ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

ALGORITMOS PARA LA ESTIMACIÓN DE MODELOS DE MEZCLAS GAUSSIANAS (ALGORITHMS FOR GAUSSIAN MIXTURE MODEL (GMM) ESTIMATION)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Alberto García Herrero

Julio - 2015

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: **Alberto García Herrero**

Director del TFG: **Jesús Pérez Arriaga**

Título: **"Algoritmos para la estimación de modelos de mezclas Gaussianas"**

Title: **"Algorithms for Gaussian mixture model (GMM) estimation"**

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente (Apellidos, Nombre):

Secretario (Apellidos, Nombre):

Vocal (Apellidos, Nombre):

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente Fdo.: El Secretario

Fdo.: El Vocal Fdo.: El Director del TFG

(sólo si es distinto del Secretario)

Vº Bº del Subdirector Trabajo Fin de Grado Nº (a asignar por Secretaría)

Agradecimientos

Este trabajo no habría sido posible sin la ayuda recibida. Por ello, quiero agradecer en primer lugar al director de este Trabajo Fin de Grado, Jesús Pérez, por ofrecerme la posibilidad de trabajar en este proyecto y por la atención, tiempo y ayuda que me ha prestado durante el desarrollo del mismo.

A todo el grupo GTAS, que me han ayudado siempre que lo he necesitado y han hecho que me sienta uno más desde el inicio.

A todos aquellos profesores que me han proporcionado los conocimientos y las herramientas necesarias durante el transcurso de la carrera.

A los amigos y compañeros con los que he compartido muchas horas de clase y han hecho que todos estos años hayan sido mucho más llevaderos.

Y por último y no menos importante a mi familia, a mis padres y mi hermana, por todo el apoyo recibido, ánimo y comprensión.

Índice general

Índice de figuras

Índice de tablas

Palabras clave

Modelo de mezclas Gaussiana, algoritmo EM, estimación máxima verosimilitud, K-means, algoritmos adaptativos, algoritmos online, descenso por gradiente, clústering, clúster

Keywords

Gaussian mixture model, EM algorithm, maximum likelihood estimation, K-means, adaptative algorithms, online algorithms, gradient descent, clustering, cluster

Capítulo 1: Introducción

1.1 Motivación

En las últimas décadas se ha producido un crecimiento exponencial en la generación de la información debido en gran parte a la automatización de procesos y a los avances en las capacidades de almacenamiento de información. A diario, se almacena gran cantidad de información representada por datos para analizarlos posteriormente. En muchos campos de la investigación está presente el problema de revelar la estructura interna de estos. Una posible forma es clasificarlos en pequeños grupos que describan sus características principales, basándose en la similitud o diferencia entre ellos. El problema de clasificar los datos almacenados para obtener información útil ha sido intensamente estudiado en el área del Reconocimiento de Patrones no supervisado.

El valor de los datos reside en la información que podamos extraer de los mismos, información que nos ayude a tomar decisiones o a simplemente mejorar nuestra comprensión de los fenómenos que nos rodean.

El trabajo desarrollado en este proyecto se enmarca en el ámbito del análisis de datos, y más concretamente en el estudio de ciertos métodos de agrupamiento de datos y su aplicación a problemas concretos. Para ello se considerará un modelo de mezclas Gaussianas que se describirá a continuación.

1.2 Planteamiento del problema: modelo de mezclas Gaussianas

Sea $\boldsymbol{Y} = [Y_1, Y_2, \dots, Y_D]^T$ una variable aleatoria real D-dimensional. Se dice que la distribución de **Y** sigue una distribución mezcla finita si su función densidad de probabilidad (fdp) se puede escribir como una combinación lineal de fdp's elementales

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{i=1}^{I} \alpha_i p(\boldsymbol{y}|C = i, \boldsymbol{\beta}_i), \qquad i\epsilon\{1, \dots, I\} \qquad (1.1)$$

Donde I representa el número de distribuciones elementales (componentes) de la mezcla, $C = 1, 2, \dots, I$, y $\boldsymbol{\theta}$ representa el conjunto de parámetros

$$\boldsymbol{\theta} = \{\alpha_1, \dots, \alpha_I, \boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_I\} \qquad (1.2)$$

Siendo $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_I\}$ el conjunto de parámetros asociados a cada distribución de la mezcla y $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_I\}$ la probabilidad o peso de cada distribución de la mezcla.

