

3.1. Modelo previo del programa

En el momento de empezar este proyecto, XMIPP ya tenía una versión previa del programa `xmipp_micrograph_mark`. Dicha versión principalmente se utilizaba para marcar de forma manual las partículas necesarias para el proceso de reconstrucción tridimensional. A pesar de que el uso principal fuera éste, el programa disponía de un desarrollo orientado al marcado automático de partículas, aunque sin finalizar.

El modelo anterior se basaba en un modelo orientado a las características (*features*). Disponía de una fase de entrenamiento, en la cual se marcaba un número determinado de partículas, para a partir de ellas extraer algunas características importantes.

De estas características extraidas, se calculaba la media y la varianza, almacenándo-

las para pasar posteriormente a la fase de clasificación. En esta fase, se iba recorriendo la micrografía en busca de partículas. Esta búsqueda se abordaba calculando para cada subimagen analizada las mismas características que calculábamos en la primera fase de aprendizaje y medir la distancia de *Mahalanobis* con respecto a la media del modelo. En la Ecuación (3.1) podemos ver la ecuación correspondiente a dicha distancia, donde Σ es la matriz de covarianza.

$$d_m(\mathbf{x}, \overline{\mathbf{x}}) = \sqrt{(\mathbf{x} - \overline{\mathbf{x}})^t \Sigma^{-1} (\mathbf{x} - \overline{\mathbf{x}})} \tag{3.1}$$

Si esta distancia era menor que un determinado umbral, entonces se consideraba como partícula candidata. Sin embargo, en el modelo antiguo existía un problema relacionado con la independencia lineal entre características, que hacían que el determinante de la matriz de covarianza fuera cero, con lo que la distancia obtenida no era estable.

Una vez obtenidas todas las partículas candidatas, se aplicaban una serie de filtros sobre éstas. El primer filtro consistía en excluir las partículas candidatas que apuntaran a la misma partícula, es decir, varias candidatas estaban tan cerca unas de otras, que realmente apuntaban a la misma partícula. El segundo filtro en cambio, pretendía excluir las partículas que estuvieran muy cerca unas de otras, excluyendo ambas en caso de que la condición se cumpliera. Para estos filtros, se utilizaban una serie de parámetros definidos por el usuario en la fase de aprendizaje, como por ejemplo el radio de la partícula.

Ya por último, una vez obtenidas las partículas filtradas, se redefinían los centros de todas ellas buscando la mejor distancia con repecto al modelo de referencia.

De todas estas partículas, las que fueran posteriormente rechazadas por el usuario, entraban a formar parte de un modelo de error, al cual se le aplicaban los mismos pasos de caracterización que para las partículas en la fase de aprendizaje.

En subsiguientes análisis, este modelo entraría en juego, calculando la distancia de la subimagen analizada, tanto con respecto al modelo de partículas positivas como para el de errores. El modelo que estuviera más cerca sería al que perteneciera.

Para la nueva versión de este programa, algunas partes de esta versión anterior han sido aprovechados, otros modificados y otros desechados. En el Capítulo 5 se entra más en detalle acerca de cómo se ha implementado la solución, haciendo incapié en qué elementos de este modelo han sido aprovechados.

3.2. Requisitos del nuevo modelo

El clasificador es la parte más importante del modelo, ya que por mucho que aprendamos 10.000 parámetros del modelo, si luego no utilizamos un clasificador con la suficiente robustez no hará bien la clasificación, de la misma forma que si no elegimos un clasificador eficiente, puede que tarde demasiado en llevar a cabo tal clasificación.

Ahora ajustaremos algunos parámetros previos de nuestro modelo. Una vez ya hemos repasado el estado del arte (Capítulo 2) en la materia, podemos establecer algunas imposiciones según los diferentes resultados obtenidos por otros grupos de investigación.

1. En primer lugar, parece que la técnica más utilizada en este campo, conocida como *Comparación de Plantillas* es demasiado estricta, puesto que requiere conocer aspectos morfológicos de las muestras y queremos que el modelo sea lo suficientemente flexible para que no haga falta conocer *a priori* estos datos. Por tanto, buscaremos una técnica alternativa, basada en Características (*features*).

2. En la mayoría de trabajos estudiados, la fase de aprendizaje creaba un conjunto de datos de aprendizaje, ya fueran plantillas o características, para posteriormente ser comparados en la fase de clasificación. Para llevar a cabo esta clasificación, la subimagen analizada ha de ser rotada muchas veces para asegurarse de que estudiamos la orientación correcta. Este último proceso podría ser eliminado si basamos nuestros datos de aprendizaje en características invariables a rotaciones. De esta manera, las características obtenidas para una determinada orientación de partícula nos servirían para otras posibles orientaciones de ese mismo tipo de partícula.

3. En las características también sería interesante introducir algunos elementos que permitan al clasificador conocer algo acerca de la geometría de las partículas, para intentar evitar en la medida de lo posible las orientaciones indeseadas, por ejemplo, orientaciones verticales de la proteína *KLH* (geometría circular) cuando intentamos encontrar las orientaciones laterales de dicha proteína (geometría rectangular). En la Sección 3.4 se entra más en detalle acerca de las características empleadas en el nuevo modelo.

4. Parece bastante interesante disponer de un modelo de error con el que comparar las muestras, de manera que el sistema pueda decidir no sólo si estamos observando una partícula o simplemente fondo, sino que también pueda ir aprendiendo de

los errores de detecciones pasadas y que posteriormente también compare las muestras con los errores pasados.

5. La función de clasificación del modelo anterior basada en la distancia de *Mahalanobis* no parece lo suficientemente robusta, ya que es difícil evitar que haya cierta dependencia lineal entre unas variables y otras. Necesitamos algún modelo que incluya una buena función de clasificación.

6. En [14] se menciona una selección de características. Básicamente, esto consistía en descartar las características menos importantes del modelo, ya que aportan menos información al clasificador.

   Para este modelo haremos algo parecido. En vez de descartar características, las ponderaremos, haciendo que las que más información aporten sean más importantes e influyan más en la decisión del clasificador.

Nuestro clasificador tendrá tres fases bien definidas:

- Una primera fase de entrenamiento en la que calculamos una serie de parámetros (características) de las subimágenes correspondientes a las partículas, al fondo de la micrografía, a los errores, etc. En este paso se puede separar en tantos subconjuntos como se quiera. Posteriormente explicaremos cuáles son los subconjuntos que hemos supuesto para nuestro modelo.

- Una segunda fase de clasificación, en la cual se recorre la micrografía en busca de partículas. Este proceso requiere recorrer la micrografía entera, lo cual llevará un cierto tiempo, por lo que debemos intentar elegir un clasificador ligero en lo que a cálculos se refiere. Para cada subimagen analizada, se utilizará alguna función de calidad del modelo para intentar averiguar si lo que estamos analizando es una partícula o no.

- Por último, tendremos que aprender los datos que hayamos ido calculando para cada micrografía, ya sean nuevas partículas, nuevos errores, muestras de fondo, etc.

3.3. El clasificador

Nuestro enfoque del problema de la detección lo afrontamos como un problema de clasificación. Por tanto, la solución consistirá en elegir un buen clasificador. Existen

principalmente dos tipos de clasificadores:

1. **Clasificadores genéricos:** Estos clasificadores se caracterizan por obtener datos de entrenamiento únicamente de muestras positivas (partículas).

2. **Clasificadores discriminativos:** Los clasificadores de este tipo obtienen características tanto de muestras positivas como negativas (no-partículas). El modelo anteriormente expuesto en la Sección 3.1 pertenecería a este tipo de clasificadores.

Para nuestra solución, elegiremos un **clasificador discriminativo**, puesto que parece bastante interesante comparar la muestra observada con más de un subconjunto de datos, ya que *a priori* debería dar más precisión al modelo. De los trabajos expuestos en la Sección 2.2, el de Mallick [14] es el que mejor explica un modelo de este tipo, en el cual, la fase de entrenamiento se realiza con muestras de no-partículas.

3.3.1. Clasificadores más comunes

Ahora que ya sabemos qué tipo de clasificador queremos, analizaremos algunos de los clasificadores más comunes en diversos problemas de detección. Nos centraremos en los Árboles de Decisión, las Redes Neuronales y las Redes Bayesianas, ya que *a priori* son las técnicas de detección que mejor permiten clasificar una serie de datos en varias clases. En esta sección no se pretende dar una clase teórica acerca de estos clasificadores, pero sí estudiar brevemente cuáles son las principales características de unos y otros para posteriormente poder elegir cuál de ellos se ajusta mejor a nuestro propósito.

Árboles de decisión

Un árbol de decisión es una herramienta de toma de decisiones que se basa en un determinado diagrama o modelo de probabilidades. Su función normalmente es buscar la mejor estrategia para conseguir una determinada meta. También son ampliamente utilizados para inferir las probabilidades condicionales de un determinado modelo. Estos modelos son llamados **modelos predictivos**, es decir, intenta averiguar cuál es el resultado más probable a partir de unas determinadas observaciones previas de ejecuciones del modelo. También se los conoce como Árboles de clasificación (cuando la salida es discreta) o Árboles de regresión (cuando la salida es contínua).

Puesto que nosotros estamos trabajando con datos discretos, nos centraremos un poco más en los árboles de clasificación. Éstos se caracterizan por, dado un conjunto de

posibles clases de unos determinados datos, predecir a qué clase puede pertenecer una determinada variable dependiente. El análisis de los árboles de clasificación es una de las principales técnicas en el campo de la minería de datos (*Data Mining*).

La flexibilidad de este modelo lo hace una opción ampliamente utilizada, lo cual no significa que otros métodos tradicionales como Análisis Discriminativo, Estimación No Lineal, etc. no puedan ser mejores para determinados casos.

La construcción de estos modelos está muy ligada a su representación gráfica. Se dibujan una serie de nodos en una estructura arbolar, donde cada nodo es una variable dependiente. Posteriormente, a partir de unos determinados datos obtenidos previamente (los cuales serían los datos de entrada del nodo raíz), cada nodo evalúa su variable, produciendo un determinado resultado que es recogido por el nodo siguiente, produciendo a su vez un resultado y así sucesivamente.

Para entender mejor este proceso, veamos un ejemplo. En [2] se pueden ver varios ejemplos de utilización de árboles de clasificación. Entre ellos, se expone uno acerca de pacientes con ataques al corazón. Cuando un paciente es admitido en un hospital por un ataque al corazón, se le hacen una docena de tests fisiológicos como medir el pulso, la tensión, etc. Además, se obtiene más información de otras fuentes como puede ser su historial médico y su edad entre otros. A partir de estos datos, lo que se pretende averiguar son las probabilidades que tiene el paciente de sobrevivir más de 30 días, de manera que el hospital pueda aprovechar mejor sus recursos. Para modelar este caso, se construyó un árbol de clasificación muy simple de 3 preguntas. La definición de este modelo podría ser

> Si la presión sistólica mínima del paciente en las primeras 24 horas es mayor de 91 y si la edad del paciente está por encima de los 62.5 años y si el paciente muestra taquicardia, entonces y sólo entonces el modelo predice que el paciente no sobrevivirá 30 días.

A partir de esta definición sería bastante simple hacer un diagrama como el de la Figura 3.1. Posteriormente, se formularían una serie de preguntas jerárquicas para posteriormente obtener una decisión que dependa de lo que hayan respondido los nodos anteriores. Una expresión más práctica, podría ser

> Siendo P la presión sistólica, E la edad del paciente y T la presencia o ausencia de taquicardia y sus respectivos coeficientes $p = -91$, $e = -62{,}5$ y $t = 0$, entonces si $p + P \leq 0$ entonces el paciente es de bajo riesgo, si

$e + E \leq 0$, entonces el paciente es de bajo riesgo y por último, si $t + T \leq 0$ entonces el paciente es de bajo riesgo. Si no se cumple ninguna de las relaciones anteriores entonces el paciente es de alto riesgo.

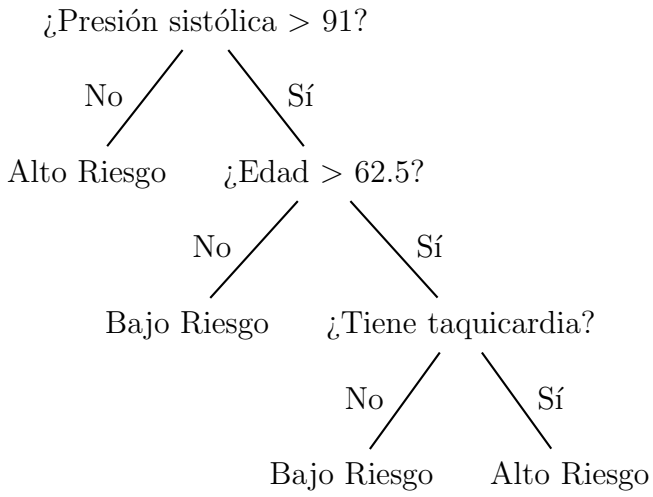


Figura 3.1: Ejemplo de un Diagrama representando un Árbol de Decisión

Una de las ventajas principales de este tipo de modelos es su gran flexibilidad. Por ejemplo, condiderando el caso anterior del paciente con ataque al corazón, la variable E que denota la edad del paciente es contínua, mientras que la variable T que denota la presencia o ausencia de taquicardia es discreta y booleana. Además, en cualquier momento se podría hacer que la edad fuera discreta, sin afectar gravemente al resultado. También es muy flexible en cuanto a introducir nuevas variables, por ejemplo, se podría introducir una nueva variable A, que denote si el paciente ha sufrido ataques anteriormente, los cuales podrían haber debilitado el corazón, que probablemente ayudaría a determinar mejor el riesgo del paciente.

Por otra parte, una gran desventaja de este modelo es que depende mucho de qué variables decidas escoger para predecir. Si estas variables no son escogidas correctamente, el modelo no obtendrá buenos resultados, lo que requiere conocer en detalle los datos que queremos modelar.

Redes neuronales

Tradicionalmente, el término **Redes Neuronales** siempre ha sido utilizado para referirse a una red de neuronas biológicas.

En una red neuronal biológica, existe un gran número de neuronas interconectadas entre sí. Cada neurona está conectada con otro número variable de neuronas. A través de estas interconexiones, las neuronas transmiten señales eléctricas en respuesta a un determinado estímulo. El entramado de estas interconexiones puede llegar a ser extremadamente complejo.

Recientemente, se ha introducido un nuevo término íntimamente relacionado con estas redes biológicas. Las **Redes Neuronales Artificiales** son modelos matemáticos basados en la Redes Neuronales Biológicas, las cuales consisten en un grupo de neuronas artificiales interconectadas entre sí que procesan cierta información con una determinada función de transferencia (relación entre la entrada y la salida). Normalmente, las redes neuronales artificiales son sistemas adaptativos, en el sentido de que cambian su estructura interna dependiendo de los datos de entrada y salida que se producen en la fase de entrenamiento. En términos más prácticos, constituyen un sistema no lineal de modelado de datos estadísticos, que pueden relacionar una entrada con una salida mediante interconexiones complejas. Algunos ejemplos de aplicación de redes neuronales han sido llevados a cabo satisfactoriamente en sistemas de reconocimiento del habla, análisis de imágenes, etc.

En la Figura 3.2 podemos ver un ejemplo de red neuronal artificial, la cual contiene 9 neuronas interconectadas entre sí. Las neuronas que reciben la señal de entrada se considera que están en la **Capa de entrada** y las que obtienen la salida del sistema se consideran pertenecientes a la **Capa de salida**. Todo lo que no esté en alguna de estas dos capas se considera que está en **Capas ocultas**.

En todo sistema modelado por una red neuronal existen dos fases: una primera fase de prueba y una fase de aprendizaje. En la primera es dónde a partir de un conjunto de parámetros representativos del sistema que queremos modelar, se forma la estructura (pesos internos de los nodos) de la red. Por otra parte, en la fase de aprendizaje o de ejecución, se van obteniendo diferentes salidas para diferentes entradas mientras que a la vez la red va aprendiendo de estas salidas y se va redefiniendo a sí misma cambiando los pesos de sus nodos internos para conseguir modelar mejor el sistema. Esto último normalmente se consigue midiendo el error a la salida, para en función de este error cambiar los pesos internos para minimizar dicho error en la siguiente iteración.

Una red neuronal típica se puede caracterizar por las descripciones funcionales de su **Red de Conexión** y de su **Red de Activación**. Cada nodo suministra a la red su valor de salida a_i. Este valor se propaga hacia otros nodos de la red, ponderado por el peso de la conexión x_{ij}, el cual determina el efecto del nodo j sobre el nodo i. La salida

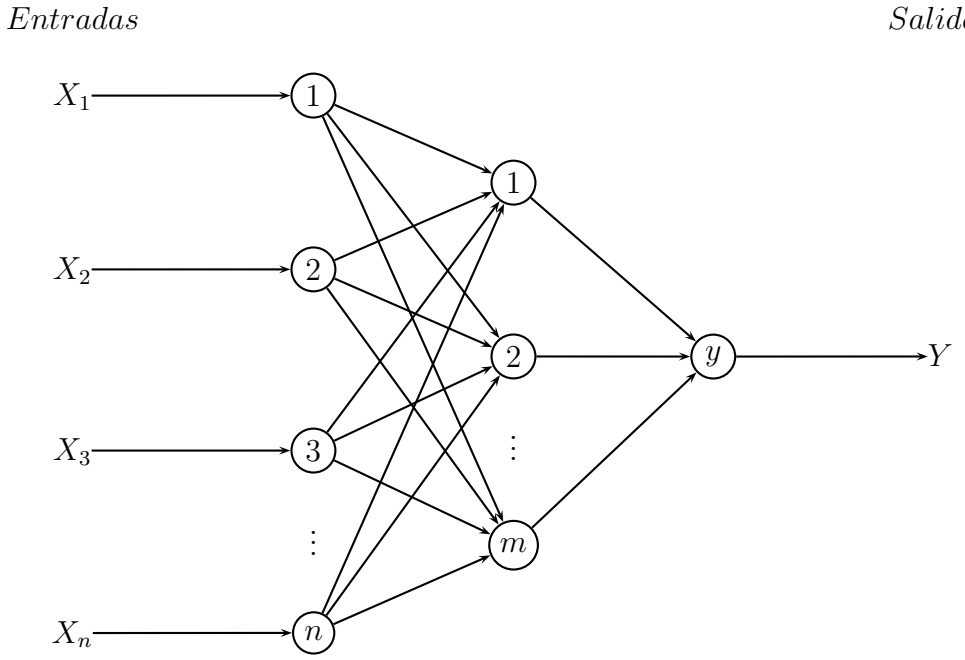


Figura 3.2: Diagrama de Red Neuronal de 3 Capas, con una estructura $m \times n \times 1$

total de la red puede ser expresada en función de los pesos de la red y de la entrada $y = \phi(x, W)$, donde x es el conjunto de entradas a la red y W el conjunto de pesos de los nodos de la red.

La capacidad de clasificación de una red neuronal depende del valor de los pesos de cada nodo, que pueden ser preestablecidos o entrenados en la fase de aprendizaje. Las redes neuronales se pueden clasificar, dependiendo de cómo sean establecidos estos pesos en dos categorías: Modelos Supervisados y Modelos No Supervisados.

En un modelo supervisado, se intenta obtener la funcón f, tal que concuerda con una serie de muestras dadas inicialmente del tipo $(x, y), x \in X, y \in Y$. En otras palabras, intentamos inferir la estructura que concuerde con los datos dados. Para esto hace falta una función que mida cuán bueno es el modelo obtenido. Esta función de calidad suele ser el error medio cuadrático, más conocido como MSE (*Mean Squared Error*), el cual trata de minimizar el error medio de la salida $f(x)$ con respecto a los datos de entrada. Normalmente, en este tipo de modelos se utiliza un algoritmo de propagación regresiva, de manera que a partir del MSE obtenido, se actualicen todos los pesos de la red para obtener un error menor. Este tipo de modelos se utilizan principalmente en problemas de reconocimiento de patrones, **clasificación**, regresión, etc.

Por otra parte, en los modelos no supervisados, partimos de unos datos x y una

función de calidad que ha de ser minimizada. Esta función es variable según el tipo de datos que queramos modelar y algunas suposiciones *a priori*, como pueden ser propiedades del modelo, parámetros y variables observadas. La principal aplicación práctica de estos modelos son problemas de compresión, filtrado, etc.

Un aspecto muy importante de las redes neuronales es el algoritmo de aprendizaje utilizado. Este algoritmo normalmente se basa en alguna función del gradiente del modelo, de manera que lo que se intenta es dirigir el modelo hacia la dirección del gradiente que minimice la función objetivo.

Para nuestro modelo, hemos desechado las Redes Neuronales debido a su gran complejidad computacional. Estos modelos son muy útilies para modelar sistemas muy complejos en los cuales no existen grandes limitaciones de tiempo y capacidad computacional. Éste no es nuestro caso, ya que al ser una aplicación de escritorio y un tiempo de referencia que es el que tardaría un experto en marcar todas las partículas exitentes, necesitamos un modelo más ligero que éste.