Cuando las distribuciones que componen la mezcla son Gaussianas, la función densidad de probabilidad *(1.1)* se conoce como mezcla gaussiana. Por tanto una mezcla Gaussiana es una distribución probabilística cuya fdp es una combinación lineal de distribuciones Gaussianas

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{i=1}^{I} \alpha_i N(\boldsymbol{y}|C = i, \boldsymbol{\beta}_i) \qquad (1.3)$$

Siendo

$$N(\boldsymbol{y}|C = i, \boldsymbol{\beta}_i) = \frac{1}{(2\pi)^{\frac{D}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_i)\right) \qquad (1.4)$$

Y los parámetros de cada Gaussiana

$$\boldsymbol{\beta}_i = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\} \qquad (1.5)$$

donde $\boldsymbol{\mu}_i \epsilon \mathbb{R}^D$ y $\boldsymbol{\Sigma}_i \epsilon \mathbb{R}^{DxD}$ son la media y la matriz de covarianza de la componente i-ésima [1], [2], [3], [4].

1.2.1 Restricciones de los parámetros de una mezcla de Gaussianas

Los parámetros $\boldsymbol{\alpha}$ y $\boldsymbol{\Sigma}_i$ presentan una serie de restricciones a tener en cuenta. Las probabilidades de la mezcla, deben verificar

$$\alpha_i \geq 0, \quad i\epsilon\{1, \dots, I\}, \qquad \sum_{i=1}^{I} \alpha_i = 1 \qquad (1.6)$$

Y las matrices de covarianza deben cumplir las siguientes restricciones

$$\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}_i^T$$
$$\sigma_{jj}^2 \geq 0, \qquad j\epsilon\{1, \dots, D\} \qquad (1.7)$$

Las matrices de covarianza deben ser simétricas, además en los elementos de la diagonal se encuentran las varianzas y deben ser no negativas [1], [2].

1.2.2 Estimación de Máxima Verosimilitud (ML)

Sea un conjunto de datos $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_J\}$ donde $\boldsymbol{y}_j \epsilon \mathbb{R}^D$ es una de las J realizaciones independientes e idénticamente distribuidas (i.i.d.) de la variable aleatoria $\boldsymbol{Y}$. Dado un modelo estadístico, la estima ML proporciona las estimas de los parámetros del modelo a partir de un conjunto de datos [1], [2].

La verosimilitud de $\boldsymbol{\mathcal{Y}}$ es

$$L(\boldsymbol{\theta}) = \prod_{j=1}^{J} p(\boldsymbol{y}_j|\boldsymbol{\theta}) \qquad (1.8)$$

La función verosimilitud es una función de los parámetros $\boldsymbol{\theta}$ del modelo. El estimador ML para $\boldsymbol{\theta}$

$$\widehat{\boldsymbol{\theta}}_{ML}(\boldsymbol{\mathcal{Y}}) = argmax_{\boldsymbol{\theta}}\, L(\boldsymbol{\theta}) = argmax_{\boldsymbol{\theta}} \prod_{j=1}^{J} p(\boldsymbol{y}_j|\boldsymbol{\theta}) \qquad (1.9)$$

Habitualmente se maximiza el logaritmo de la verosimilitud por ser más sencillo de resolver analíticamente. Esto es posible debido a que el logaritmo es una función monótona creciente

$$\widehat{\boldsymbol{\theta}}_{ML}(\boldsymbol{\mathcal{Y}}) = argmax_{\boldsymbol{\theta}} \sum_{j=1}^{J} log\, p(\boldsymbol{y}_j|\boldsymbol{\theta}) \qquad (1.10)$$

Aplicado al modelo de mezclas Gaussianas

$$\widehat{\boldsymbol{\theta}}_{ML}(\boldsymbol{\mathcal{Y}}) = argmax_{\boldsymbol{\theta}} \sum_{j=1}^{J} log \left\{ \sum_{i=1}^{I} \alpha_i N(\boldsymbol{y}_j|C = i, \boldsymbol{\beta}_i) \right\} \qquad (1.11)$$

Se trata de un problema de optimización difícil, no existe una forma cerrada para $\widehat{\boldsymbol{\theta}}_{ML}$. Para ello se proponen diferentes algoritmos en los capítulos siguientes, en concreto el algoritmo Expectation Maximization (algoritmo EM) en el *Apartado 2.1* y el algoritmo de Descenso por Gradiente Estocástico (SGD) en el *Apartado 3.1*.