Redes bayesianas

Una Red Bayesiana es un modelo probabilístico que relaciona un conjunto de variables aleatorias mediante un diagrama que indica explícitamente influencia causal. Gracias a su motor de actualización de probabilidades, el Teorema de Bayes, las redes bayesianas son una herramienta extremadamente útil en la estimación de probabilidades ante nuevas evidencias.

Formalmente, las Redes Bayesianas son grafos acíclicos dirigidos (*Directed Acyclic Graph*), normalmente conocidos como DAGs, cuyos nodos representan variables y los arcos que los unen codifican dependencias condicionales entre las variables. Existen algoritmos que realizan inferencias del modelo a partir de unos datos obtenidos para averiguar otros.

En la Figura 3.3 podemos ver un ejemplo de Red bayesiana comúnmente utilizada para explicar este tipo de modelos. Dicha figura, modela un sistema para averiguar la razón por la cual ha podido sonar la alarma de nuestra casa (A), si porque han entrado ladrones (L) o porque ha habido un terremoto (T). Para averiguarlo, podemos modificar nuestras hipótesis según si hemos oido en la radio que iba a haber un terremoto (R). Además, ambos podrían provocar que la alarma sonara con más o menos probabilidades (en principio es más probable que la alarma suene porque hayan entrado ladrones que porque haya habido un terremoto). Sobre estas hipótesis también puede influir en

nuestra observación el hecho de que un vecino haya llamado avisando (V). Este tipo de razonamiento es la base de las redes bayesianas, intentar inferir una causa a partir de unos determinados datos u observaciones.

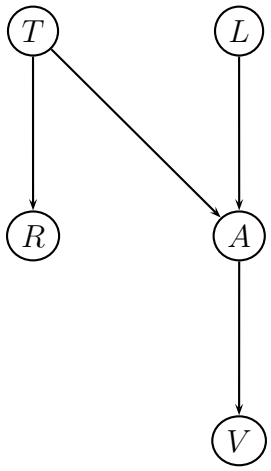


Figura 3.3: Ejemplo de una Red Bayesiana comúnmente utilizada como ejemplo

A partir del diagrama, se pueden establecer dependencias entre variables e inferir su distribución conjunta de probabilidad. Si existe un arco que va desde el nodo A hasta el nodo B, entonces A se considera padre de B. El conjunto de padres de un determinado nodo X_i se representa como $padres(X_i)$. A partir de estos nodos, la distribución conjunta de probabilidad sería de la forma (3.2) donde n es el número de nodos de la red.

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | padres(X_i)) \tag{3.2}$$

Puesto que una red bayesiana modela completamente las variables de un determinado sistema así como la relación entre ellas, también puede ser utilizada para hacer determinadas consultas acerca de una determinada variable del modelo. Por ejemplo, se pueden utilizar para averiguar cómo varían el resto de variables del modelo en el caso de que estemos observando algunas otras. Este proceso de calcular probabilidades *a posteriori* de unas determinadas variables se conoce comúnmente como **inferencia** probabilística. Esta técnica es ampliamente utilizada en determinados problemas en los que se quiere elegir un determinado valor para unas variables con el propósito de minimizar alguna función de calidad del modelo (normalmente algún tipo de función de error). Este proceso de inferencia es justamente lo que intentamos aplicar a nuestro mo-

delo, ya que lo que queremos es inferir qué clase (partícula/no partícula) se ajusta mejor (en otras palabras, cuál genera un error menor) a unos datos dados (características).

Existen numerosas maneras de aplicar esta inferencia, siendo las más destacadas la eliminación de variables, condicionamiento recursivo, etc. En redes bayesianas de un tamaño mínimamente aceptable, este proceso de inferencia tiene unos costes computacionales extremadamente grandes, por lo que no son especialmente utilizadas como clasificadores, siendo especialmente utilizadas en campos como el procesamiento de imágenes, reducción de ruido (*denoising*) entre otros.

Existe un caso concreto de Red Bayesiana realmente simple que desde hace tiempo ha sido muy utilizado en diversos problemas de clasificación, mostrando resultados sorprendentemente buenos. A este tipo de redes se les llama *Naive Bayes*. Estas redes se caracterizan por tener un único nodo padre del cual descienden todos los hijos. Para nuestro caso concreto, el nodo padre sería la clase a la que pertenece la muestra observada, mientras que los nodos hijos serían las características de la muestra observada. El propósito sería inferir a qué clase (nodo padre) pertenece la muestra observando sus características (nodos hijos).

En estas redes bayesianas, esta inferencia se convierte en un proceso mucho menos pesado, puesto que se asume que los nodos hijos son todos independientes unos de otros, lo cual reduce considerablemente la cantidad de estimaciones necesarias en la inferencia.

Un punto muy favorable de este modelo es su simplicidad, que aunque pueda parecer poco efectivo, debido a que realmente esa independencia no es real puesto que las características suelen estar correladas entre sí, se ha convertido en un modelo ampliamente utilizado en diversos problemas de clasificación, logrando resultados sorprendentemente buenos con una complejidad mínima.

Entre las principales aplicaciones de este modelo se encuentra la detección de correos electrónicos basura (*SPAM*). Muchos programas de gestión de correo electrónico utilizan algoritmos basados en redes *Naive Bayes*.

3.3.2. Elección del clasificador

De los clasificadores expuestos anteriormente, el de las Redes Bayesianas nos parece el más adecuado debido principalmente a su simplicidad. Puesto que vamos a desarrollar una aplicación que tiene que clasificar muchísimas muestras, nos interesa que el proceso de clasificación sea lo más breve posible, por supuesto siempre dentro de unos determinados márgenes de calidad. Este clasificador tiene la ventaja adicional de

que existen ejemplos de otras aplicaciones que utilizan este tipo de clasificador, lo cual puede ayudar al desarrollo y al conocimiento del mismo.

3.4. Características (*features*)

En la Sección 2.2 se expusieron diversos trabajos de otros grupos de investigación que han realizado trabajos parecidos. Algunos de estos trabajos utilizaban la técnica de la correlación cruzada para la detección, con la cual no se utilizan características. Estos modelos suelen ser problemáticos, ya que requiere conocer la estructura tridimensional que estamos analizando. Otros de estos trabajos utilizaban técnicas alternativas basadas en características de las subimágenes obtenidas al marcar las partículas.

Estos modelos suelen ser más flexibles, debido a que se van amoldando mejor al tipo de estructura tridimensional analizada. Es por este último aspecto por el que consideramos que nuestro modelo debería utilizar características para clasificar.

3.4.1. Definición de característica

En esta sección se pretende estudiar el concepto de característica (*feature*) más a fondo para intentar de esta manera concretar acerca de qué características nos pueden interesar para generar nuestro modelo de detección.

Si analizamos cómo clasifica el ser humano determinados objetos como pueden ser un coche o una persona, está muy claro que utilizamos algunas características de estos objetos para dicha clasificación. Las personas tienen piernas, las cuales son una característica que los coches no tienen. En cambio, los coches tienen ruedas, mientras que las personas no.

Eligiendo correctamente un determinado conjunto de características, podemos obtener una buena clasificación. Esto requiere conocer algunas cualidades interesantes de lo que queramos clasificar para obtener el mejor clasificador posible. Volviendo al ejemplo anterior, el hecho de que un coche tenga ruedas lo distingue claramente de una persona. Sin embargo, no ayuda a distinguirlo de un tren puesto que éste también dispone de ruedas. Es por esto que esta elección de características es uno de los puntos claves de cualquier modelo de clasificación y por lo que requiere un estudio minucioso.

En primer lugar, veamos qué tipo de características son las más habituales en diversos problemas de microscopía electrónica para así intetar decidir cuáles nos pueden interesar más.

3.4.2. Características utilizadas en Microscopía Electrónica

En los trabajos de [14], [31] se describen algunas características empleadas para sus respectivos trabajos.

En el caso de Mallick, Zhu y Kriegman, las características las obtiene a partir de las imágenes integrales de las partículas/no partículas. Estas imágenes son muy eficientes computacionalmente hablando, pero requieren alinear todas las partículas en la fase de entrenamiento para posteriormente, en la fase de clasificación, ir rotando la subimagen analizada e ir calculando sus características hasta que éstas se parezcan lo suficiente a las características de los datos de entrenamiento.

Por otra parte, en el trabajo de Yu y Bajaj, se obtienen unas características a partir de la geometría de las partículas. Este modelo tampoco nos sirve, puesto que en nuestro trabajo pretendemos que sea posible obtener una única orientación de una determinada partícula estudiada (por ejemplo, las orientaciones laterales de la proteina KLH) y en este trabajo se describe un modelo para obtener todas las orientaciones. En cualquier caso, el hecho de incluir algo de información relacionada con la geometría de las partículas deseadas es interesante, ya que podría ayudar a descartar falsas partículas y elementos espurios.

Por tanto, tendremos que buscar alguna otra técnica que no se haya utilizado para detectar partículas.

3.4.3. Elección de las características del modelo

La elección de las características de un modelo de clasificación es una tarea realmente complicada en el caso de que no se conozca con la suficiente exactitud el conjunto de datos que estamos tratando. En nuestro caso, en la microscopía electrónica no hay demasiada información acerca de qué tipo de características podríamos emplear para llevar a cabo la detección, puesto que casi siempre la técnica utilizada para llevar a cabo la detección de partículas está basada en comparación de plantillas, la cual no requiere calcular ningún tipo de característica concreta, sino simplemente fabricar una plantilla a partir de varias micrografías y correlarla con la micrografía que estemos analizando en ese momento.

En la Sección 3.2 habíamos impuesto a nuestro modelo que utilizara algún mecanismo de detección que eliminara en la medida de lo posible la necesidad de tener que rotar los datos analizados hasta que concuerden con los datos de entrenamiento que tengamos almacenados.

Para conseguir esto, necesitamos encontrar unos datos en la micrografía que sean invariables a rotaciones. Un ejemplo de este tipo de cálculo, aunque demasiado simple podría consistir en el sumatorio de la subimagen que estemos viendo, ya que da igual la orientación de la partícula, que el sumatorio siempre acabará siendo el mismo. Lógicamente, este cálculo es demasiado simple para nuestros cálculos, pero la idea en la que se basa es muy interesante.

En [4] se expone un análisis de imágenes generadas por microscopios electrónicos con simetría rotacional. Estas simetrías rotacionales sobre las partículas son conseguidas mediante el espectro de potencia rotacional de éstas. En el siguiente apartado se describe en profundidad esta técnica.

Simetría rotacional sobre micrografías

Esta técnica es utilizada en análisis de micrografías para poder estudiar las características relevantes de éstas.

El método más utilizado en este campo se conoce como el Método de Superposición Óptica, en el cual se produce una imagen compuesta. Para un objeto con una simetría de m grados, se compone una imagen de m copias de la original con variaciones de $2\pi/m$ sobre el eje de simetría. Las características que cumplan con esta relación de simetría m se conservan, siendo desechadas todas las demás. A pesar de que esta técnica produce grandes mejoras en la imagen cuando se conoce la estructura de la partícula, puede ser bastante poco fiable en el caso de que no se conozca con exactitud, lo cual lleva a que exista una fase posterior de revisión por parte de algún experto.

En cambio, el método propuesto en [4], basado en el espectro de potencia de la micrografía, hace posible encontrar más fácilmente la simetría de la misma y además da un parámetro de calidad de cuán buena es esa simetría. Después de este primer paso, es posible obtener una imagen filtrada a partir de estas simetrías.

Este proceso explicado se lleva a cabo en una serie de pasos intermedios:

1. Análisis de la imagen

   Se obtiene un array de datos para que puedan ser procesados por un ordenador. Posteriormente se pasan estos datos a coordenadas polares (τ, ϕ) para expandir su densidad $\rho(\tau, \phi)$ en ondas cilíndricas

$$\rho(\tau, \phi) = \sum_{n=-\infty}^{\infty} g_n(\tau) \exp(in\phi) \tag{3.3}$$

donde $g_n(\tau)$ representa el peso de las n componentes azimutales de las ramas en un radio ρ. Esta relación ha de satisfacer la relación $g_{-n}(\tau) = g_n^*(\tau)$ puesto que la densidad ρ es real.

Integrando esta densidad dentro del radio a de la partícula, obtenemos el componente rotacional de las n ramas. A partir de este rotacional podemos definir el espectro de potencia de la componente n como

$$P_n = \xi_n \int_0^a |g_n(\tau)|^2 r dr \longleftarrow \xi_n = \begin{cases} 1 & , n = 0 \\ 2 & , n > 0 \end{cases} \tag{3.4}$$

El factor ξ_n se incluye para destacar que P_n tiene contribuciones tanto de $g_{-n}(\tau)$ como de $g_n * (\tau)$ para $n > 0$.

Si pintamos P_n, entonces ya tenemos el espectro de potencia de la imagen, el cual es un método ampliamente utilizado para representar el análisis de simetrías. A pesar de que se podría trabajar sobre los valores ópticos de la imagen, es más conveniente pasar al campo de Fourier, quedando

$$F(R, \Phi) = \sum_{n=-\infty}^{\infty} G_n(R) \exp(in(\Phi + \frac{\pi}{2})) \tag{3.5}$$

para obtener así los conjuntos de coeficientes conectados por una función de Fourier-Bessel

$$g_n = \int_0^{\infty} G_n(R) J_n(2\pi R r) 2\pi R dR \tag{3.6}$$

2. Determinación de la posición de origen:

   Puesto que este análisis lo estamos realizando en polares, es fundamental que el origen de estas coordenadas esté posicionado exactamente en el centro simétrico de la imagen. Esto puede resultar complicado a simple vista, pero numéricamente hablando es bastante sencillo.

   Primero se coge un centro simétrico aproximado y se utiliza este punto como la fase original para calcular las transformadas de Fourier y calculamos cuán alejado está ese punto de la simetría m suponiendo que la imagen tiene m grados de simetría, sumando las diferencias entre los valores de la transformada en un radio dado y los valores de la misma y a un desplazamiento de $360/m$. Desplazando

la fase original y recalculando este residuo, se produce un mapa bidimensional el cual tendrá el mínimo en el punto donde mejor se puedan obtener las simetrías.

Una vez ya tenemos nuestras imágenes filtradas, cabe destacar que pueden aparecer componentes de ruido pertenecientes a elementos no simétricos.

En la Figura 3.4 podemos ver un ejemplo del espectro rotacional de una partícula. El eje horizontal mide el número de la simetría, mientras que el eje vertical mide el valor de dicha simetría.

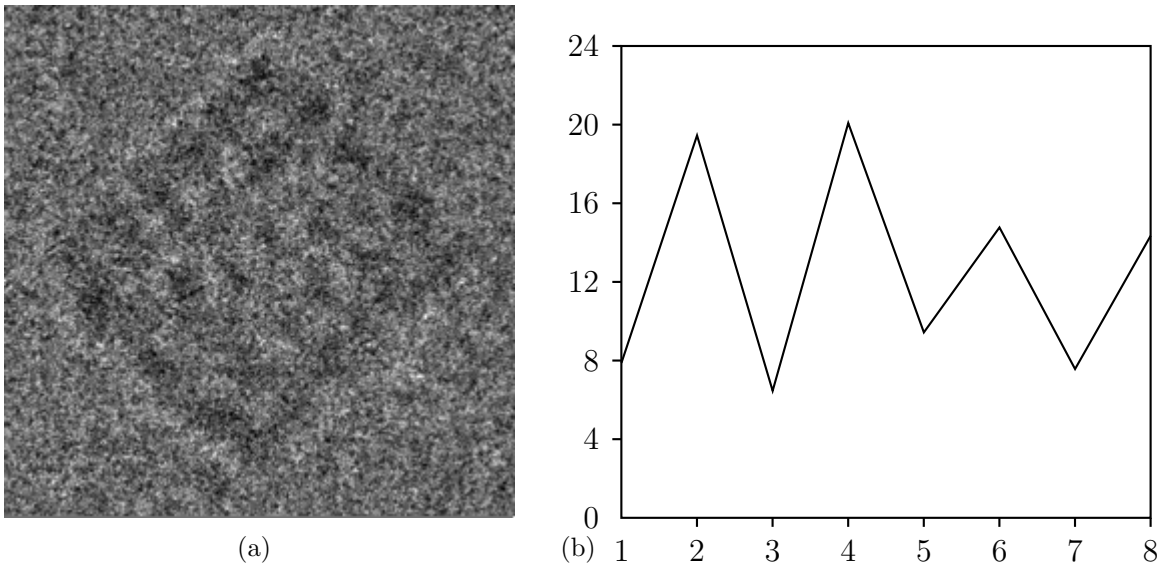


Figura 3.4: Espectro Rotacional de una partícula

Así pues, si calculamos este espectro rotacional a cada partícula, obtenemos una serie de características que son invariantes a rotaciones, el cual era uno de nuestros requisitos. Para nuestro modelo, hemos considerado un espectro rotacional de 15 simetrías. Así pues, en nuestro vector de características incluiremos 15 características que se corresponden con el espectro rotacional de la partícula.

Otras características de nuestro modelo

Hasta ahora hemos introducido en el modelo una serie de características invariables a rotaciones basadas en el espectro rotacional de las partículas. A parte de estas características, también será necesario tener bastantes características relacionadas con la

imagen en sí para poder facilitar la detección. Se podría decir que las características invariantes a rotaciones son un segundo filtro más fino para desechar posibles falsos positivos, pero antes hemos de tener algunas características más básicas que reconozcan un elemento diferente del fondo de la micrografía.

Para calcular estas características, se define una máscara para calcular las características a partir de los datos que caigan dentro de ella según se va desplazando por la micrografía. En la Figura 3.5(a) podemos ver el aspecto que tiene la máscara. Los píxeles que están en blanco son los que se considerarán como partícula. El resto en cambio son desechados. Esto hace que tengamos que desplazarnos de poco en poco por la micrografía de manera que no nos saltemos partículas. Vemos como es circular de 128 $\times$ 128 píxeles. Este valor puede ser redefinido por el usuario para escanear partículas de diferente tamaño.

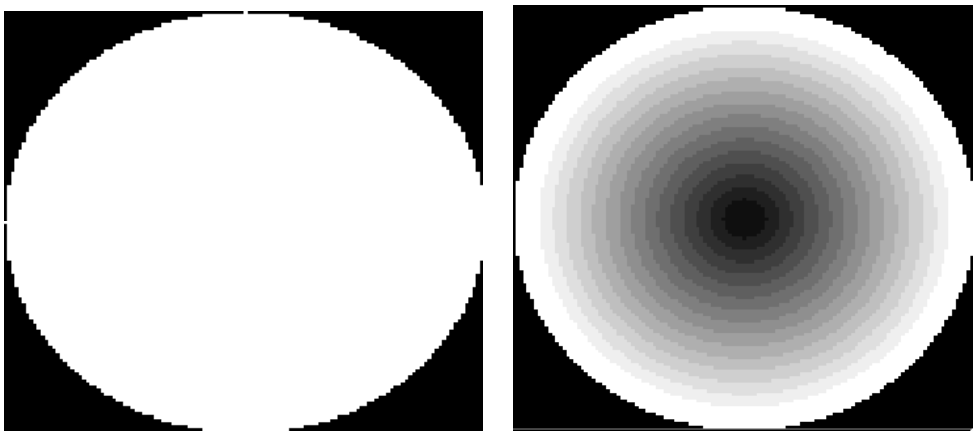
A continuación damos una breve explicación de algunas otras características que se han introducido. En el Capítulo 5 se explican en más detalle cada uno de los pasos del procesamiento de las partículas.

- En primer lugar, calculamos el histograma normalizado de la imagen. Normalizamos la imagen para evitar los efectos indeseados de variaciones de intensidad y de contraste. A continuación calculamos su histograma de 16 niveles, con lo que ya tenemos nuestras primeras 15 características (el histograma de 16 niveles tiene 15 intervalos).

- Posteriormente calculamos el histograma de cada uno de los anillos radiales de la partícula. Se definen una serie de anillos radiales para cada partícula, los cuales son anillos concéntricos que parten desde el centro de la partícula. En la Figura 3.5(b) podemos ver cómo son estos anillos radiales. Cada nivel de gris representa un anillo diferente. Se han considerado 16 anillos radiales. Estas características se definen así porque dan más precisión de qué valores son más frecuentes en cada parte de las partículas. Para cada anillo radial se calcula el histograma de 8 niveles de gris, con lo que tenemos 112 características más (16 radial bins $\times$ 7 intervalos de grises).

3.5. Características empleadas

Por tanto, ya tenemos nuestro vector de características. La longitud total de dicho



(a) Máscara utilizada para escanear las partículas

(b) Representación de los anillos radiales de una partícula

Figura 3.5: Representación de cómo se obtienen algunas características en nuestro modelo

0 HI ... 14	15 HAR ... 126	127 ER ... 141

Figura 3.6: Vector de características definitivo de nuestro modelo

vector es de 142 características. En la Figura 3.6 podemos ver la representación gráfica de este vector. En dicha figura, vemos por una parte los índices del vector y qué características están almacenadas en dichos índices. Las abreviaturas se corresponden con HI = Histograma de la Imagen, HAR = Histograma de los Anillos radiales y ER = Espectro Rotacional.

Así pues, la mayoría de las características que hemos introducido en el modelo son acerca de la imagen, ya que al ser estos más difíciles de clasificar, consideramos que tendrán que ser más. En la Sección 7.1 se explica una posible mejora a estas suposiciones.

3.5.1. Ponderación de las características

Para hacer el modelo más robusto, creemos que es bastante interesante introducir algún tipo de ponderación de características. Esta ponderación puede ayudar bastante al clasificador, ya que probablemente haya determinadas características del modelo que realmente no aporten demasiada información. Si estas características tienen el mismo peso que las características que realmente aportan información, pueden provocar que

el clasificador se equivoque, especialmente en aquellas partículas complicadas en las que la competición entre partícula/no-partícula está muy reñida. Además, una vez midamos la calidad de nuestro clasificador, podemos observar estas ponderaciones para ver qué características son menos importantes, de manera que puedan ser eliminadas para agilizar el proceso, o ser sustituidas por otras que aporten más información al clasificador.

Para ponderar las características, mediremos la cantidad información que aporta cada una de ellas. Para ello, utilizaremos la divergencia de *Kullback-Leibler*, la cual sirve para calcular cómo de diferentes son dos distribuciones de probabilidad.

En el Capítulo 5 entraremos más en detalle acerca de cómo es realmente esta ponderación.

3.6. Definición del clasificador

Ahora que ya tenemos claro cómo debe ser nuestro modelo, podemos definirlo formalmente. El modelo utilizado para abordar la detección automática de partículas es un clasificador *Naive Bayes* de 142 características como las representadas en la Figura 3.6. Intentaremos deducir si los datos observados $(F_1, \ldots, F_n)$ corresponden a una partícula o no, y además, a medida que vamos aprendiendo datos de diferentes micrografías y vayamos corrigiendo errores en éstas, iremos añadiendo datos a una tercera clase al clasificador *Naive Bayes*. Esta tercera clase corresponde a los errores cometidos por el clasificador en micrografías anteriores, de manera que el clasificador tiene tres opciones de clasificación para un conjunto de datos dado: partícula, error y otros (fondo, impurezas, no-partículas desechadas correctamente, etc.). Los nombres asociados a partir de ahora a cada una de estas clases son:

- Partícula: Clase que representa las partículas positivas.
- Error: Clase que representa las partículas marcadas erróneamente por el modelo.
- No-Partícula: Clase que contiene los elementos que no son ni partículas ni errores.

Capítulo 4

Introducción a las redes bayesianas

Una vez que ya hemos decidido en qué clasificador se basará nuestro modelo, necesitamos entrar más en materia acerca de cómo funciona exactamente este clasificador. En este capítulo entraremos en detalle acerca de qué son las Redes Bayesianas, cómo funcionan, así como el uso que podemos darle dentro del marco de nuestro modelo de clasificación.

4.1. Thomas Bayes (1702 - 1761)

Thomas Bayes fue un matemático británico del siglo XVIII que estudió el problema de la determinación de la probabilidad de las causas a través de los efectos observados.

El teorema que lleva su nombre se refiere a la probabilidad de un suceso condicionado por la ocurrencia de otro suceso. Bayes fue ordenado ministro presbiteriano.

Miembro de la *Royal Society* desde 1742, Bayes fue uno de los primeros en utilizar la probabilidad inductivamente y establecer una base matemática para la inferencia probabilística.

4.2. Teorema de Bayes

Su publicación tuvo lugar en el año 1763. Su expresión matemática se define en (4.1)

$$p(H|M,c) = \frac{p(H|c)p(M|H,c)}{p(M|c)} \tag{4.1}$$

donde podemos actualizar nuestras hipótesis H con las medidas M y el contexto c.

El término a la izquierda de la igualdad es conocido como la probabilidad *a posteriori* o lo que es lo mismo la probabilidad de H, considerando los efectos de M sobre c. El término $p(H|c)$ es conocido como la probabilidad *a priori* de H dado sólo c. El término $p(M|H,c)$ es comúnmente conocido como *likelihood* (verosimilitud) y es la probabilidad de la medida M, asumiendo la hipótesis H y que el contexto c se cumple. Por último, el último factor $p(M|c)$ es independiente de H y está considerado como un factor de normalización.

Cabe destacar que estas probabilidades son probabilidades condicionales, lo que significa que no tienen ningún valor si no calculamos primero las probabilidades asociadas.

4.2.1. Ejemplo de aplicación

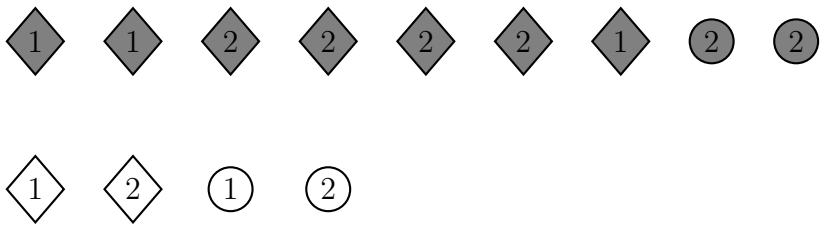


Figura 4.1: Conjunto de imágenes de ejemplo para aplicar el teorema de Bayes

Para entender mejor la utilización de este teorema, veamos un ejemplo sacado de [17]. Supongamos que tenemos el conjunto de imágenes de la Figura 4.1 y le asignamos a cada objeto una proabilidad de $\frac{1}{13}$. Asignamos el subconjunto Uno a los objetos que contengan un 1, Dos a los objetos que tengan un 2 y $Gris$ a los que sean de color gris. Entonces, según el Teorema de Bayes

$$p(Uno|Gris) = \frac{p(Gris|Uno)p(Uno)}{p(Gris|Uno)p(Uno)+p(Gris|Dos)p(Dos)}$$

$$= \frac{(\frac{3}{5})(\frac{5}{13})}{(\frac{3}{5})(\frac{5}{13})+(\frac{6}{8})(\frac{8}{13})} = \frac{1}{3}$$

A pesar de que este ejemplo no sea demasiado interesante puesto que era realmente simple calcular $p(Uno|Gris)$ directamente, nos sirve para ver una pequeña aplicación del Teorema de Bayes.

4.3. Definición de una red bayesiana

Las redes bayesianas son modelos probabilísticos gráficos que representan un conjunto de variables y sus dependencias. Por ejemplo, una Red Bayesiana podría representar la relación probabilística de unos síntomas con una determinada enfermedad. Dados unos síntomas, la red podría ser utilizada para calcular la probabilidad de que exista alguna enfermedad concreta (ver [2]).

Una red bayesiana sobre el dominio U representa una función de probabilidad conjunta sobre dicho dominio. Esta representación consiste en una serie de probabilidades conjuntas condicionales, así como de ciertas presunciones de independencia, que permite construir una función de probabilidad conjunta global a partir de funciones de probabilidad conjunta locales. Para obtener estos valores, se puede simplificar mediante la regla de la cadena

$$p(x_1, ...x_n|x_i) = \prod_{i=1}^{n} p(x_i|x_1, ...x_{i-1}, x_i) \tag{4.2}$$

En los años 80, hubo numeroros trabajos de investigación orientados hacia el desarrollo de las Redes Bayesianas (redes de confianza, redes causales, diagramas de influencia, etc.), algoritmos de inferencia y aplicaciones para su utilización.

Para entender mejor este tipo de redes, debemos explicar brevemente algunos conceptos de Modelos Gráficos, ya que las redes bayesianas se basan principalmente en el diagrama o grafo que la representa.

4.4. Modelos gráficos

Los modelos gráficos son diagramas en los cuales los nodos representan variables aleatorias, mientras que las flechas o arcos representan dependencia probabilística (por tanto, la ausencia de ellos representa independencia).

Los **Grafos no dirigidos**, también llamados campos aleatorios de Markov, tienen una definición de independencia simple: dos nodos A y B son independientes dado un tercer nodo C, si todos los caminos entre A y B deben pasar por C.

En cambio, los **Grafos dirigidos**, también llamados redes bayesianas, se caracterizan por tener una definición de independencia algo más compleja donde también entra en juego la dirección de los arcos. Este tipo de grafos, en realidad no utilizan ningún método bayesiano para su construcción ni nada parecido, simplemente utilizan

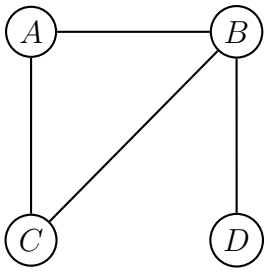


Figura 4.2: Ejemplo de un grafo no dirigido

el teorema de Bayes para hacer inferencias sobre la red.

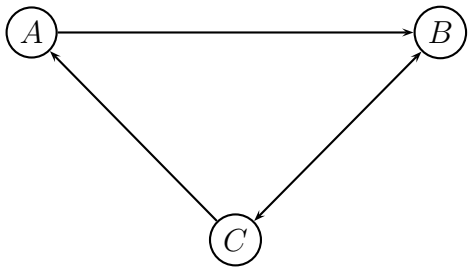


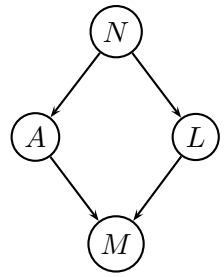
Figura 4.3: Ejemplo de un grafo dirigido

A parte del grafo del modelo, también es necesario especificar los parámetros del modelo. Para un grafo no dirigido, deberemos especificar una función de potencial para cada arco, mientras que para las redes bayesianas debemos especificar la distribución condicional de probabilidad para cada nodo. En el caso de que estas variables sean discretas, esta distribución se puede representar como una tabla de probabilidades del nodo hijo según el valor de los nodos padres.

En la Figura 4.4 podemos ver un ejemplo de una Red Bayesiana con sus respectivos parámetros. Éste ejemplo aparece explicado en [16]. Resumiendo, se trata de un modelo creado a partir de un jardín con un césped. Dicho césped puede estar mojado o no (variable aleatoria M) dependiendo de si llueve (L) o si el aspersor está encendido (A). También existe otra variables que describe si el cielo está nublado o no (N). En las tablas tenemos las probabilidades condicionales según un valor para el nodo padre ($C = Cierto$ y $F = Falso$).

Podemos ver que el hecho de que el césped esté mojado ($M = cierto$) puede venir dado por dos condiciones diferentes. Por una parte, puede estar lloviendo ($L = cierto$), pero también podría ser que el aspersor esté encendido (A=cierto). La dependencia de

p(N=F)	p(N=C)
0.5	0.5



N	p(A=F)	p(A=C)
F	0.5	0.5
T	0.9	0.1

N	p(L=F)	p(L=C)
F	0.8	0.2
T	0.2	0.8

A	L	p(M=F)	p(M=C)
F	F	1.0	0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.09

Figura 4.4: Red bayesiana de ejemplo

estas relaciones se puede ver en la tabla. Por ejemplo, podemos ver que la probabilidad de que el césped esté mojado si el aspersor está encendido ($p(M = cierto|L = falso, A = cierto)$) es de 0.9, lo cual implica que la probabilidad de que no esté mojado dadas las mismas condicione es de $1 - 0{,}9 = 0{,}1$ ($p(M = falso|L = falso, A = cierto)$). Puesto que el nodo *Nublado* (N) no tiene padres, su función de probabilidad conjunta sería simplemente la probabilidad *a priori* de que esté nublado, la cual sin tener más datos es del 50 %.

La relación probabilística más simple que podemos ver en una red bayesiana es la que acabamos de ver. Podemos deducir que la probabilidad de que el césped esté mojado no depende de si el cielo está nublado o no, siempre que conozcamos si está lloviendo o si el aspersor está encendido. Es decir, que un nodo es independiente de sus nodos ancestrales (nodos que están jerárquicamente situados por encima de sus nodos padres) siempre que conozcamos el valor de los nodos padres.

Por la regla de la cadena probabilística, la probabilidad conjunta de todos los nodos de la Figura 4.4 quedaría como en (4.3).

$$p(N, A, L, M) = p(N)p(A|N)p(L|N, A)p(M|N, A, L) \tag{4.3}$$

Utilizando reglas de independencia podemos reescribir la ecuación como

$$p(N, A, L, M) = p(N)p(A|N)p(L|N)p(M|A, L) \tag{4.4}$$

donde hemos modificado el tercer término, ya que L es independiente de A dado N. Además, también hemos cambiado el último término, ya que la probabilidad de que el césped esté mojado M no depende de si el cielo está nublado N siempre y cuando podamos observar si llueve L y si el aspersor está encendido A.

Con este ejemplo tan simple hemos podido ver cómo se pueden simplificar enormemente las relaciones probabilísticas utilizando una red bayesiana y las independencias implícitas que ésta conlleva. Debido a la simplicidad de este ejemplo, los ahorros sobre éste son mínimos, pero en general, si tenemos n nodos binarios ($cierto/falso$), entonces la probabilidad conjunta requeriría $O(2^n)$ para ser representada, mientras que con las simplificaciones aplicadas requeriría $O(n2^k)$, donde k es el numero máximo de entradas de un nodo.

4.5. Inferencia

La tarea más común para la que se utilizan las Redes Bayesianas es la **inferencia probabilística**. Por ejemplo, volviendo al ejemplo anterior del césped, supongamos que conocemos que el césped está mojado. Entonces, sabemos que el césped puede estar mojado por dos causas diferentes. O bien está lloviendo, o es que el aspersor está encendido. Estudiar cuál de las dos causas es más probable es lo que se conoce como inferencia probabilística. Esta inferencia se puede llevar a cabo aplicando el teorema de Bayes para conocer las probabilidades *a posteriori* de cada una de las hipótesis. (A partir de ahora consideraremos que $cierto = 1$ y $falso = 0$).

$$p(A = 1|M = 1) = \frac{p(A=1,M=1)}{p(M=1)} = \frac{\sum_{n,l} p(N=n,A=1,L=l,M=1)}{p(M=1)}$$

$$= 0{,}2781/0{,}6471 = 0{,}430$$

$$p(L = 1|M = 1) = \frac{p(L=1,M=1)}{p(M=1)} = \frac{\sum_{n,a} p(N=n,A=a,L=1,M=1)}{p(M=1)}$$

$$= 0{,}4581/0{,}6471 = 0{,}708$$

donde

$$p(M = 1) = \sum_{n,l,a} p(N = c, L = l, A = a, M = 1) = 0{,}6471$$

es una constante de normalización. Por lo tanto, podemos ver que es más probable que el césped esté mojado porque esté lloviendo que porque el aspersor esté encendido.

Es importante resaltar que las dos causas compiten para explicar los datos, lo cual es lo mismo que decir que L y A se convierten en dependientes una vez estamos observando su nodo hijo M, a pesar de que son marginalmente independientes. Este tipo de razonamiento se conoce como diagnóstico o "de abajo a arriba", puesto que razona las causas a partir de los efectos. Las redes bayesianas también permiten razonamientos opuestos ("de arriba a abajo"). Un ejemplo podría ser intentar averiguar si el césped está mojado sabiendo que el cielo está nublado.

Este proceso de inferencia es para lo que normalmente se utilizan las redes bayesianas. Sin embargo, el tiempo que cuesta calcular todas estas probabilidades crece exponencialmente con el número de nodos, por lo que un gran campo de estudio es el de algoritmos de optimización para procesos de inferencia en redes bayesianas. Para poder optimizar este proceso, es necesario conocer muy bien el sistema que se está modelando, es decir, las dependencias/independencias existentes entre las posibles variables estudiadas del sistema en cuestión.

4.6. Aplicación de las redes bayesianas al modelo estudiado

Como se ha explicado anteiormente en la Sección 4.5, la inferencia pretende buscar los valores de unas variables dependiendo de lo que observemos en otras.

En nuestro caso, lo que pretendemos es, a partir de un conjunto de características, inferir si lo que estamos analizando es una partícula o no lo es. Para ello queremos construir una red bayesiana que represente nuestro modelo.

En la sección anterior vimos que la inferencia en redes bayesianas conlleva unas limitaciones computacionales muy grandes que crecen exponencialmente con el número de nodos. Puesto que en nuestro caso estamos considerando 142 características y además queremos que nuestro programa sea relativamente rápido, no podemos asumir el coste de construir una red bayesiana compleja como las estudiadas hasta el momento. Por este motivo, debemos estudiar alguna solución a este problema.

Existe un clasificador que se basa en las redes bayesianas llamado clasificador *Naive Bayes Classifier* que se adecúa perfectamente a nuestras pretensiones y que además es

increiblemente simple desde un punto de vista computacional.

4.7. El clasificador *Naive Bayes*

4.7.1. Introducción

Un clasificador *Naive Bayes* es un modelo de probabilidades muy simple que se basa en el teorema de Bayes con un presunción de independencia muy grande. En la Figura 4.5 podemos observar el modelo gráfico asociado a este tipo de clasificador.

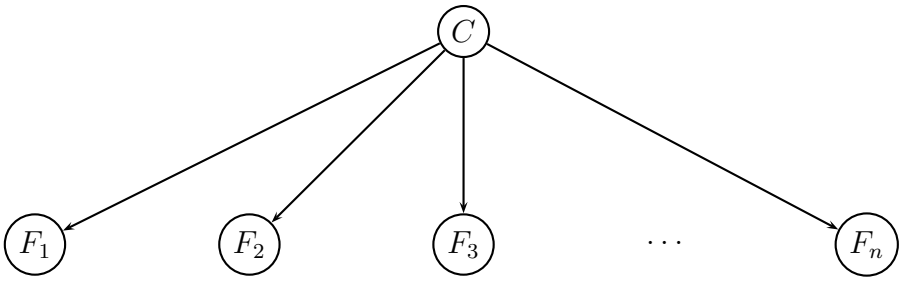


Figura 4.5: Red bayesiana de tipo *Naive Bayes*

Observando la figura podemos ver cómo sólo existe un nodo raíz C del que descienden todos los demás $F_1, \ldots, F_n$. Lo que es realmente importante de esta figura es que no existe ningún arco entre los nodos hijos, lo cual quiere decir que son independientes.

Existen algunas variaciones de este tipo de red bayesiana, siendo la más utilizada la *Augmented Naive Bayes* más conocida como ANB, la cual asume cierta dependencia entre algunas de las características (nodos hijos). En la Figura 4.6 podemos ver un ejemplo de este tipo de red.

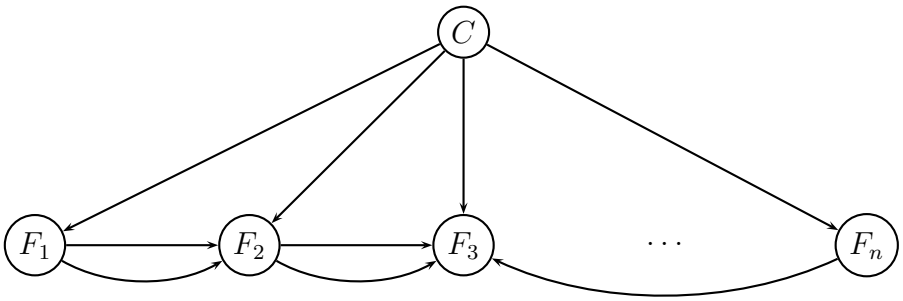


Figura 4.6: Ejemplo de una red ANB (*Augmented Naive Bayes*)

Dependiendo de la naturaleza del modelo que se esté estudiando, este clasificador puede ser entrenado muy eficientemente en un sistema de aprendizaje supervisado (ver Sección 3.3.1). En muchas aplicaciones, este tipo de clasificador hace estimaciones de parámetros a partir de una técnica conocida como *Maximum Likelihood*, con lo que en este caso la estructura realmente no tendría nada que ver con una red bayesiana.

A pesar de su gran simplicidad estructural y de que sus suposiciones son demasiado simplificadas, los clasificadores de este tipo han demostrado funcionar muchísimo mejor de lo que cabría esperar en algunas situaciones reales de gran complejidad. Una explicación de este buen resultado de tan simples clasificadores la podemos encontrar en [32], en el cual se discute que este buen resultado en algunos campos se debe a que esta presunción de independencia no real no influye demasiado en el caso de que las dependencias reales existentes en el modelo estén distribuidas equitativamente por los nodos. En este caso, se podría considerar que estas dependencias se anulan entre sí, pudiendo interpretar cierta independencia en el modelo global.

Una de las grandes ventajas de este modelo es que requiere muy pocos datos de entrenamiento para estimar los parámetros necesarios (medias y varianzas) para un determinado problema de clasificación. Esto es debido a la gran independencia entre variables considerada, lo que evita tener que calcular matrices de covarianza.