Capítulo 2: Algoritmos Batch

Los algoritmos batch (bloque) son algoritmos iterativos que se caracterizan por procesar en cada iteración todo el conjunto de observaciones $\boldsymbol{\mathcal{Y}}$. En este capítulo se presentará el algoritmo EM y el algoritmo K-means. El primero se explicará aplicado al modelo de mezclas Gaussianas para facilitar la estimación ML *(1.11)*. El algoritmo K-means no tiene como objetivo resolver *(1.11)* sino que presenta un planteamiento diferente, resuelve un problema geométrico.

2.1 Algoritmo EM

El algoritmo EM es un método iterativo para encontrar la estima ML del conjunto parámetros de un modelo estadístico. Estos modelos probabilísticos dependen de un conjunto de parámetros no observables $\boldsymbol{\theta}$ [1], [2].

Se consideran dos tipos de datos, los datos observados $\boldsymbol{\mathcal{Y}}$ generados por el modelo estadístico y un conjunto de datos ocultos (no observados) $\boldsymbol{\mathcal{Z}} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots, \boldsymbol{z}_J\}$, donde $\boldsymbol{z}_j \epsilon\, \mathbb{R}^D$ es una de las J realizaciones i.i.d. de la variable aleatoria $\boldsymbol{\mathcal{Z}}$. Se denominará $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{Y}} \cup \boldsymbol{\mathcal{Z}}$ al conjunto de datos completos.

La introducción de los datos ocultos es una construcción artificial, sin embargo, aunque parezca sorprendente favorecerá la estimación de $\widehat{\boldsymbol{\theta}}_{ML}$. Hay que tener en cuenta que cada elemento de $\boldsymbol{\mathcal{Z}}$ es una realización de una variable aleatoria oculta, estas realizaciones no existen en la realidad ($\boldsymbol{\mathcal{Z}}$ es desconocida).

Se plantea la función de log-verosimilitud para los datos completos

$$\log L_c(\boldsymbol{\theta}) = \log p(\boldsymbol{\mathcal{X}}|\boldsymbol{\theta}) \qquad (2.1)$$

En el algoritmo EM, la función verosimilitud es función de los datos completos, es una variable aleatoria ya que depende de los datos ocultos $\boldsymbol{\mathcal{Z}}$.

El algoritmo EM hace frente a este problema calculando en primer lugar el valor esperado de la log-verosimilitud para los datos completos *(2.1)* condicionada a los datos observados $\boldsymbol{\mathcal{Y}}$ y una estima previa de $\boldsymbol{\theta}$ ($\widehat{\boldsymbol{\theta}}^{(n)}$)

$$\mathbb{E}_{\boldsymbol{z}}[\log L_c(\boldsymbol{\theta})|\boldsymbol{\mathcal{Y}}, \widehat{\boldsymbol{\theta}}^{(n)}] \qquad (2.2)$$

A continuación, el algoritmo EM obtiene una nueva estima de los parámetros maximizando *(2.2)*

$$\widehat{\boldsymbol{\theta}}^{(n+1)} = argmax_{\boldsymbol{\theta}}\, \mathbb{E}_{\boldsymbol{z}}[\log L_c(\boldsymbol{\theta})|\boldsymbol{\mathcal{Y}}, \widehat{\boldsymbol{\theta}}^{(n)}] \qquad (2.3)$$

El cálculo de *(2.2)* se conoce como Expectation step (Paso E) y el cálculo de *(2.3)* como Maximization step (Paso M). Estos dos pasos se repiten iterativamente desde una estima inicial $\boldsymbol{\theta}^{(0)}$ hasta la convergencia de la estima [1].

La parte más complicada del algoritmo EM, es elegir la $\boldsymbol{\mathcal{Z}}$ apropiada para cada problema. En el caso de utilizar un modelo de mezclas Gaussianas [1], [5], una buena elección para los datos no observados $\boldsymbol{\mathcal{Z}}$ es una matriz en cuyos elementos z_{ij} sean igual a uno si la observación j-ésima proviene o corresponde a la Gaussiana i-ésima (clase $C = i$). La matriz $\boldsymbol{\mathcal{Z}}$ presenta la siguiente estructura

$$\boldsymbol{\mathcal{Z}} = \begin{pmatrix} z_{11} & \cdots & z_{1J} \\ \vdots & \ddots & \vdots \\ z_{I1} & \cdots & z_{IJ} \end{pmatrix} \in \mathbb{N}^{\mathrm{IxJ}} \qquad (2.4)$$

En cada columna habrá un 1, los demás valores serán 0. Por medio de esta elección se introduce información adicional al proceso.