4.7.2. Definición

El modelo de probabilidades para un clasificador *Naive Bayes* es un modelo condicional del tipo $p(C|F_1, ..., F_n)$, sobre un conjunto de clases C con un número normalmente pequeño de posibles valores sobre un conjunto de características *features* desde F_1 hasta F_n. En la mayoría de situaciones, n suele ser de gran tamaño. Esto último supone un gran problema si queremos construir unas tablas para las correspondientes varianzas, lo cual nos lleva a reformular el modelo de otra forma. Utilizando el teorema de Bayes anteriormente descrito, podemos obtener la Ecuación (4.5).

$$p(C|F_1, ..., F_n) = \frac{p(C)p(F_1, ..., F_n|C)}{p(F_1, ..., F_n)} \longrightarrow posteriori = \frac{priori \times likelihood}{medida} \qquad (4.5)$$

En la práctica lo que nos interesa de este fórmula es el numerador de la fracción, ya que el denominador es constante y está considerado como un factor de normalzación. Este numerador es equivalente a la distribución conjunta de probabilidad $p(C, F_1, ..., F_n)$

y puede ser reescrito como en la Ecuación 4.6

$$p(C, F_1, ..., F_n) = p(C)p(F_1)p(F_2, ..., F_n) = p(C)p(F_1)p(F_2)p(F_3, ..., F_n) \tag{4.6}$$

Y así sucesivamente. Ahora es cuando entran en juego las suposiciones de independencia impuestas por el modelo. Si suponemos que una característica F_i es independiente de otra característica F_j siendo $i \neq j$, entonces nos queda

$$p(F_i|C, F_j) = p(C)p(F_i|C) \tag{4.7}$$

y por tanto, el modelo conjunto puede ser expresado como

$$p(C|F_1, ..., F_n) = p(C)p(F_1|C)p(F_2|C)p(F_3|C)...p(F_n|C) = p(C)\prod_{i=1}^{n} p(F_i|C) \tag{4.8}$$

Esta última condición nos indica que bajo la suposición de independencia, la distribución condicional sobre la clase C puede ser expresada como

$$p(C|F_1, ..., F_n) = \frac{1}{Z}p(C)\prod_{i=1}^{n} p(F_i|C) \tag{4.9}$$

donde Z es una constante que depende de $F_1, ..., F_n$, es decir, una constante en el caso de que los valores de las características sean conocidos.

Los modelos de este tipo son bastante más manejables, ya que se pueden factorizar en una probabilidad *a priori* de la clase $p(C)$ y una serie de distribuciones $p(F_i|C)$. Si hubiera K clases, y si el modelo para $p(F_i)$ pudiera ser expresado en función de r parámetros, entonces el modelo *naive bayes* resultante tendría $(K-1) + nrk$ parámetros. En la práctica, normalmente $K = 2$ (clasificación binaria) y $r = 1$ (variables de Bernouilli utilizadas como características), entonces el modelo tiene $2n + 1$ parámetros, siendo n el número de características binarias utilizadas para la detección.

Los parámetros del modelo (probabilidades de las clases, distribuciones de las características) normalmente son aproximados a partir de frecuencias relativas (número de muestras en un determinado intervalo/número total de muestras). Por ejemplo, en nuestro caso de las partículas, las probabilidades *a priori* de las clases se obtedrán calculando el área total de partículas existente en los datos de entrenamiento y dividiendo por el área total de las micrografías de entrenamiento (ver el Capítulo 5 para más in-

formación). Además, cabe resaltar que las características que no sean discretas deberán ser discretizadas antes para poder utilizar este modelo. En este último paso hay que tener cuidado, ya que si uno de esos intervalos de discretización no ocurriese nunca en ninguna de las muestras de la fase de entrenamiento (por ejemplo, la característica 47 nunca tiene un valor entre 0 y 10 suponiendo que uno de los intervalos discretizados fuera exactamente ese), entonces esa característica tendría una frecuencia relativa de 0, lo cual es bastante problemático puesto que estamos multiplicando probabilidades. Es por esto que habrá que hacer algún tipo de mínima corrección a los datos obtenidos por frecuencias relativas, ya que en la realidad ninguna probabilidad será exactamente 0. Lo normal en este tipo de casos es aplicar una pequeña corrección a todos los datos, por ejemplo, sumando 1 a todos los elementos. De esta manera, afectamos a todos los datos por igual y no al resultado global de la clasificación.

4.7.3. Construcción del clasificador

Hasta ahora hemos discutido en qué consite el modelo de probabilidades que utiliza un clasificador *naive bayes*. Ahora estudiemos cómo clasifica exactamente éste.

Para clasificar, lo único que hace el clasificador *naive bayes* es aplicar una regla de decisión. La regla más común para este clasificador consiste simplemente en adoptar la hipótesis más probable a partir de los datos obtnidos. Esto se conoce como la Probabilidad *A posteriori* Máxima (*MAP*). Con esto, nuestro clasificador quedaría como

$$c(f_1, \ldots, f_n) = argmax_c\{p(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c)\} \tag{4.10}$$

donde obtendríamos una clase resultado c según un conjunto de características dado $f_1, \ldots, f_n$. Esta ecuación, simplemente recorre los nodos hijos (uno por característica del modelo) y pregunta a cada uno por la probabilidad de que un determinado valor de característica pertenezca a cada una de las clases del modelo. El nodo hijo devuelve un conjunto de probabilidades (una probabilidad por clase) y el modelo va multiplicando asociativamente todas estas probabilidades para cada una de las clases. Una vez hemos recorrido todos los nodos hijos, ponderamos las probabilidades acumuladas por la probabilidad *a priori* de cada una de las clases, obteniendo así las probabilidades *a posteriori* de las mismas. La clase resultante será aquella que tenga una probabilidad *a posteriori* más alta.

Por tanto, ya disponemos de nuestro clasificador de partículas, por lo que podemos

pasar a la parte de implementación.

Capítulo 5

Implementación del modelo

5.1. Notas de la implementación

Para realizar este proyecto, se ha utilizado el paquete de programas para el procesamiento de imágenes microscópicas XMIPP [27]. La implementación se ha realizado en C++ sobre UNIX (Linux SuSe 10.0) en un PC IBM ThinkPad con un procesador Pentium Intel Centrino 1.4 GHz y una memoria RAM de 256 MB.

5.2. Estructura de XMIPP

En esta sección no queremos hablar demasiado en la estructura de XMIPP, ya que es un tema demasiado extenso. Para más información acerca de XMIPP, ver [27]. Sin embargo, es conveniente conocer un mínimo acerca de la estructura de directorios de XMIPP, puesto que tendremos que modificar algunos ficheros de diferentes directorios de la estructura.

Para nuestros propósitos, simplemente debemos conocer dos directorios fundamentales de XMIPP. Estos son el directorio `libraries` y el directorio `applications`. En el primero se encuentran las librerías de las que se valen todos los programas existentes en el paquete, mientras que en el segundo se encuentran los programas propiamente dichos.

Puesto que nosotros vamos a partir de un programa existente en el paquete llamado `xmipp_micrograph_mark`, necesitamos saber que ese programa se encuentra en el directorio `xmipp/applications/programs/micrograph_mark`.

Además, también crearemos un nuevo clasificador *Naive Bayes*, el cual incluiremos

en el directorio de librerías de XMIPP, de manera que pueda ser utilizado por otros programas del paquete. En el directorio `xmipp/libraries/classification` existen numerosos clasificadores existentes en el paquete. Por tanto, es aquí donde debemos incluir nuestro nuevo clasificador.

5.3. Estructura del programa

El programa `xmipp_micrograph_mark` consta de los siguientes ficheros asocados:

```
auto_menu.cpp
auto_menu.h
color_label.cpp
color_label.h
dialog_families.cpp
dialog_families.h
dialog_properties.cpp
dialog_properties.h
popup_menu_mark.h
file_menu.cpp
file_menu.h
filters_controller.cpp
filters_controller.h
filter.cpp
filter.h
filter_menu.cpp
filter_menu.h
image.cpp
image.h
image_micrograph.cpp
image_micrograph.h
image_overview_micrograph.cpp
image_overview_micrograph.h
image_converter.cpp
image_converter.h
main_widget_mark.cpp
main_widget_mark.h
main.cpp
widget_micrograph.cpp
widget_micrograph.h
widget_psd.cpp
widget_psd.h
```

Los ficheros que realmente nos interesan para modificar el funcionamiento del programa son `widget_micrograph.cpp` y `widget_micrograph.h`, ya que son los que controlan el marcado de partículas a un nivel más global. A lo largo del proyecto se han tenido que hacer pequeños ajustes en algunos ficheros adicionales, pero que no se explicarán aquí debido a su escasa importancia en lo que al marcado automático de partículas se refiere.

Los ficheros `widget_micrograph`, tienen las definiciones de una serie de objetos utilizados por el programa. Los que más nos interesan son el objeto `Particle` y el objeto `Classification_model`, los cuales contienen los datos de las partículas y del modelo de clasificación repectivamente. Hay que resaltar que el objeto `Particle` no tiene por qué representar una partícula. Realmente se asocia con unos determinados píxeles de la micrografía, los cuales poseen una serie de características. Por ejemplo, la clase No-Partícula contendrá muchísimos objetos `Particle` que representan puntos de la micrografía donde no hay partículas, como el fondo, las impurezas, etc.

Un objeto `Particle` consta de los siguientes elementos

- Sus coordenadas (x, y) en la micrografía.
- El índice dentro de la lista de partículas. Este índice es incremental, se va asociando según se van marcando partículas en la micrografía. En el caso de que el objeto no esté asociado a la clase Partícula, este índice es irrelevante.
- El vector de características asociado a la partícula.

Por su parte, un objecto `Classification_model` contiene:

- Un vector de dos dimensiones, que contiene los objectos `Particle` para cada una de las clases existentes: Partícula, Error y No-Partícula.
- El número de micrografías analizadas.
- Un vector con las áreas de las micrografías analizadas.
- Un vector con el número de partículas marcadas en las micrografías anteriores.
- Un vector con el número de errores cometidos por el modelo en micrografías anteriores.

Los últimos cuatro elementos se utilizan para calcular las probabilidades *a priori* para cada una de las clases.

Además de estos objetos descritos, estos ficheros contienen toda la lógica de las fases de entrenamiento y clasificación.

5.4. Breve introducción al programa

Antes de explicar cómo hemos desarrollado los nuevos elementos del programa, primero explicaremos brevemente cómo funciona el programa `xmipp_micrograph_mark`.

En la Figura 5.1 podemos ver el programa en ejecución. Dispone de dos ventanas. Una primera ventana principal donde vemos la micrografía completa y una segunda ventana secundaria donde vemos una subimagen de la micrografía. Al pinchar con el ratón en alguna zona de la micrografía primaria, la ventana secundaria muestra una subimagen alrededor de dicha zona. Para marcar una partícula, tenemos que pinchar con el ratón en la subimagen el punto donde queremos marcar la partícula.

Además disponemos de tres menús en la parte superior: *File*, *Filters* y *AutoSelection*. El primero de ellos controla algunos parámetros globales del programa, como dónde

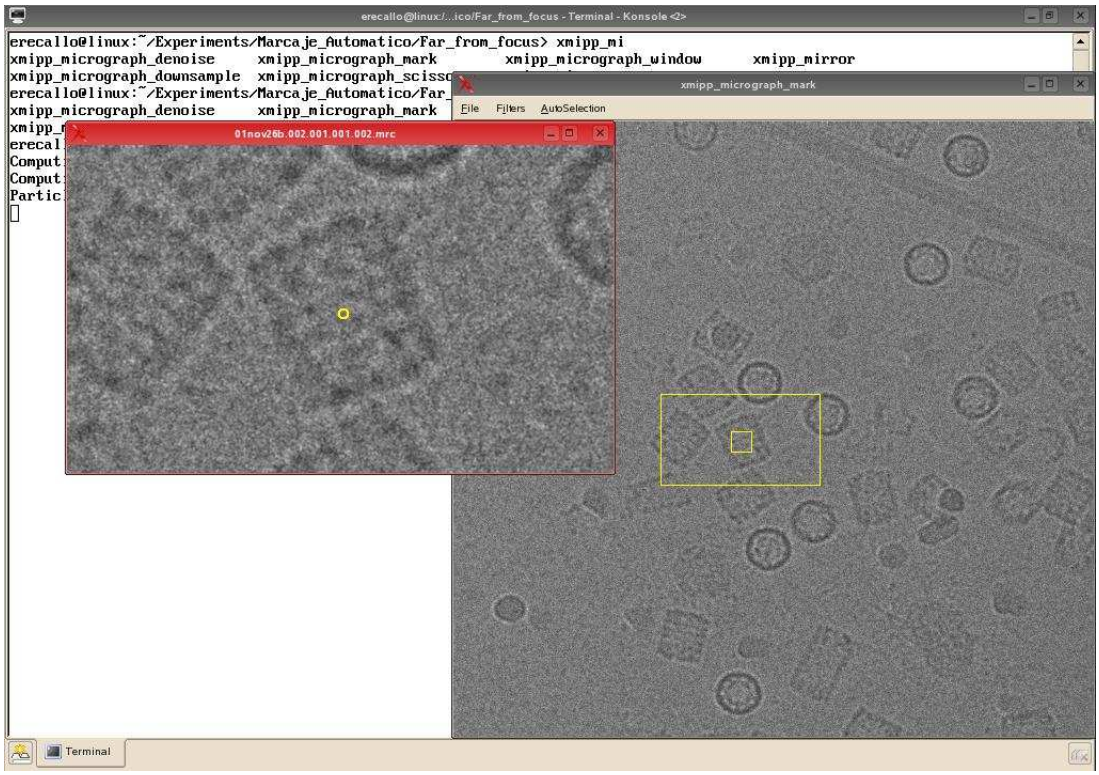


Figura 5.1: Imagen del programa en ejecución

guardar las partículas marcadas, cargar partículas de un fichero, etc. El menú *Filters* permite controlar algunos parámetros relacionados directamente con la imagen, como por ejemplo ajustar el contraste o aplicar filtros a la imagen. Por último, el menú que nos insteresa, *AutoSelection*, nos permite aprender partículas, buscar partículas, guardar datos del modelo, cargar datos del modelo y por último, controlar algunos parámetros de clasificación como por ejemplo el tamaño de las partículas.

Hasta ahora, el programa sólo era utilizado para marcar partículas y guardar sus coordenadas, las cuales eran utilizadas posteriormente por otros programas que también forman parte del proceso de la reconstrucción tridimensional. Este proceso se hacía marcando las partículas presentes en la imagen, y guardándolas con la opción de menú *File→Save Coords*.

5.5. Nuevas funciones del programa

A pesar de que el menú *AutoSelection* ya existía, no tenía funciones asociadas. Este desarrollo está orientado a asociar nuevas funcionalidades con estas opciones de menú. Éstas son:

1. *Configure*: Esta función permite al usuario modificar algunos parámetros de búsqueda a través de un diálogo flotante. Los parámetros más importantes son *Particle radius* y *Mask overlap*, los cuales sirven para definir el radio de la partícula y el solapamiento de la máscara al escanear la micrografía en busca de partículas (más adelante se explicará mejor este proceso).

2. *Load model*: Permite cargar los datos de partículas, errores, etc. de algún modelo que hayamos almacenado anteriormente.

3. *Learn particles*: Aprende los datos de las partículas marcadas actualmente. Además, si hemos borrado alguna partícula, la aprenderá como error.

4. *AutoSelect*: Escanea la micrografía en busca de partículas.

5. *Save model*: Guarda los datos aprendidos de la micrografía en un determinado modelo.

5.6. División del desarrollo

Para facilitar la lectura de este capítulo, se ha dividido la parte de desarrollo en dos partes (módulos). Por una parte explicaremos las modificaciones realizadas sobre el programa ya existente `xmipp_micrograph_mark` para crear, entrenar y comunicarse con el nuevo modelo de clasificación. A esta parte la llamaremos **Módulo de Gestión**, mientras que a la parte del modelo de clasificación lo llamaremos **Módulo de Clasificación**.

Empezaremos explicando el Módulo de Clasificación, puesto que es instanciado por el Módulo de Gestión y parece más fácil explicar éste último conociendo cómo funciona el clasificador.

5.7. Módulo de Clasificación

Este módulo no existía anteriormente en XMIPP, por lo que tendremos que crearlo. La función del módulo será crear un clasificador *Naive Bayes* y entrenarlo, para posteriormente, clasificar unos datos recibidos en un número de clases determinado. Para ello, simplemente crearemos dos nuevos ficheros, `naive_bayes.cpp` y `naive_bayes.h` en el directorio de clasificación de XMIPP descrito anteriormente.

Las funciones de este módulo deben ser las siguientes:

1. Recibir unas determinadas características de aprendizaje $F_1^k, \ldots, F_n^k$, donde n es el número de características del modelo y k es el indicador de la clase (en este paso debemos recibir datos de aprendizaje para todas las clases) y unas probabilidades *a priori* (para nuestro caso, deberían ser las probabilidades de que lo que estamos analizando sea una partícula, un error o simplemente fondo, es decir, la frecuencia relativa de las diferentes clases en la micrografía).

2. Crear la estructura del clasificador *Naive Bayes* con todos los nodos hijos correspondientes a las características (142 nodos en nuestro caso).

3. Entrenar el modelo con los datos de aprendizaje, para así tener el clasificador preparado para clasificar.

4. Clasificar un conjunto de características que le puedan llegar.

Hay que destacar, que los puntos 1, 2 y 3 son fases síncronas, es decir, que se realizan en el momento de inicializar el modelo, mientras que el punto 4 es asíncrono, ya que una vez que el clasificador esté entrenado, en cualquier momento se le puede pedir que clasifique una serie de datos.

Para ver más intuitivamente cómo está distribuido el módulo del clasificador, supongamos que tenemos el clasificador *Naive Bayes* de la Figura 5.2. Podríamos considerar que lo que llamamos clasificador es el nodo C, que es el que realmente nos acaba diciendo si un determinado conjunto de datos pertenece a una clase o a otra. El clasificador, a su vez, debe disponer de un nodo hijo por cada característica F_i, de manera que cada nodo hijo almacene los valores de esa característica F_i en concreto para todos los vectores de entrenamiento. A partir de estos datos almacenados, cada nodo hijo F_i calcula su histograma de D niveles, para a continuación, calcular sus frecuencias relativas P_d^k, donde d es el índice del nivel del histograma y k es la clase.

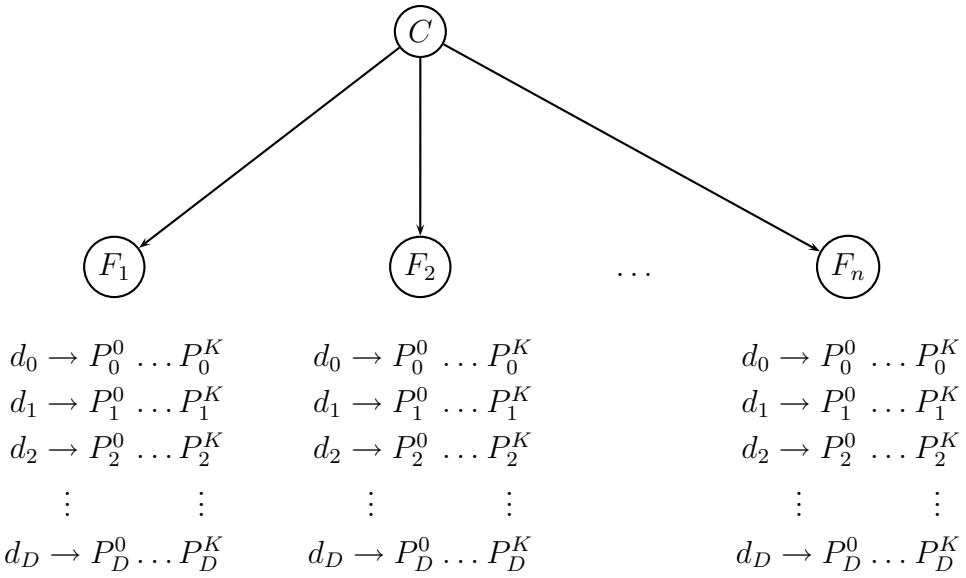


Figura 5.2: Representación gráfica de nuestro clasificador con las respectivas tablas de datos

En lo que a la programación de este módulo se refiere, se ha creado una clase que ejerce de clasificador C, llamada `xmippNaiveBayes` y una clase que ejerce de nodo hijo `LeafNode`. Por tanto, un elemento de tipo `xmippNaiveBayes`, al inicializarse debe crear n elementos `LeafNode`, donde n es el número de características del modelo.

5.7.1. Entrenamiento de la red bayesiana

Puesto que la red bayesiana es genérica y está incluida en uno de los directorios de linbrerías de XMIPP, vamos a explicar sus diferentes fases a parte, ya que podría ser que sus datos no fueran características de partículas, sino de cualquier otro tipo de dato. Para esta fase, se presupone que a la red bayesiana le llegan una serie de datos que se han obtenido por algún método independiente a la propia red.

Datos de entrenamiento

Los datos a partir de los cuales podemos inicializar la red son:

- Un vector A con tantos elementos como clases K tenga el modelo. Cada elemento del vector, a su vez, debe contener una matriz bidimensional E en la que estén los vectores de entrenamiento. El número de filas de esta matriz es el número de vectores de entrenamiento que estemos utilizando, cada uno con tantas columnas

como características hayamos calculado. Es decir, tendríamos una matriz E de elementos e_{vc}, donde el subíndice v indica el vector de entrenamiento (la fila) y c la característica (la columna).

$$A = (E_1, E_2, \ldots, E_K) \longrightarrow E_i = \begin{pmatrix} e_{11} & e_{12} & e_{13} & \ldots & e_{1n} \\ e_{21} & e_{22} & e_{23} & \ldots & e_{2n} \\ e_{31} & e_{32} & e_{33} & \ldots & e_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ e_{V1} & e_{V2} & e_{V3} & \ldots & e_{Vn} \end{pmatrix}$$

- Un vector de probabilidades P. Este vector debe contener las probabilidades *a priori* para cada una de las clases, por tanto, tendrá tantos elementos como clases contenga el modelo.

$$P = [p(C = 0), p(C = 1), \ldots, p(C = K)]$$

Entrenamiento del clasificador

Una vez que ya hemos recibido los datos de entrenamiento, podemos pasar a la fase de entrenamiento propiamente dicha. El entrenamiento de la red bayesiana consta de los siguientes pasos:

1. Para cada V_k donde k es el índice de la clase, se extraen las columnas de su matriz de entrenamiento (cada columna contiene los valores de una determinada característica para todos los vectores de entrenamiento). Por ejemplo, la columna 1 de la matriz de entrenamiento de la clase Partícula (k=0), contendrá el valor que ha tomado la característica 1 para cada Partícula de entrenamiento. La columna 1 de la matriz de entrenamiento de la clase Error (k=2), contendrá el valor que ha tomado la característica 1 para cada Error encontrado en las anteriores micrografías y así para cada clase. Lógicamente, cada nodo debe ser inicializado con la columna correspondiente de cada una de las matrices de entrenamiento de cada clase.

2. Se crea un nodo (`LeafNode`) para cada característica. Por ejemplo, si existen dos clases, cada una con un vector de 142 características, se crearán 142 nodos hijos $F_1, F_2, \ldots, F_{142}$.

3. Se inicializa cada nodo con sus datos de entrenamiento. Estos datos de entrenamiento, como ya dijimos antes, serán una matriz, con tantas columnas como clases haya en el modelo y con tantas filas como vectores de entrenamiento. Es decir, si el modelo tiene K clases y V vectores de entrenamiento, entonces la matriz que le llega a cada nodo será de la forma

$$T = \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1K} \\ t_{21} & t_{22} & \dots & t_{2K} \\ t_{31} & t_{32} & \dots & t_{3K} \\ \vdots & \vdots & & \vdots \\ t_{V1} & t_{V2} & \dots & t_{VK} \end{pmatrix}$$

4. Cada nodo recibe sus datos de entrenamiento y calcula el histograma de 100 niveles para cada columna del vector (cada columna contiene los valores de la característica para una clase determinada). Es decir, cada nodo tiene K histogramas, uno por cada clase.

5. En este momento, ya los nodos hijos están inicializados con K histogramas de 100 niveles equidistantes, es decir, hemos hecho una cuantización lineal. A continuación, aplicaremos a estos nodos una segunda inicialización que consiste en una cuantización no lineal sobre los histogramas, reduciendo los niveles de dicho histograma de 100 a D (para este proyecto $D = 8$). Para ello, iremos creando particiones del histograma según la entropía de los mismos. La entropía mide la variabilidad de la información en un determinado conjunto de datos. Dicho de otra forma, la entropía mide la cantidad de información que aporta un determinado conjunto de datos, ya que si unos datos no varían, realmente no aportan información valiosa, puesto que ya la conocemos. Por ejemplo, si un periódico publicara siempre la misma información, ya no nos estaría informando de nada, puesto que ya conoceremos esa información sólo con haber leído el periódico el día anterior. En [5] se explica más detenidamente este método de cuantización, el cual es muy interesante, puesto que distribuye los datos de manera que podamos saber más sobre ellos.

   En la Ecuación (5.1) podemos ver la fórmula aplicada, donde queremos obtener el punto de partición P para el que maximicemos la entropía. K es el número de clases del modelo y x_k es un índice aleatorio dentro del histograma de la clase k.

Para obtener el punto de mayor entropía, definimos un índice x con el que vamos recorriendo el histograma y calculamos la entropía para ambos subconjuntos y los sumamos. El punto en el que maximicemos esta entropía conjunta será el que aporte mayor información.

$$P = max_x \left\{ \sum_{k=1}^{K} p(x_k \leq x) \log \left(\frac{1}{p(x_k \leq x)} \right) + p(x_k > x) \log \left(\frac{1}{p(x_k > x)} \right) \right\} \quad (5.1)$$

El proceso de partición del histograma se lleva a cabo de la siguiente manera:

a) Normalizamos el histograma. Haciendo esto, pasamos a trabajar con las frecuencias relativas.

b) Empezamos la generación de particiones con el histograma completo, es decir, con los 100 niveles.

c) Recorremos el histograma buscando la partición que genere una mayor entropía. Para ello, suponiendo que el histograma empieza en l_0 y acaba en l_f, vamos recorriendo el histograma con un índice l. Para cada l calculamos la entropía considerando las particiones $[l_0, l]$ y $[l+1, l_f]$ como en (5.1). Repetimos el proceso hasta que encontremos la entropía máxima. Este será el índice por donde partamos el histograma.

d) Partimos cada histograma generado en el paso anterior con el mismo método y así sucesivamente hasta que tengamos D niveles.

En nuestro caso, queremos 8 niveles, por lo que hay que hacer 8 particiones. Para ello, primero partimos el histograma total, quedándonos 2 particiones. Posteriormente partimos de nuevo cada una de las particiones de nuevo, obteniendo 4 particiones y por último, partimos cada una de ellas de nuevo para obtener las 8 particiones deseadas.

Al hacer estas particiones, hay que tener cuidado con controlar que las particiones iniciales estén demasiado cerca de los límites. Por ejemplo, si la primera partición del histograma de 100 niveles se hace en el índice 1 del histograma, entonces la siguiente iteración intentará hacer una partición del intervalo $[0, 1]$ y no podrá particionarlo, generando resultados inesperados. Para evitar esto, controlaremos que si la partición se va a hacer en un punto demasiado cercano a uno de los límites,

partamos por la mitad del histograma. Esto quita eficiencia al método, pero los resultados siempre serán mejores que si no hacemos esta comprobación.

6. Así pues, con este proceso hemos creado un histograma irregular, es decir, donde sus límites toman valores no regulares. Ya tenemos creada la tabla de probabilidades descrita en la Figura 5.2. Cada una de estas probabilidades P_D^K representa la probabilidad de que el valor de la característica esté comprendido comprendido entre d_i y d_{i-1}, donde d_i son los valores por los que se ha partido el histograma.

7. Una vez tenemos todos los nodos `LeafNode` creados e inicializados, queremos incluir algún mecanismo que nos permita decidir qué nodos son más importantes que otros, es decir, qué características tienen más relevancia en el modelo que otras.

   Para conseguir esta relación de importancias, mediremos cómo de diferentes son los histogramas de cada nodo. Cuanto más diferentes sean estos histogramas unos de otros más información aportarán a la hora de clasificar. Por ejemplo, supongamos que tenemos un clasificador de 2 clases con 10 características. En las 9 primeras características, los valores obtenidos en la fase de entrenamiento eran muy parecidos, por lo que los histogramas del nodo que representa esa característica son muy parecidos. En cambio, para la décima característica los valores eran muy diferentes, por lo que se distinguen muy bien las clases a partir de esta característica. A la hora de clasificar, nos interesa que esa décima característica tenga más peso, porque ayuda a diferencias mejor las clases.

   Para medir cómo de diferente es un histograma de otro, utilizaremos la divergencia de *Kullback-Leibler*, también conocida como distancia de *Kullback-Leibler*. En (5.2) podemos ver la definición de dicha distancia.

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{5.2}$$

   donde D y Q son dos distribuciones de probabilidad (histogramas).

   Esta divergencia realmente no es una distancia, puesto que no es simétrica. Es por esto que lo que haremos en nuestro clasificador sea calcular la media de las distancias en ambas direcciones entre los K histogramas de cada nodo.

   Así pues, el último paso de la fase de entrenamiento consiste en crear una tabla con las distancias asociadas a cada nodo, para posteriormente, en la fase de

clasificación, ponderar el resultado de cada nodo por esta distancia.

Así pues, ya tenemos nuestro clasificador *Naive Bayes* preparado para clasificar cualquier vector de características. Sobra decir que los vectores de características que deseemos clasificar deben tener la misma dimensión que los vectores de características con los que hemos entrenado el clasificador.

5.7.2. Clasificación de características

Esta segunda fase del clasificador es asíncrona. En cualquier momento se le puede pedir al clasificador que clasifique un determinado vector de características. El tiempo en resolver esta clasificación será lineal con el número de clases existentes en el modelo.

Para resolver la clasificación de unas características, utilizaremos una variación de la relación que definíamos en la Sección 4.7.3 acerca del clasificador *Naive Bayes*. Esta variación se debe a la ponderación de las características que explicábamos anteriormente. Así pues, nuestro clasificador será como en (5.3).

$$c(f_1, ..., f_n) = argmax_c \left\{ p(C = c) \prod_{i=1}^{n} (p(F_i = f_i | C = c)^{w_i}) \right\} \tag{5.3}$$

Observando esta fórmula, podemos aclarar algunos términos. La probabilidad $p(C = c)$ es la probabilidad *a priori* para cada una de las clases, es decir, es uno de los parámetros que se le pasan a la red bayesiana para que se inicialice. Por otra parte, el término $p(F_i = f_i | C = c)$ es la probabilidad de que la característica F_i tome un determinado valor f_i, suponiendo que la clase es c. Además, para cada característica, vemos un peso de ponderación w_i, el cual se corresponde con la distancia normalizada que habíamos calculado en la fase de entrenamiento para cada una de las características.

Ahora, pasamos a escalas logarítmicas por simplicidad, ya que así trabajamos con sumas en vez de productos, quedando la función de clasificación como en (5.4).

$$c(f_1, ..., f_n) = argmax_c \left\{ \log(p(C = c)) + \sum_{i=1}^{n} [w_i \log(p(F_i = f_i | C = c))] \right\} \tag{5.4}$$

Resumiendo, debemos encontrar la clase c que maxifica esa relación. Para ello, debemos sumar la relación $p(F_i = f_i | C = c)$ de todas las características para cada una de las clases, multiplicar por su correspondiente probabilidad *a priori* (p(C=c)) y quedarnos con el valor mayor.

La parte de clasificación es bastante simple y consta de los siguientes pasos:

1. Inicializamos un vector de resultados $R = (r_1, \ldots, r_K)$ de K elementos, uno por cada clase a su respectiva probabilidad *a priori*, habiendo sido éstas pasadas a escalas logarítmicas. Este vector es el que contendrá las probabilidades finales asignadas por el clasificador para cada una de las clases.

2. Posteriormente, vamos recorriendo los diferentes nodos `LeafNode` del modelo (uno por cada característica) y pidiendo sus probabilidades asociadas a cada clase para un valor de característica dado.

3. Cada nodo hijo recorre sus histogramas (tiene uno por cada clase) mirando qué probabilidad corresponde a ese determinado valor de característica. Suponiendo que al nodo hijo le preguntan por un determinado valor x, dicho nodo buscará en cada histograma y devolverá unas probabilidades $P_{d_i}^k$ tal que $d_i \geq x > d_{i-1}$, donde d_i y d_{i-1} son los límites superior e inferior respectivamente, de las particiones obtenidas en la fase de entrenamiento (los límites del histograma) y k es cada una de las clases del modelo.

4. Se comprueba que ninguna de estas probabilidades devuelta por los nodos hijos sean cero, ya que esto provocaría un valor de $-\infty$ al estar trabajando con valores logarítmicos. Si se encuentra una probabilidad cero, se aproxima a 0.01 (-2 en valores logarítmicos).

5. Se va almacenando el sumatorio de todas estas probabilidades en sus correspondiente índice del vector R.

6. Una vez se han recorrido todos los nodos hijos y se han ido sumando sus respectivas probabilidades para cada una de las clases, la clase asignada por el clasificador será aquella que tenga un valor mayor en el vector R.

5.8. El módulo de Gestión

En esta sección explicaremos detalladamente el funcionamiento de la nueva versión del programa `xmipp_micrograph_mark`, el cual ha sido modificado para crear el clasificador y entrenarlo, y posteriormente clasificar vectores de características.

Nos centraremos principalmente en la gestión de la red bayesiana, ya que ésta ha sido explicada anteriormente en la Sección 5.7.

5.8.1. Preprocesamiento de las partículas

Antes de empezar con los detalles del desarrollo, conviene explicar cómo se obtienen las características de un determinado objecto `Particle`.

En `xmipp_micrograph_mark`, cada objeto `Particle` está dentro de una **pieza**. Una pieza es un trozo de micrografía en el que está contenido nuestro objeto, es decir, es la manera que tiene el programa de dividir la micrografía en objetos más pequeños. Esto se ha pensado así, porque para las técnicas de preprocesamiento sobre las subimágenes siempre es mejor coger trozos de micrografía más grandes que una simple partícula, para hacer un filtrado y un *denoising* más efectivo.

Cuando queremos calcular las características de un determinado objeto en una determinada pieza, se realizan los siguientes pasos:

1. Se hace un filtrado paso alto de la pieza mediante un filtro de tipo coseno alzado, los cuales se caracterizan por reducir al mínimo la interferencia entre símbolos (*aliasing*). El valor de corte lo puede definir el usuario en el diálogo de configuración.

2. Se aplica un *denoising* de tipo bayesiano a la pieza para quitar ruido y suavizar el fondo de la micrografía. Ver [28] para más información acerca de este paso intermedio.

3. Se hace un filtrado de los datos anómalos, descartando aquellos que sean demasiado diferentes de la media. Esto sirve para purificar la imagen.

4. Se ecualiza el histograma de la pieza.

5. Se vuelven a descartar los datos anómalos para la imagen ecualizada.

6. Ya por último se obtienen las características del objecto `Particle` contenido en la pieza.

5.8.2. Escaneado de las micrografías

Regiones de escaneo

El escaneado de las micrografías es una de las partes más importantes de este proyecto, ya que es fundamental escanear la micrografía con la suficiente precisión como

para no saltarnos ninguna posible partícula. Para conseguir esto, tendremos que definir unas determinadas regiones de escaneo en las que el programa buscará la posible presencia de partículas.

- Por una parte, tenemos el conepto de **pieza**, el cual ya explicamos en la sección anterior. Básicamente es una región de la micrografía varias veces más grande que una partícula (el tamaño de las piezas es definido por el usuario), sobre la cual se aplican una serie de técnicas de procesamiento de imágenes eliminar ruido y mejorar la calidad de las mismas y facilitar el proceso de clasificación. A medida que nos vamos desplazando por la micrografía, tendremos que ir desplazando la pieza.

- Además, también tenemos una región más pequeña llamada **máscara**. En el capítulo 3 ya hablamos brevemente sobre ella y vimos su aspecto (Figura 3.5(a)). Este trozo de micrografía es en el que debe encajar las partículas. Es decir, a medida que vamos moviendo la máscara por la micrografía, vamos calculando las características de la subimagen que está contenida en la máscara. Puesto que el tamaño de la máscara debe estar ajustada al de las partículas, es muy importante que la partícula esté lo más centrada posible dentro de la máscara para que el clasificador reconozca las partículas correctamente.

Desplazamiento de las regiones por la micrografía

Debido al hecho de que las partículas deben estar muy centradas dentro de la máscara, es necesario recorrer la micrografía de pocos en pocos píxeles. Para esto se han definido unos valores de solapamiento tanto para el desplazamiento de las piezas como el de las máscaras. El proceso de escaneo es como sigue:

1. Situamos la pieza y la máscara en la esquina superior izquierda coincidiendo con la esquina superior izquierda de la micrografía.

2. Dentro de esa pieza, vamos desplazando la máscara con un solapamiento tanto vertical como horizontal de S_m píxeles. Así pues, en una pieza de tamaño T_p (son cuadradas, con lo que T_p representa tanto el alto como el ancho) y con una máscara de tamaño T_m (a pesar de que la máscara sea circular, se define su tamaño como un cuadrado de alto y ancho T_m) habrá $[T_p/(T_m - S_m)]^2$ desplazamientos. En la práctica lo que haremos será definir un porcentaje de solapamiento entre partículas

en función del radio de la partícula definido por el usuario, por ejemplo, si tenemos un radio de partícula de 128 píxeles y definimos un porcentaje de solapamiento entre partículas del 50 %, entonces tendremos un solapamiento S_m de 64 píxeles.

3. Una vez ya hemos escaneado la pieza completa, desplazamos la pieza hasta la siguiente posición de escaneo. A este desplazamiento también se le aplica un porcentaje de solapamiento S_p, tanto para desplazamientos verticales como horizontales, en función del radio de la partícula. Este porcentaje lo difinimos también en función del radio de las partículas de la misma manera que en el paso anterior con las máscaras. Una vez posicionados sobre la siguiente pieza de escaneo, volvemos a escanear con la máscara de la forma descrita en el punto anterior. Así pues, vamos moviendo la pieza por la micrografía y escaneando cada pieza con la máscara hasta que lleguemos a la esquina inferior derecha de la micrografía.

En la Figura 5.3 podemos ver una representación gráfica de cómo sería el proceso de escaneo por la micrografía.

5.8.3. Almacenamiento del modelo

Puesto que para la clasificación de una determinada partícula debemos entrenar una red bayesiana, primero debemos generar esos vectores de entrenamiento. La generación de estos vectores de entrenamiento se explica en detalle en la Sección 5.8.5.

Una vez hemos aprendido una micrografía, debemos guardar los datos aprendidos para que puedan ser cargados por la próxima micrografía. Para ello, existe la opción de menú *AutoSelection→Save Model*, la cual guarda los datos del modelo actual.

Dichos datos tienen dos partes:

- Una primera parte contiene los datos de configuración. Entre estos datos se encuentran algunos valores definidos por el usuario, como por ejemplo el radio de las partículas, el tamaño de la máscara, etc. Todos estos datos de configuración se guardan en un fichero de nombre *NombreModelo.param*.

- El segundo conjunto de datos son los vectores de entrenamiento. En estos vectores tendremos los datos para las tres clases del modelo: Partícula, No-Partícula y Error. Estos datos se guardan en un fichero de nombre *NombreModelo.training*. Además, al comienzo de este fichero tenemos una serie de datos relacionados con

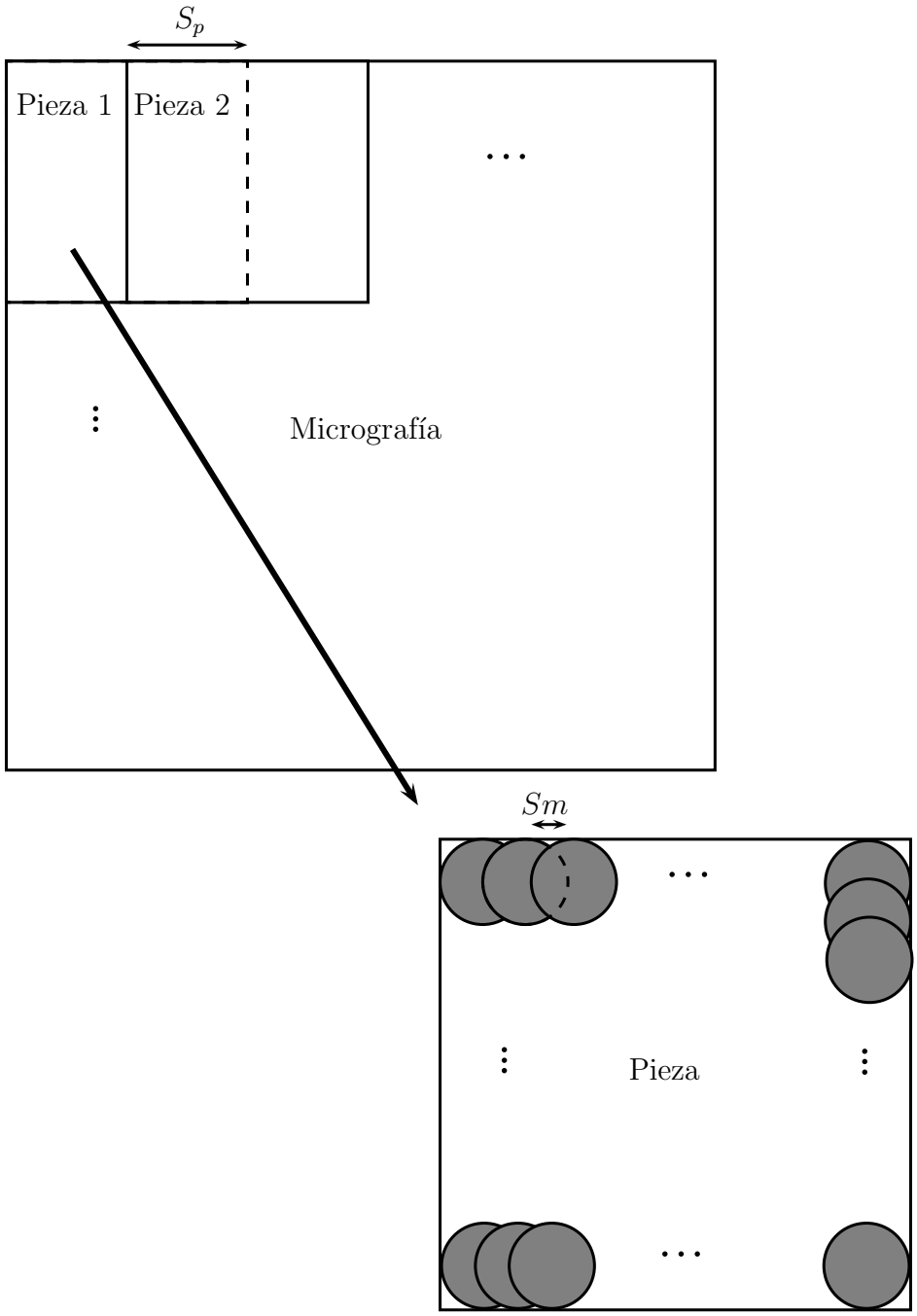


Figura 5.3: Representación del escaneo de una micrografía. En la subimagen superior podemos ver cómo se desplaza la pieza por la micrografía, con un solapamiento S_p tanto horizontal como vertical. Por otra parte, en la subimagen inferior aparece representado el desplazamiento de la máscara por cada pieza, con un solapamiento horizontal y vertical S_m

las micrografías que ya tenemos aprendidas. En resumen, ese fichero contiene los siguientes datos:

1. El número de micrografías que hemos aprendido
2. El área de cada una de las micrografías que hemos aprendido. Este dato lo utilizamos para calcular las probabilidades *a priori* de cada una de las clases.
3. El número de partículas que hemos marcado para cada una de las micrografías.
4. El número de errores que cometió el clasificador en cada una de las micrografías anteriores.
5. La longitud del vector de características.
6. El número de clases del modelo
7. Para cada clase, un número que indica cuántos objetos `Particle` contiene dicha clase, seguido de dichos objetos.