Para el modelo de mezclas Gaussianas la verosimilitud para los datos completos es

$$L_c(\boldsymbol{\theta}) = p(\boldsymbol{\mathcal{X}}|\boldsymbol{\theta}) = \prod_{j=1}^{J} \sum_{i=1}^{I} z_{ij}\alpha_i\, N(\boldsymbol{y}_j|C = i, \boldsymbol{\beta}_i) \qquad (2.5)$$

Por tanto,

$$\log L_c(\boldsymbol{\theta}) = \sum_{j=1}^{J} \log\left(\sum_{i=1}^{I} z_{ij}\alpha_i\, N(\boldsymbol{y}_j|C = i, \boldsymbol{\beta}_i)\right) \qquad (2.6)$$

La expresión anterior puede reescribirse como

$$\log L_c(\boldsymbol{\theta}) = \sum_{j=1}^{J} \sum_{i=1}^{I} z_{ij} \log \alpha_i\, N(\boldsymbol{y}_j|C = i, \boldsymbol{\beta}_i) \qquad (2.7)$$

Ya que en el sumatorio interno de la expresión *(2.6)* solo hay un término distinto de 0.

En el *Apéndice I* se detalla la aplicación del Paso E del algoritmo a este problema. A continuación se muestran los estimadores para el caso de *D* dimensiones que se obtendrían en el paso de Maximización [5]. Para una explicación detallada del procedimiento de obtención de los estimadores de cada parámetro en el caso de una dimensión consultar *Apéndice II.*

Optimización con respecto a α_i

$$\hat{\alpha}_i^{(n+1)} = \frac{1}{J}\sum_{j=1}^{J} \mathbb{E}_{\boldsymbol{z}}\left[z_{ij} | \boldsymbol{y}_j, \hat{\boldsymbol{\theta}}_i^{(n)} \right] \qquad (2.8)$$

Maximización con respecto $\boldsymbol{\mu}_i$

$$\hat{\boldsymbol{\mu}}_i^{(n+1)} = \frac{\sum_{j=1}^{J} \boldsymbol{y}_j \mathbb{E}_{\boldsymbol{z}}\left[z_{ij} | \boldsymbol{y}_j, \hat{\boldsymbol{\theta}}_i^{(n)} \right]}{\sum_{j=1}^{J} \mathbb{E}_{\boldsymbol{z}}\left[z_{ij} | \boldsymbol{y}_j, \hat{\boldsymbol{\theta}}_i^{(n)} \right]} \qquad (2.9)$$

Maximización con respecto $\boldsymbol{\Sigma}_i$

$$\hat{\boldsymbol{\Sigma}}_i^{(n+1)} = \frac{\sum_{j=1}^{J} \mathbb{E}_{\boldsymbol{z}}\left[z_{ij} | \boldsymbol{y}_j, \hat{\boldsymbol{\theta}}_i^{(n)} \right] \left(\boldsymbol{y}_j - \hat{\boldsymbol{\mu}}_i^{(n+1)} \right) \left(\boldsymbol{y}_j - \hat{\boldsymbol{\mu}}_i^{(n+1)} \right)^T}{\sum_{j=1}^{J} \mathbb{E}_{\boldsymbol{z}}\left[z_{ij} | \boldsymbol{y}_j, \hat{\boldsymbol{\theta}}_i^{(n)} \right]} \qquad (2.10)$$

donde

$$\mathbb{E}_{\boldsymbol{z}}\left[z_{ij} | \boldsymbol{y}_j, \hat{\boldsymbol{\theta}}_i^{(n)} \right] = \frac{N\left(\boldsymbol{y}_j \middle| C = i, \hat{\boldsymbol{\beta}}_i{}^{(n)} \right) \cdot \hat{\alpha}_i^{(n)}}{\sum_{i=1}^{I} \hat{\alpha}_i^{(n)} N\left(\boldsymbol{y}_j \middle| C = i, \hat{\boldsymbol{\beta}}_i{}^{(n)} \right)} \qquad (2.11)$$