En la Figura 5.4 podemos ver el diálogo que aparece en el programa cuando hacemos *click* sobre la opción de menú para guardar el modelo.

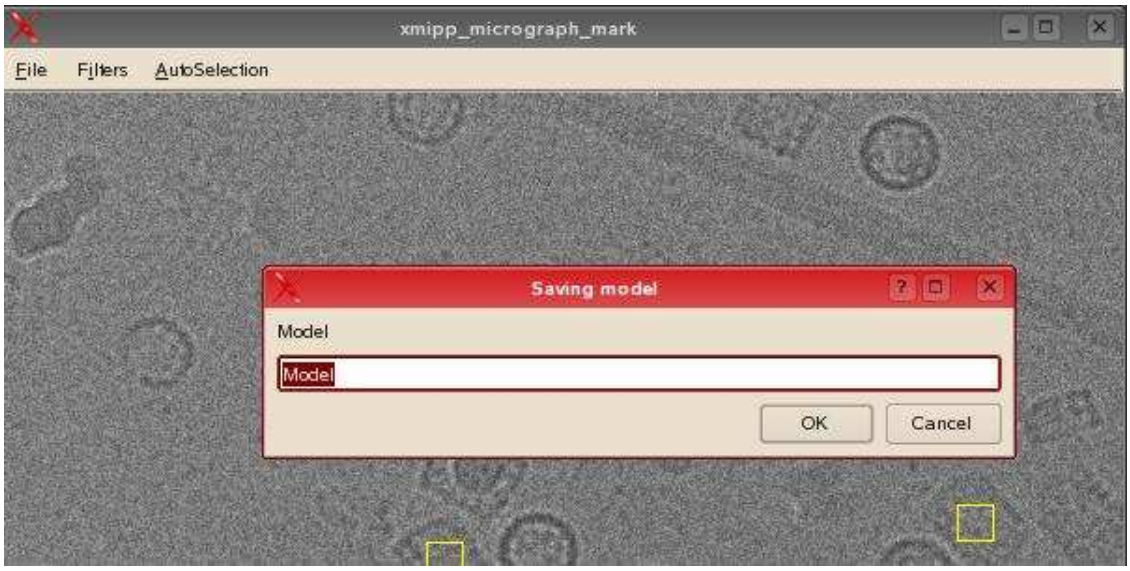


Figura 5.4: Diálogo emergente para guardar un modelo

5.8.4. Carga de un modelo previo

La carga del modelo es el paso inverso al descrito en la versión anterior. Debemos tener en cuenta con qué nombre habíamos guardado el modelo, ya que como podemos ver en la Figura 5.5, debemos introducir el nombre del modelo que queremos cargar (sin incluir la extensión). Esto se ha hecho así pensando en la posibilidad de querer tener varios modelos para diferentes tipos de partículas o de orientaciones de un mismo tipo de partícula.

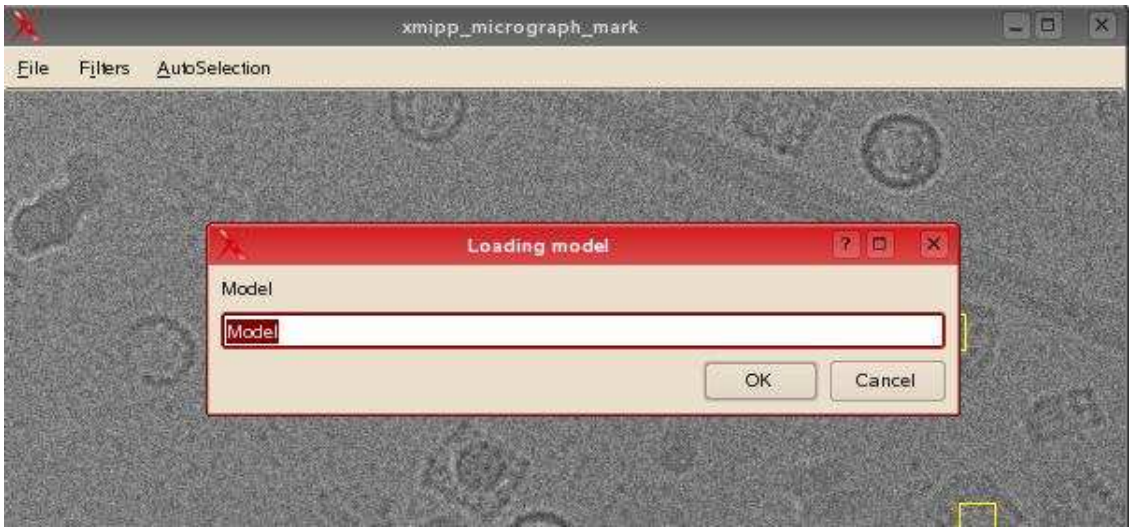


Figura 5.5: Diálogo emergente para cargar un modelo

5.8.5. La fase de entrenamiento

La fase de entrenamiento o de aprendizaje es aquella en la que debemos recolectar un conjunto de datos que le sirvan al modelo para distinguir las diferentes clases existentes. Este es posiblemente el proceso más importante de todos, o al menos el más complicado, ya que es realmente difícil encontrar un conjunto de datos realmente representativos de las clases consideradas dentro del modelo.

En el Capítulo 3 ya expusimos cuáles iban a ser los datos que van a ser utilizados como entrenamiento. Una pregunta clave en este proceso es ¿cuántos datos son necesarios para empezar a clasificar automáticamente?. La respuesta a esta pregunta es muy variable dependiendo del modelo, pero existe una regla que dice que el número de vectores de entrenamiento necesarios para una clasificación óptima es de alrededor

10 veces el número de características existentes en el modelo. Es decir, que si se tienen 142 características, como es nuestro caso, en teoría se necesitarían 1420 vectores de entrenamiento para empezar a clasificar.

En nuestro modelo, la fase de entrenamiento y la de clasificación están ligeramente entremezcladas, puesto que el modelo sigue aprendiendo de las clasificaciones que se van realizando, es decir, cada vez que clasificamos una determinada micrografía, la corregimos y aprendemos los datos corregidos. En las siguientes secciones explicaremos en más detalle cómo aprendemos de estas correcciones.

En este proceso, llegará un momento en que el modelo ya se sature y ya no sea necesario seguir aprendiendo datos, puesto que no aportan mejoras sustanciales.

La fase de entrenamiento tiene dos partes. Una primera parte de obtención de datos que se produce cuando el usuario selecciona la opción de menú *AutoSelection→Learn particles* y una segunda parte que empieza en el momento en que el usuario quiere clasificar partículas (*AutoSelection→AutoSelect*). Para que se entienda mejor, explicaremos cada parte por separado.

- Esta primera fase consta de los siguientes pasos secuenciales para la recolección de datos, suponiendo que ya hemos marcado las partículas y rechazado los errores en el caso de que hayamos buscado las partículas automáticamente:

    1. Guardamos en un vector los objetos `Particle` de las partículas marcadas manualmente. Estos objetos todavía no tienen asociado un vector de características. Simplemente contienen sus coordenadas dentro de la micrografía y el índice que indica en qué orden fueron marcadas.
    2. Calculamos las características asociadas a estas partículas como se ha descrito en la Sección 3.4.3 y las guardamos en sus correspondientes objetos `Particle`.
    3. En caso de que hubiera errores en la micrografía (porque hubieramos intentado buscar partículas automáticamente), ya tendríamos calculadas sus características, ya que las hemos utilizado previamente para clasificarlas como partículas, con lo cual simplemente guardamos las características en el vector correspondiente del objeto `Classification_model`.
    4. Escaneamos la micrografía como hemos explicado en 5.8.2, calculando las características de todo lo que no sea partícula, es decir, recorremos la micrografía saltándonos aquellas coordenadas que puedan contener una partícula,

y guardamos estos vectores de características en el correspondiente objeto `Particle`, el cual a su vez guardaremos en el vector de la clase No-Partícula (C=1) del objeto `Classification_model`. Hay que resaltar que en este proceso se establece un solapamiento entre máscaras en el escaneo del 50 % del radio de la partícula definido por el usuario (ver 5.8.2 para más información) para aprender este tipo de elementos más rápidamente. No hay solapamiento entre piezas, ya que no aportaría mucha más información y ralentizaría demasiado el proceso.

5. Una vez ya hemos aprendido todos los datos, los guardamos con la opción de menú *AutoSelection→Save models*.

- Por otro lado, hay una segunda parte que, a pesar de pertenecer a la fase de entrenamiento, se produce en el momento de empezar a clasificar partículas. En la fase anterior simplemente se obtenían los datos, pero realmente no se hacía nada con ellos. Se aprendían (se guardaban en determinados vectores) pero no se utilizaban para entrenar. Es en esta parte donde se efectúa este entrenamiento sobre la red bayesiana. Esto se hace para mejorar el rendimiento del programa, puesto que no necesitamos entrenar la red bayesiana a menos que vayamos a buscar partículas automáticamente. Por ejemplo, se podría dar el caso de que el usuario simplemente quisiera aprender de una determinada micrografía las características de las partículas y del fondo (sin clasificar la micrografía) y guardarlos. En este caso, no nos interesa calcular nada de la red bayesiana porque no la vamos a utilizar. Es por esto, que la red bayesiana se entrena en el momento de empezar a clasificar una micrografía. Este proceso de entrenamiento consta de los siguientes pasos:

  1. A partir de los datos guardados de otras micrografías, calculamos las probabilidades *a priori* de las clases del modelo (Partícula, Error y No-Partícula). Estas probabilidades se calculan mediante el área de las partículas y las micrografías. Puesto que conocemos el radio de las partículas (definido por el usuario en el diálogo de configuración), podemos calcular el área de cada una de ellas como $A = \pi R^2$. También conocemos las diferentes áreas de las micrografías, así como las partículas marcadas en cada una de ellas. A partir de estos datos, podemos obtener la probabilidad *a priori* de las partículas como en (5.5), donde A_p es el área total de las partículas aprendidas y A_m el área total de las micrografías aprendidas.

$$p(C = 0) = \frac{A_p}{A_m} \tag{5.5}$$

Podemos obtener la probabilidad *a priori* de los errores exactamente de la misma manera y por último, la de la clase No-Partícula sería $p(C = 1) = 1 - (p(C = 0) + p(C = 2))$. (En el programa se han definido las clases como C=0 para la clase Partícula, C=1 para No-Partícula y C=2 como Error).

2. Una vez obtenidas las probabilidades para cada una de las clases del modelo, extraemos las características del objeto `Classification_model`, el cual como explicamos antes, contiene las características de las clases.
3. Inicializamos la red bayesiana con las características y las probabilidades de la forma que se ha descrito en 5.7.1.
4. Ahora ya hemos completado la fase de entrenamiento. La red bayesiana ya tiene todas sus tablas de probabilidades creadas y puede clasificar cualquier vector de características que se le pase.

5.8.6. La fase de clasificación

Como habíamos dicho anteriormente, las fases de entrenamiento y de clasificación están ligeramente entremezcladas, puesto que al empezar esta última, lo primero que hacemos es inicializar la red bayesiana con las características que teníamos almacenadas.

Para poder entrar en esta fase de clasificación, es necesario haber aprendido datos de otras micrografías previas, ya que nos hacen falta características y probabilidades *a priori* para entrenar el clasificador *Naive Bayes*. En caso de que el usuario intente clasificar sin haber cargado un modelo previamente, el programa avisará al usuario de que necesita cargar un modelo.

Esta fase tiene una gran similitud con la parte de la fase de entrenamiento en la cual vamos recorriendo la micrografía aprendiendo elementos de tipo No-partícula (todo aquello que no es ni Partícula ni Error), aunque con algunas diferencias destacables.

1. Inicializamos la red bayesiana tal como se explica en la Sección 5.8.5.
2. Vamos recorriendo la micrografía como se describe en la Sección 5.8.2 con un solapamiento entre máscaras y piezas del 90 % y del 50 % del radio de la partícula respectivamente. El solapamiento entre máscaras es tan grande debido a la precisión necesaria en lo que al centro de la parte escaneada a la hora de calcular las

características se refiere. Este centro ha de ser tan cercano al de la partícula (en el caso de que nos encontremos con una partícula) como sea posible, ya que si no al clasificador le resultara muy complicado, si no imposible, clasificar correctamente una partícula. El solapamiento entre piezas ha sido escogido teniendo en cuenta el peor caso, que es aquel en el borde de la pieza "parte"por la mitad una partícula. Con un solapamiento entre piezas del 50 % del radio de la partícula (definido por el usuario) conseguimos que en la siguiente pieza de escaneo, esa partícula entre entera en la pieza sin ser partida, de manera que al recorrer la pieza por partes podamos evaluar una posición lo suficientemente cercana al centro de la partícula como para que el clasificador pueda tener posibilidades de clasificarla correctamente.

3. Mientras vamos recorriendo la micrografía y calculando las características de los trozos de micrografía observados, preguntamos al clasificador *Naive Bayes* a qué clase pertenecen éstas. Puesto que en todo momento hemos asumido que la clase Partícula es la clase $C = 0$, si el clasificador responde que las características pertenecen a la clase $C = 0$, entonces significa que ha encontrado una partícula.

4. En el caso de que encontremos una partícula, creamos el objeto `Particle` a partir de sus características y sus coordenadas en la micrografía.

5. Añadimos las partículas encontradas a un vector de candidatos.

6. Una vez hemos terminado de recorrer toda la micrografía, tendremos un vector de candidatos bastante elevado, ya que probablemente hemos marcado varias veces las mismas partículas debido al solapamiento aplicado al recorrer la micrografía. Para descartar estas repeticiones aplicamos un filtro, el cual mira si dos partículas están más cerca de R, siendo R el radio de la partícula definido por el usuario, se queda con la partícula que esté antes en la lista de candidatos.

7. Almacenamos ese vector de candidatos con los candidatos finales en un modelo temporal. Estos datos serán almacenados en el modelo de aprendizaje (se añaden a los datos de entrenamiento) en caso de que el usuario así lo seleccione mediante la opción de menú *AutoSelection→Learn Particles*.

8. El usuario debe corregir las partículas marcadas en caso de que existan errores. Para borrar una partícula que ha sido marcada por el modelo, debemos pulsar

sobre la marca en la subimagen del programa con el botón derecho del ratón. En ese momento nos aparecerá un diálogo que nos pregunta si queremos borrar la marca, o simplemente moverla. Si la borramos, esa región de la micrografía pasará a ser un Error en el modelo, mientras que si la movemos, simplemente cambiará sus coordenadas. Además, el usuario también puede marcar las partículas que el modelo haya pasado por alto.

Una vez ya tengamos marcadas las partículas que nosotros consideremos como acertadas, podemos guardarlas para que el modelo aprenda de estos nuevos datos. Para ello, se utiliza la opción de menú *AutoSelecion→Save Models*.

Con este último paso, hemos aprendido los datos de esta clasificación. Así pues, hemos conseguido implementar un modelo bastante dinámico, ya que aprende a medida que se utiliza. Sobra decir que es posible clasificar, guardar las coordenadas de las partículas y no aprender los nuevos datos, es decir, podemos simplemente clasificar una micrografía a partir de un conjunto de datos previos.

Capítulo 6

Resultados obtenidos

6.1. Suposiciones

Antes de exponer los resultados, debemos aclarar algunas suposiciones hechas a la hora de marcar partículas, considerar errores, etc. Se ha considerado como partícula válida cualquier partícula con orientación lateral (partículas rectangulares) con la suficiente resolución como para distinguir los bordes (si está demasiado difusa no se considera como partícula válida). Además, deben estar lo suficientemente alejadas de los límites laterales de la micrografía, aproximadamente unos 10 píxeles de diferencia entre el borde de la partícula y los ejes. Las partículas con orientaciones verticales son consideradas falsos positivos, al igual que partículas superpuestas, partículas demasiado próximas unas de otras y partículas con elementos no deseados superpuestos y partículas incompletas (si están demasiado cerca de los ejes, por ejemplo).

6.2. Experimentos

Para exponer los resultados obtenidos, mostramos en primer lugar los resultados del modelo definitivo. Hemos medido la relación de falsos positivos (FPR), la tasa de aciertos (TPR), la tasa de falsos negativos, es decir, de partículas ignoradas (FNR). Se han hecho dos medidas diferentes para el modelo estudiado:

- Una calculando estos parámetros para cada micrografía y estudiando la mejora de una a otra.

- Y otra calculando los parámetros sobre una micrografía de referencia conforme vamos aprendiendo el resto de micrografías de la lista con el modelo.

6.2.1. Micrografía a micrografía

En la Tabla 6.1 podemos ver los resultados obtenidos micrografía a micrografía, donde P/E es el número de partículas/errores aprendidos por el modelo, M es el número de partículas marcadas por el modelo de forma automática, F es el número de errores que hemos cometido, C el número de aciertos, I el número de partículas ignoradas y T el número real de partículas existente en la micrografía. Además, también mostramos la tasa de falsos positivos (FPR), la de falsos negativos (FNR) y la de aciertos (TPR). Los pasos seguidos para obtener los resultados son:

1. Abrimos la primera micrografía, marcamos las partículas existentes y aprendemos los resultados con el modelo.

2. Abrimos la siguiente micrografía, buscamos las partículas presentes con el modelo automático

3. Corregimos los resultados, borrando los errores (para que el modelo los aprenda como errores) y marcando las partículas que faltan.

4. Aprendemos los datos corregidos con el modelo.

5. Repetimos los pasos de 2 a 4 hasta llegar a las 80 micrografías.

#	P/E	M	F	C	I	T	FPR	FNR	TPR
1	17/0	3	2	1	12	13	66,67 %	92,31 %	7,69 %
2	30/2	13	8	5	9	14	61,54 %	64,29 %	35,71 %
3	44/10	35	25	10	1	11	71,43 %	9,09 %	90,91 %
4	55/35	30	17	13	4	16	56,67 %	23,53 %	76,47 %
5	71/52	36	25	11	2	13	69,44 %	15,38 %	84,62 %
6	84/77	18	9	9	0	9	50,00 %	0,00 %	100,00 %
7	93/86	35	17	18	1	19	48,57 %	5,26 %	94,74 %
8	112/103	41	21	20	1	21	51,22 %	4,76 %	95,24 %
9	133/124	23	12	11	2	13	52,17 %	15,38 %	84,62 %
10	146/136	42	27	15	4	19	64,29 %	21,05 %	78,95 %

11	165/163	28	8	20	1	21	28,57 %	4,76 %	95,24 %
12	186/171	36	19	17	0	17	52,78 %	0,00 %	100,00 %
13	203/190	9	0	9	0	9	0,00 %	0,00 %	100,00 %
14	212/190	18	8	10	1	11	44,44 %	9,09 %	90,91 %
15	223/198	14	8	6	0	6	57,14 %	0,00 %	100,00 %
16	229/206	9	7	2	0	2	77,78 %	0,00 %	100,00 %
17	231/213	14	4	10	1	11	28,57 %	9,09 %	90,91 %
18	242/217	16	7	9	0	9	43,75 %	0,00 %	100,00 %
19	251/224	9	1	8	1	9	11,11 %	11,11 %	88,89 %
20	260/225	30	15	15	1	16	50,00 %	6,25 %	93,75 %
21	276/240	15	7	8	0	8	46,67 %	0,00 %	100,00 %
22	284/247	8	6	2	1	3	75,00 %	33,33 %	66,67 %
23	287/253	11	2	9	1	10	18,18 %	10,00 %	90,00 %
24	297/255	21	11	10	1	11	52,38 %	9,09 %	90,91 %
25	308/266	29	7	22	0	22	24,14 %	0,00 %	100,00 %
26	330/273	19	9	10	1	11	47,37 %	9,09 %	90,91 %
27	341/282	20	11	9	0	9	55,00 %	0,00 %	100,00 %
28	350/293	18	5	13	0	13	27,78 %	0,00 %	100,00 %
29	363/298	16	7	9	1	10	43,75 %	10,00 %	90,00 %
30	373/305	16	4	12	0	12	25,00 %	0,00 %	100,00 %
31	385/309	19	9	10	0	10	47,37 %	0,00 %	100,00 %
32	395/318	14	6	8	1	9	42,86 %	11,11 %	88,89 %
33	404/324	24	7	17	1	18	29,17 %	5,56 %	94,44 %
34	422/331	14	8	6	0	6	57,14 %	0,00 %	100,00 %
35	428/339	15	2	13	0	13	13,33 %	0,00 %	100,00 %
36	441/341	22	8	14	1	15	36,36 %	6,67 %	93,33 %
37	456/349	17	5	12	4	16	29,41 %	25,00 %	75,00 %
38	472/354	29	13	16	2	18	44,83 %	11,11 %	88,89 %
39	490/367	16	8	8	0	8	50,00 %	0,00 %	100,00 %
40	498/375	17	7	10	2	12	41,18 %	16,67 %	83,33 %
41	510/382	24	7	17	1	18	29,17 %	5,56 %	94,44 %
42	528/389	27	11	16	1	16	40,74 %	5,88 %	94,12 %
43	544/400	26	9	17	0	17	34,62 %	0,00 %	100,00 %

44	561/409	26	10	16	1	17	38,46 %	5,88 %	94,12 %
45	578/419	28	6	22	0	22	21,43 %	0,00 %	100,00 %
46	600/425	19	10	9	9	18	52,63 %	50,00 %	50,00 %
47	618/435	21	6	15	0	15	28,57 %	0,00 %	100,00 %
48	633/442	34	15	19	2	21	44,12 %	9,52 %	90,48 %
49	654/457	38	15	23	0	23	39,47 %	0,00 %	100,00 %
50	677/472	22	9	13	1	14	40,91 %	7,14 %	92,86 %
51	691/481	19	6	13	1	14	31,58 %	7,14 %	92,86 %
52	705/487	27	11	16	1	17	40,74 %	5,88 %	94,12 %
53	722/498	28	11	17	1	18	39,29 %	5,56 %	94,44 %
54	740/509	19	8	11	0	11	42,11 %	0,00 %	100,00 %
55	751/517	29	16	13	4	17	55,17 %	23,53 %	76,47 %
56	768/533	22	13	9	3	12	59,09 %	25,00 %	75,00 %
57	780/546	23	6	17	1	18	26,09 %	5,56 %	94,44 %
58	798/552	30	11	19	0	19	36,67 %	0,00 %	100,00 %
59	817/563	10	4	6	1	7	40,00 %	14,29 %	85,71 %
60	824/567	2	1	1	0	1	50,00 %	0,00 %	100,00 %
61	825/568	37	20	17	1	18	54,05 %	5,56 %	94,44 %
62	843/588	10	6	4	0	4	60,00 %	0,00 %	100,00 %
63	847/594	26	11	15	0	15	42,31 %	0,00 %	100,00 %
64	862/605	23	11	12	4	16	47,83 %	25,00 %	75,00 %
65	878/616	30	11	19	3	22	36,67 %	13,64 %	86,36 %
66	900/628	28	11	17	2	19	39,29 %	10,53 %	89,47 %
67	919/639	20	11	9	1	10	55,00 %	10,00 %	90,00 %
68	929/650	17	2	15	0	15	11,76 %	0,00 %	100,00 %
69	944/652	26	9	17	1	18	34,62 %	5,56 %	94,44 %
70	962/661	15	6	9	0	9	40,00 %	0,00 %	100,00 %
71	971/667	23	9	14	0	14	39,13 %	0,00 %	100,00 %
72	985/676	19	6	13	0	13	31,58 %	0,00 %	100,00 %
73	998/682	15	7	8	3	11	46,67 %	27,27 %	72,73 %
74	1009/689	13	2	11	0	11	15,38 %	0,00 %	100,00 %
75	1020/691	14	4	10	0	10	28,57 %	0,00 %	100,00 %
76	1030/695	25	9	16	1	17	36,00 %	5,88 %	94,12 %

77	1047/704	18	9	9	0	9	50,00 %	0,00 %	100,00 %
78	1056/713	12	6	6	0	6	50,00 %	0,00 %	100,00 %
79	1062/719	22	10	12	0	12	45,45 %	0,00 %	100,00 %
80	1074/729	21	6	15	3	18	28,57 %	16,67 %	83,33 %
Total		1707	725	974	103	1075	42,47 %	9,56 %	90,44 %

Tabla 6.1: Resultados obtenidos micrografía a micrografía.

Vemos como una vez hemos aprendido las 80 micrografías, la media de los falsos positivos sigue siendo bastante alta (en torno al 30-40 %). Demasiado alta comparada con las medias de otros modelos descritos en el Capítulo 2. Sin embargo, también cabe destacar la tasa de aciertos, ya que está entre las mejores de dichos modelos (alrededor del 90 %). Esta conclusión es bastante lógica, ya que en la mayoría de trabajos relacionados se habla de una relación negativa entre los falsos positivos y la tasa de aciertos.

Podemos ver estos datos representados gráficamente en la Figura 6.1. Observando detenidamente, vemos como el modelo se ve afectado claramente por la micrografía que hemos aprendido. Esto en cierta medida es normal debido a la varianza de algunos parámetros de las micrografías analizadas. En el proceso de obtención de estos datos se ha observado que, dependiendo de la definición y la luminosidad de la micrografía, los resultados tendían a variar. Este podría ser un punto a mejorar de nuestro modelo, ya que se debería intetar que estas variaciones afecten lo mínimo posible a nuestro modelo. Sin embargo, la medida de esta inestabilidad del modelo se estudia mejor en la siguiente sección, ya que hacemos todos los cálculos sobre una micrografía de referencia.

Como ejemplo representativo, podemos ver en la Figura 6.2 una micrografía al azar de entre la lista, la cual hemos clasificado con nuestro modelo. En dicha figura, las partículas que tienen un rectángulo son partículas marcadas correctamente, mientras que las que tienen un círculo son partículas marcadas erróneamente (falsos positivos). Vemos cómo el modelo ha marcado 26 partículas, de las cuales 8 son errores, es decir, alrededor de un 30 % de falsos positivos, mientras que no se ha dejado ninguna sin marcar.

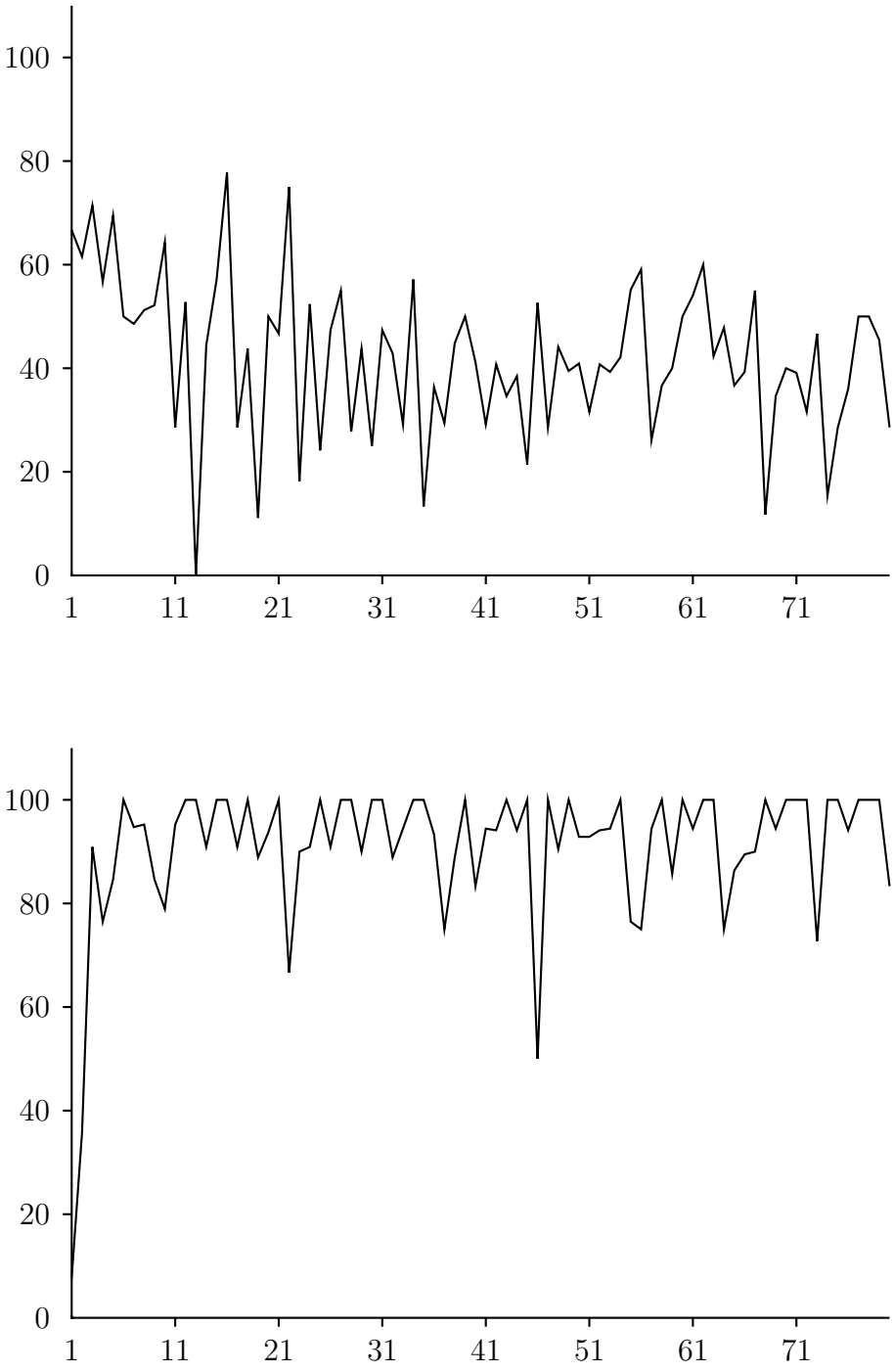


Figura 6.1: Gráfico del progreso del modelo micrografía a micrografía. En la figura superior podemos ver la relación de falsos positivos (FPR), mientras que en la figura inferior tenemos la tasa de aciertos (TPR). El eje vertical se corresponde con el porcentaje correspondiente, mientras que el eje horizontal se corresponde con el número de micrografía analizada

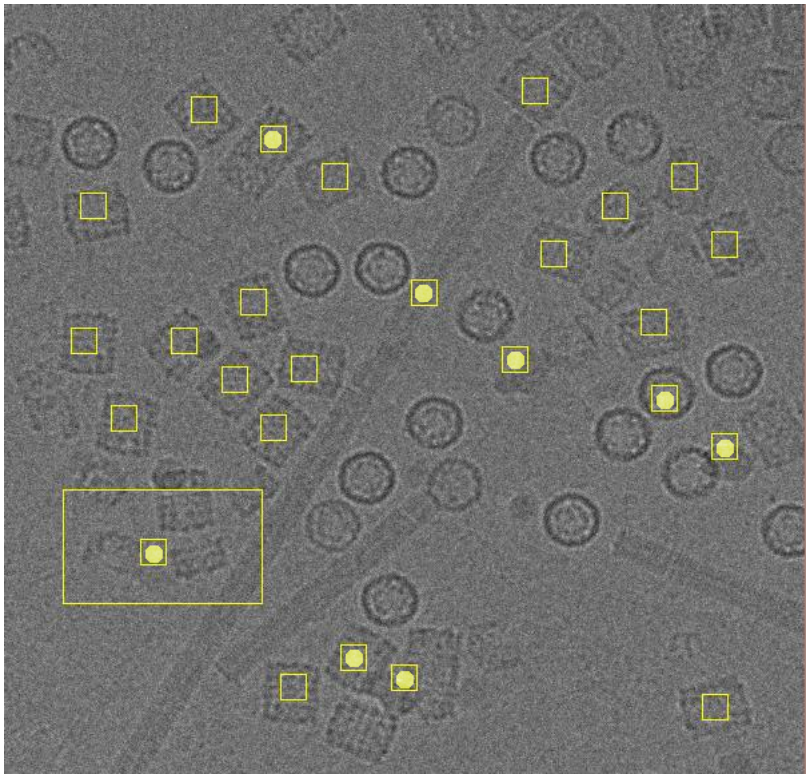


Figura 6.2: Ejemplo de una clasificación con nuestro modelo

6.2.2. Micrografía de referencia

La micrografía de referencia

La micrografía de referencia utilizada para los datos de esta sección fue elegida al azar de entre la lista de micrografías, siempre cumpliendo un mínimo de calidad de imagen para obtener unos resultados fiables. La micrografía elegida es la que tiene como nombre `01nov26b.016.009.001.002.mrc`. En la Figura 6.3 podemos ver dicha micrografía con las partículas que nosotros consideramos correctas marcadas con un rectángulo. Esta micrografía tiene ciertos componentes interesantes para nuestras medidas, puesto que tiene algunas partículas con poco contraste que son difíciles de clasificar. Además, también tiene dos partículas superpuestas y otras dos que están demasiado cerca de los ejes.

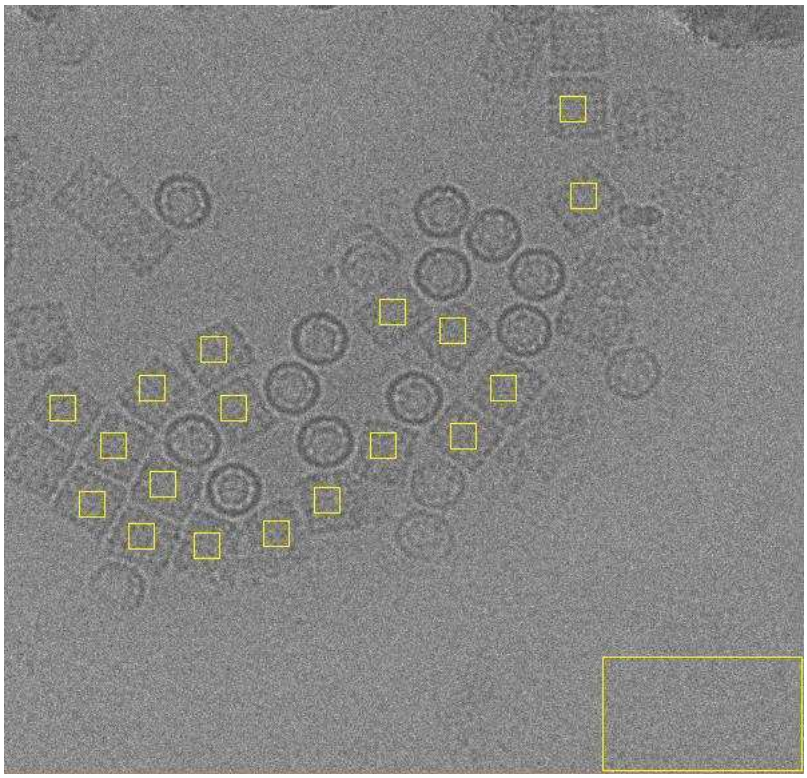


Figura 6.3: Micrografía de referencia para obtener nuestros datos

El criterio para marcar partículas en una micrografía puede variar considerablemente de una persona a otra ya que es bastante subjetivo. En [33] se explican algunos de los criterios más utilizados para marcar partículas en micrografías. Como veíamos en la

figura anterior, hemos considerado que existen 18 partículas.

Resultados

En esta sección, los datos que se muestran son diferentes. En la Tabla 6.2 tenemos los mismos cálculos que en la sección anterior, pero calculados siempre sobre la misma micrografía de referencia (`01nov26b.016.009.001.002.mrc`). Estos datos son más significativos en cuanto a lo que al aprendizaje del modelo se refiere, ya que no hay variaciones de definición, color, impurezas de una medida a otra. Los pasos seguidos para realizar estas medidas son:

1. Dejamos a parte la micrografía `01nov26b.016.009.001.002.mrc` que es con la que vamos a realizar las medidas.
2. Abrimos la siguiente micrografía (la primera si es la primera iteración).
3. Aprendemos la micrografía (buscando partículas y corrigiendo los resultados).
4. Aprendemos la micrografía de referencia.
5. Repetimos los pasos de 2 a 4 hasta aprender todas las micrografías de la lista.

#	P/E	C	F	I	FPR	FNR	TPR
1	17/0	1	5	17	83,33 %	94,44 %	5,56 %
2	30/2	5	11	13	68,75 %	72,22 %	27,78 %
3	44/10	10	18	8	64,29 %	44,44 %	55,56 %
4	55/35	14	18	4	56,25 %	22,22 %	77,78 %
5	71/52	15	20	3	57,14 %	16,67 %	83,33 %
6	84/77	14	19	4	57,58 %	22,22 %	77,78 %
7	93/86	14	18	4	56,25 %	22,22 %	77,78 %
8	112/103	15	20	3	57,14 %	16,67 %	83,33 %
9	133/124	14	17	4	54,84 %	22,22 %	77,78 %
10	146/136	15	16	3	51,61 %	16,67 %	83,33 %
11	165/163	14	11	4	44,00 %	22,22 %	77,78 %
12	186/171	15	12	3	44,44 %	16,67 %	83,33 %
13	203/190	15	12	3	44,44 %	16,67 %	83,33 %
14	212/190	15	15	3	50,00 %	16,67 %	83,33 %

15	223/198	15	12	3	44,44 %	16,67 %	83,33 %
16	229/206	15	13	3	46,43 %	16,67 %	83,33 %
17	231/213	16	9	2	36,00 %	11,11 %	88,89 %
18	242/217	16	11	2	40,74 %	11,11 %	88,89 %
19	251/224	16	12	2	42,86 %	11,11 %	88,89 %
20	260/225	16	13	2	44,83 %	11,11 %	88,89 %
21	276/240	12	13	6	52,00 %	33,33 %	66,67 %
22	284/247	12	13	6	52,00 %	33,33 %	66,67 %
23	287/253	12	14	6	53,85 %	33,33 %	66,67 %
24	297/255	11	12	7	52,17 %	38,89 %	61,11 %
25	308/266	13	13	5	50,00 %	27,78 %	72,22 %
26	330/273	16	11	2	40,74 %	11,11 %	88,89 %
27	341/282	15	14	3	48,28 %	16,67 %	83,33 %
28	350/293	15	11	3	42,31 %	16,67 %	83,33 %
29	363/298	15	11	3	42,31 %	16,67 %	83,33 %
30	373/305	15	10	3	40,00 %	16,67 %	83,33 %
31	385/309	14	10	4	41,67 %	22,22 %	77,78 %
32	395/318	14	9	4	39,13 %	22,22 %	77,78 %
33	404/324	13	9	5	40,91 %	27,78 %	72,22 %
34	422/331	12	10	6	45,45 %	33,33 %	66,67 %
35	428/339	12	8	6	40,00 %	33,33 %	66,67 %
36	441/341	13	9	5	40,91 %	27,78 %	72,22 %
37	456/349	13	10	5	43,48 %	27,78 %	72,22 %
38	472/354	14	10	4	41,67 %	22,22 %	77,78 %
39	490/367	16	10	2	38,46 %	11,11 %	88,89 %
40	498/375	15	10	3	40,00 %	16,67 %	83,33 %
41	510/382	15	11	3	42,31 %	16,67 %	83,33 %
42	528/389	14	11	4	44,00 %	22,22 %	77,78 %
43	544/400	15	10	3	40,00 %	16,67 %	83,33 %
44	561/409	16	11	2	40,74 %	11,11 %	88,89 %
45	578/419	16	8	2	33,33 %	11,11 %	88,89 %
46	600/425	15	8	3	34,78 %	16,67 %	83,33 %
47	618/435	16	10	2	38,46 %	11,11 %	88,89 %

48	633/442	16	10	2	38,46 %	11,11 %	88,89 %
49	654/457	16	10	2	38,46 %	11,11 %	88,89 %
50	677/472	16	11	2	40,74 %	11,11 %	88,89 %
51	691/481	15	9	3	37,50 %	16,67 %	83,33 %
52	705/487	16	11	2	40,74 %	11,11 %	88,89 %
53	722/498	16	11	2	40,74 %	11,11 %	88,89 %
54	740/509	16	10	2	38,46 %	11,11 %	88,89 %
55	751/517	15	7	2	31,82 %	11,11 %	83,33 %
56	768/533	16	7	2	30,43 %	11,11 %	88,89 %
57	780/546	16	9	2	36,00 %	11,11 %	88,89 %
58	798/552	16	7	2	30,43 %	11,11 %	88,89 %
59	817/563	16	6	2	27,27 %	11,11 %	88,89 %
60	824/567	16	6	2	27,27 %	11,11 %	88,89 %
61	825/568	16	9	2	36,00 %	11,11 %	88,89 %
62	843/588	15	6	3	28,57 %	16,67 %	83,33 %
63	847/594	14	4	4	22,22 %	22,22 %	77,78 %
64	862/605	14	4	4	22,22 %	22,22 %	77,78 %
65	878/616	16	5	2	23,81 %	11,11 %	88,89 %
66	900/628	16	5	2	23,81 %	11,11 %	88,89 %
67	919/639	15	5	3	25,00 %	16,67 %	83,33 %
68	929/650	15	6	3	28,57 %	16,67 %	83,33 %
69	944/652	16	5	2	23,81 %	11,11 %	88,89 %
70	962/661	17	6	1	26,09 %	5,56 %	94,44 %
71	971/667	17	7	1	29,17 %	5,56 %	94,44 %
72	985/676	17	7	1	29,17 %	5,56 %	94,44 %
73	998/682	17	10	1	37,04 %	5,56 %	94,44 %
74	1009/689	17	8	1	32,00 %	5,56 %	94,44 %
75	1020/691	16	6	2	27,27 %	11,11 %	88,89 %
76	1030/695	16	5	2	23,81 %	11,11 %	88,89 %
77	1047/704	16	6	2	27,27 %	11,11 %	88,89 %
78	1056/713	16	8	2	33,33 %	11,11 %	88,89 %
79	1062/719	16	8	2	33,33 %	11,11 %	88,89 %
80	1074/729	15	6	3	28,57 %	16,67 %	83,33 %

Tabla 6.2: Resultados obtenidos para una micrografía de referencia

Vemos como los datos van mejorando a medida que vamos recorriendo la lista de micrografías y vamos aprendiendo partículas y errores nuevos. Vemos cómo las varianzas de una micrografía a otra afectan al modelo, ya que en algunos puntos de la gráfica hay cambios de pendiente importantes, pero no son variaciones demasiado preocupantes.

6.3. Pesos de las características

Para posteriores investigaciones sobre el modelo implementado, consideramos bastante útil estudiar los pesos que han sido utilizados para ponderar cada una de las 142 características. En la Figura 6.5 podemos ver el resultado de este estudio.

Observando detenidamente los resultados, vemos cómo las características más importantes están situadas en la parte final del histograma de los anillos radiales (ver Sección 3.5). Esto es bastante lógico, teniendo en cuenta que la parte que más varía entre una Partícula y una No-Partícula o un Error es el borde de la subimagen estudiada. Una Partícula tiene el borde bastante más oscuro que el interior de la misma. Además, los Errores y las No-Partículas tienen menos intensidad en dichos píxeles. Las primeras características correspondientes al histograma de la imagen original tienen un peso bastante bajo. Probablemente el modelo mejoraría sustituyendo estas características por otras más útiles. Por último, las características del espectro rotacional no son las más importantes, pero parece que aportan la suficiente información como para ser relevantes.

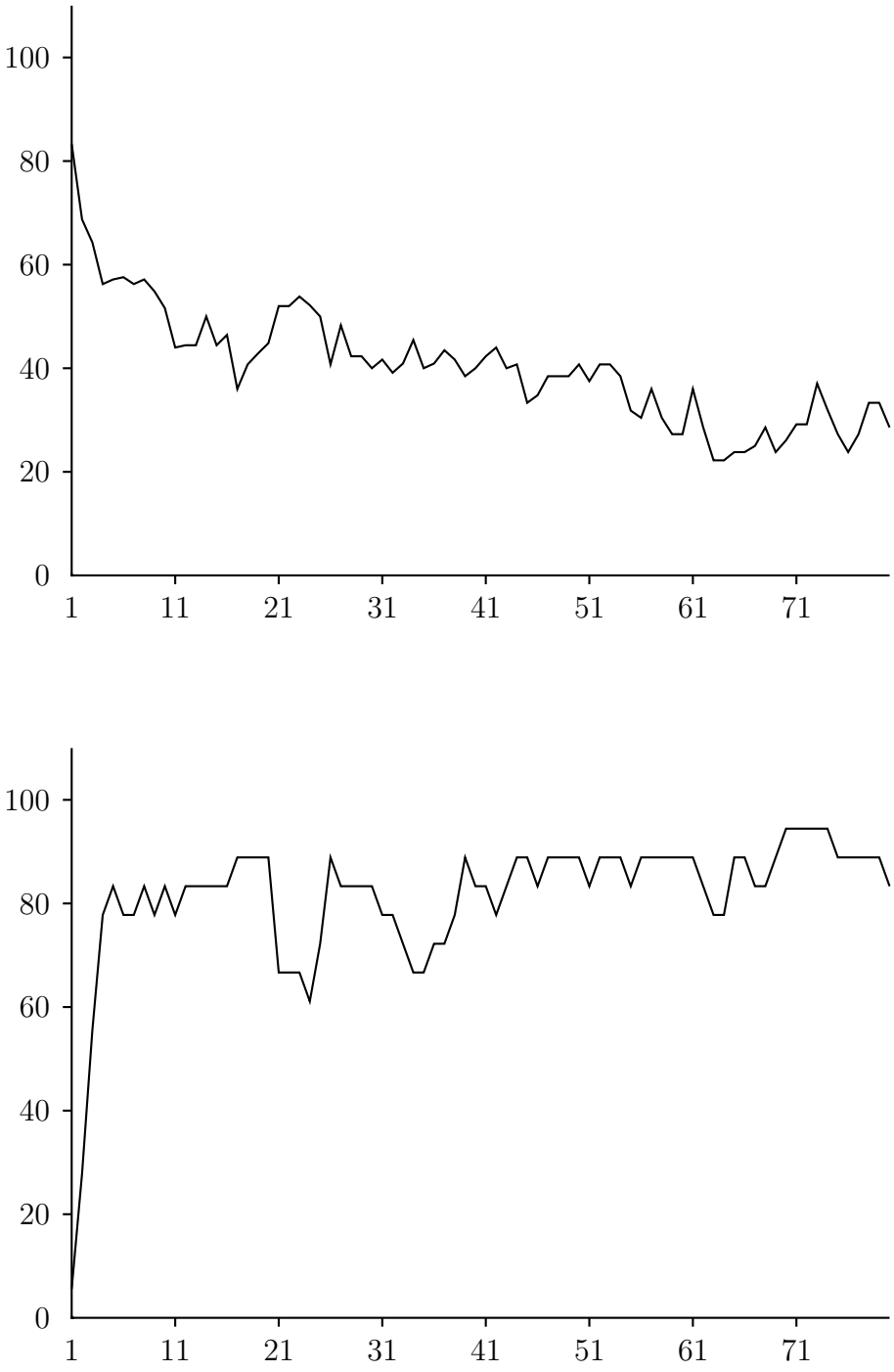


Figura 6.4: Gráfico del progreso del modelo para la micrografía de referencia. En la figura superior podemos ver la relación de falsos positivos (FPR), mientras que en la figura inferior tenemos la tasa de aciertos (TPR). El eje vertical se corresponde con el porcentaje correspondiente, mientras que el eje horizontal se corresponde con el número de micrografía analizada

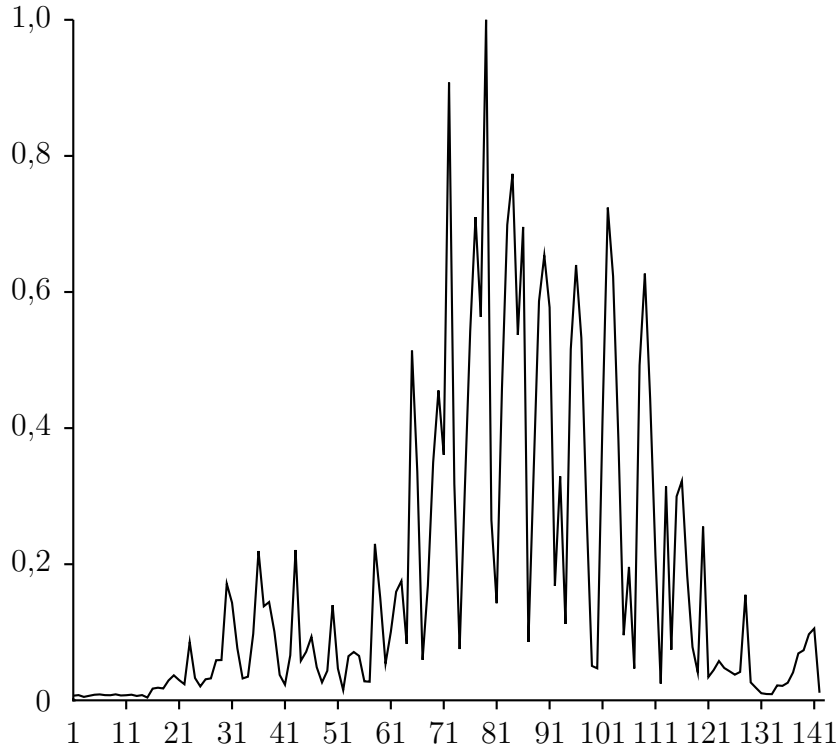


Figura 6.5: Pesos de las características. En el eje vertical se representa el peso asignado, mientras que las características están indexadas en el eje horizontal

Capítulo 7

Conclusiones y futuro

Un nuevo modelo para la detección automática de partículas basado en las Redes Bayesianas ha sido expuesto en este trabajo. Este nuevo modelo tiene como principal característica su gran flexibilidad, puesto que aprende dinámicamente según se van marcando nuevas partículas y ajusta las probabilidades del modelo para futuras detecciones. Además, cabe resaltar su simplicidad, puesto que se basa en un caso concreto de las redes bayesianas, llamado Clasificador *Naive Bayes*, el cual es increiblemente sencillo, pero ha demostrado su eficacia como clasificador en numerosos campos.

En el campo de la microscopía electrónica existen numerosos modelos similares. Hay muchos de estos modelos que han conseguido resultados mejores, sobre todo en cuanto a la eliminación de los falsos-positivos, estando entre los mejores en lo que a los falsos-negativos (tasa de acierto) se refiere.

Cabe destacar que los principales problemas de este modelo son referentes a las orientaciones indeseadas, pero más concretamente a aquellas que tienen cierto parecido con las orientaciones laterales (bordes menos marcados, forma semicircular, etc.) y cuanto más difusas peor. Además, también tiene serios problemas para desechar partículas superpuestas.

En conclusión, este modelo detecta muy bien las partículas reales y todo lo que tenga un parecido razonable con ellas, lo que en una micrografía convencional puede dar lugar a numerosos falsos positivos. Esta robustez del modelo ante las partículas positivas puede dar lugar a mejoras del modelo que consigan descartar mejor los posibles falsos positivos. En la siguiente sección se dan alguas ideas sobre futuras mejoras que se podrían aplicar sobre el modelo ya implementado.

7.1. Posibles mejoras

Como futuras ampliaciones a este proyecto se proponen diversas mejoras:

- En primer lugar, sería necesario mejorar la velocidad de ejecución del programa, ya que se tarda demasiado en escanear las micrografías. Se podría optimizar el código para minimizar en la medida de lo posible los bucles y comprobaciones existentes en el proceso de detección. Se podría intetar hacer el código paralelizable, de manera que si se tiene un equipo multiprocesador, cada procesador pueda escanear una parte diferente de la micrografía.

- Uno de los principales problemas en la detección han sido las orientaciones indeseadas de la partícula estudiada (las orientaciones verticales aparecen como círculos, mientras que las orientaciones laterales aparecen como rectángulos), así como partículas superpuestas e impurezas. Para evitar estos errores habría que estudiar introducir alguna nueva característica en el modelo de clasificación, de manera que el clasificador distinga mejor las formas y contornos de las partículas. Una posible aplicación de esta idea podría ser aplicar un filtro de Canny (detector de ejes) a la partícula escaneada, de manera que un número de características del vector se refieran a cómo es el contorno de la muestra observada.

- Uno de los pasos de la inicialización de la red bayesiana consiste en cuantizar el histograma de cada características en 8 niveles para cada una de las clases. Esta cuantización ahora mismo se hace mediante el cálculo de la entropía de cada histograma, partiéndolo varias veces por el punto donde aparece la máxima entropía, de manera que cada histograma contenga la mayor cantidad de información posible. Es posible que el modelo mejorase si en este paso se introdujera algún tipo de cuantización vectorial más avanzada para precisar más a la hora de determinar si un determinado valor de característica pertenece a una clase o a otra.

- Otra posible mejora sobre el modelo podría ser cambiar ligeramente el clasificador. En vez de utilizar un clasificador *Naive Bayes* que clasifique según las 142 características de cada subimagen, se podría crear lo que se conoce como **Bosque Aleatorio** (*Random Forests*). Este tipo de clasificadores se componen de un número determinado de árboles de decisión. El resultado del clasificador es una función de la salida de cada uno de estos árboles de decisión. Por ejemplo, se podría hacer que el modelo fuera un bosque aleatorio de cuatro árboles de

decisión. Se divide el conjunto de características de manera que cada árbol de decisión clasifique sobre un subconjunto de características (por ejemplo, que el primero clasifique sobre las 20 primeras características, el segundo sobre las 54 siguientes, etc.). Esta clasificación de cada árbol de decisión se llevaría a cabo de la misma manera que la descrita en este proyecto, es decir, mediante un clasificador *Naive Bayes*. Por tanto, tendríamos un clasificador que sería una función de cuatro clasificadores *Naive Bayes*, cada uno clasificando sobre un subconjunto de características diferente. Esto debería darle al modelo mayor precisión, ya que cada clasificador *Naive Bayes* sería más específico al tener que clasificar menos variables.

- Agilizar al proceso global de detección, puesto que ahora es algo engorroso:

  1. Aprender las partículas una primera micrografía y guardar el modelo.
  2. Cargar el modelo, escanear la micrografía en busca de partículas.
  3. Eliminar las partículas sobrantes (las cuales se aprenden como errores) y marcar las que el modelo no haya marcado.
  4. Aprender los datos.

- Una mejora interesante aplicable a este modelo podría ser aprender cómo mejorar los pesos de las características dinámicamente, de manera que podamos minimizar el error del modelo modificando estos pesos. Esto se haría en la fase de entrenamiento de la red bayesiana, y para conseguirlo, podríamos intentar predecir el resultado a medida que aprendemos los diferentes vectores de entrenamiento e ir modificando los pesos de las características hasta que consigamos que el error sea mínimo.

Bibliografía

[1] `http://fai.unne.edu.ar/biologia/microscopia/meb.htm`.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[3] H. Chi Wong, J. Chen, F. Mouche, I. Rouiller, and M. Bern. Model-based particle picking for cryo-electron microscopy. *Journal of Structural Biology*, 145:157–167, 2004.

[4] R.A. Crowther and Linda A. Amos. Harmonic analysis of electron microscope images with rotational symmetry. *Journal of Molecular Biology*, 60:123–130, 1971.

[5] J. T. A. S. Ferreira, D. G. T. Denison, and D. J. Hand. Weighted naive bayes modelling for data mining, 2001.

[6] Eibe Frank, Mark Hall, and Bernhard Pfahringer. Locally weighted naive bayes, 2003.

[7] Nir Friedman. The Bayesian structural EM algorithm. In *UAI*, pages 129–138.

[8] R. J. Hall and A. Patwardhan. A two step approach for semi-automated particle selection from low contrast cryo-electron micrographs. *Journal of Structural Biology*, 145:19–28, 2004.

[9] David Heckerman. A tutorial on learning with bayesian networks. pages 301–354, 1999.

[10] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994.

[11] Z. Huang and P. A. Penczek. Application of template matching technique to particle detection in electron micrographs. *Journal of Structural Biology*, 145:29–40, 2004.

[12] Petri Kontkanen, Petri Myllymaki, Tomi Silander, and Henry Tirri. BAYDA: Software for bayesian classification and feature selection. In *Knowledge Discovery and Data Mining*, pages 254–258, 1998.

[13] V. Kumar, J. Heikkonen, P. Engelhardt, and K. Kaski. Robust filtering and particle picking in micrograph images towards 3d reconstruction of purified proteins with cryo-electron microscopy. *Journal of Structural Biology*, 145:41–51, 2004.

[14] S. P. Mallick, Y. Zhu, and D. Kriegman. Detecting particles in cryo-em micrographs using learned features. *Journal of Structural Biology*, 145:52–62, 2004.

[15] J. A. Marchant and Onyango C. M. Comparison of a bayesian classifier with a multilayer feed-forward neural network using the example of plant/weed/soil discrimination. *Computers and Electronics in Agriculture*, 39:3–22, 2003.

[16] Kevin Murphy. A brief introduction to graphical models and bayesian networks.

[17] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, April 2003.

[18] W. V. Nicholson and Glaeser R. M. Review: Automatic particle detection in electron microscopy. *Journal of Structural Biology*, 133:90–101, 2001.

[19] T. Ogura and C. Sato. Auto-accumulation method using simulated annealing enables fully automatic particle pickup completely free from a matching template or learning data. *Journal of Structural Biology*, 146:344–358, 2004.

[20] T. Ogura and C. Sato. Automatic particle pickup method using a neural network has high accuracy by applying an initial weight derived from eigenimages: a new reference free method for single-particle analysis. *Journal of Structural Biology*, 145:63–75, 2004.

[21] J. R. Plaisier, R. I. Koning, H. K. Koerten, M. van Heel, and J. P. Abrahams. Tyson: Robust searching, sorting, and selecting of single particles in electron micrographs. *Journal of Structural Biology*, 145:76–83, 2004.

[22] B. K. Rath and J. Frank. Fast automatic particle picking from cryo-electron micrographs using a locally normalized cross-correlation function: a case study. *Journal of Structural Biology*, 145:84–90, 2004.

[23] A. M. Roseman. Findem–a fast, efficient program for automatic selection of particles from electron micrographs. *Journal of Structural Biology*, 145:91–99, 2004.

[24] J. M. Short. Sleuth–a fast computer program for automatically detecting particles in electron microscope images. *Journal of Structural Biology*, 145:100–110, 2004.

[25] F. J. Sigworth. Classical detection theory and the cryo-em particle selection problem. *Journal of Structural Biology*, 145:111–122, 2004.

[26] K. Singh, D. C. Marinescu, and T. S. Baker. Image segmentation for automatic particle identification in electron micrographs based on hidden markov random field models and expectation maximization. *Journal of Structural Biology*, 145:123–141, 2004.

[27] C. O. S. Sorzano, R. Marabini, J. Velquez-Muriel, J. R. Bilbao-Castro, S. H. W. Scheres, J. M. Carazo, and A. Pascual-Montano. Xmipp: a new generation of an open-source image processing package for electron microscopy. *Journal of Structural Biology*, 148:194–204, 2004.

[28] C.O.S. Sorzano, E. Ortiz, M. Lopez, and J. Rodrigo. Improved bayesian image denoising based on wavelets with applications to electron microscopy. *Pattern Recognition*, 39(6):1205–1213, 2006.

[29] P. S. Umesh Adiga, R. Malladi, W. Baxter, and R. Glaeser. A binary segmentation approach for boxing ribosome particles in cryo em micrographs. *Journal of Structural Biology*, 145:142–151, 2004.

[30] N. Volkmann. An approach to automatic particle picking from electron micrographs based on reduced representation templates. *Journal of Structural Biology*, 145:152–156, 2004.

[31] Z. Yu and C. Bajaj. Detecting circular and rectangular particles based on geometric feature detection in electron micrographs. *Journal of Structural Biology*, 145:168–180, 2004.

[32] Harry Zhang. The optimality of naive bayes. In *FLAIRS Conference*. AAAI Press, 2004.

[33] Y. Zhu, B. Carragher, R. M. Glaeser, D. Fellmann, C. Bajaj, M. Bern, F. Mouche, F. de Haas, R. J. Hall, D. J. Kriegman, S. J. Ludtke, S. P. Mallick, P. A. Penczek, A. M. Roseman, F. J. Sigworth, N. Volkmann, and C. S. Potter. Automatic particle selection: results of a comparative study. *Journal of Structural Biology*, 145:3–14, 2004.