Una propiedad importante del algoritmo EM es que las estimas que se producen en cada nueva iteración siempre aumentan el valor de la verosimilitud *(1.11)*. La ejecución de los pasos E y M iterativamente se produce hasta que se cumple un criterio de convergencia. Habitualmente se itera hasta que la diferencia entre las actualizaciones de las estimas de los parámetros *(1.2)* es muy pequeña o hasta que la diferencia entre el valor de la verosimilitud de los datos incompletos *(1.11)* en una iteración con respecto a la anterior es inferior a un umbral fijado por el usuario.

2.1.1 El problema de la inicialización y la determinación del nº de clases

Una de las mayores dificultades es la inicialización del número de Gaussianas *I* de la mezcla en problemas en los que *I* es desconocido a priori. Un número elevado de Gaussianas podría provocar una partición excesiva de los datos y un número muy reducido puede que no sea lo suficientemente flexible para la aproximación al modelo real que siguen los datos para un problema determinado. Por tanto, en principio es preferible no utilizar un valor de *I* muy bajo a la hora de elegir el número de clases iniciales de la mezcla. Existen diversas formas de encontrar el valor de *I* adecuado para un problema. En ocasiones si se sobredimensiona *I*, al analizar los datos, para algunas clases pueden obtenerse unos valores de $\boldsymbol{\alpha}$ muy pequeños (varios órdenes de magnitud con respecto a los demás valores de $\boldsymbol{\alpha}$). Por medio de los valores de $\boldsymbol{\alpha}$ obtenidos se puede tener una idea de en cuanto se ha sobredimensionando *I* a priori. De esta forma se podrían volver a procesar los datos con un valor de *I* más pequeño. En otras ocasiones a la hora de analizar los resultados, se pueden obtener ciertas clases "repetidas"

debido a la coincidencia de los parámetros $\boldsymbol{\beta}_i$. En estos casos se volvería a simular disminuyendo I hasta que no se obtengan clases con los parámetros $\boldsymbol{\beta}_i$ repetidos. Por otro lado existen diferentes modelos para estimar el número de componentes de la mezcla como por ejemplo el criterio de información de Akaike [6], el criterio BIC y AWE (Approximate Weight of Evidence) [7]. I se elegirá de forma diferente de acuerdo al problema a estudiar en cada caso (en algunos problemas será conocido y en otros se determinará de manera experimental, como se ha comentado anteriormente).

Como el algoritmo considerado es iterativo, la elección de los valores iniciales de $\boldsymbol{\theta}$ tiene una gran importancia en la velocidad de convergencia del algoritmo y en la posibilidad de converger a máximos locales de la función de verosimilitud.

Como valores iniciales de $\boldsymbol{\alpha}$ es habitual establecer la misma probabilidad para cada clase cumpliendo las restricciones *(1.6)* [8], [9]

$$\alpha_i = \frac{1}{I} \qquad (2.12)$$

Como valores iniciales de $\boldsymbol{\mu}$, es habitual la elección de I observaciones elegidas de forma aleatoria y utilizar esas observaciones como media inicial para las clases. Entre otras opciones, se encuentra la elección de I medias equi-espaciadas dentro del espacio geométrico de los datos [8], [9], [10].

Con respecto a los valores de $\boldsymbol{\Sigma}_i$, lo más común es utilizar para todas las clases la estima muestral de la matriz de covarianza de los datos [9], [10]. La inicialización de $\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$ de la forma comentada anteriormente se designará con el nombre de "inicialización aleatoria" y será la utilizada en los problemas y aplicaciones de este trabajo.

Otras opciones a la hora de elegir la inicialización son la utilización de algoritmos de clústering, como por ejemplo el algoritmo K-means, e inicializar el algoritmo EM con los valores obtenidos con este algoritmo [10], [11]. Debido a que el algoritmo EM no siempre converge al óptimo global, una opción es inicializar el algoritmo con diferentes inicializaciones aleatorias y quedarse con las estimas que proporcionen un valor mayor de verosimilitud *(1.11)*, [9], [12].

Puede ocurrir que en alguna iteración algunas de las matrices de covarianza obtenidas estén mal condicionadas o sean singulares (esto no permitirá evaluar *(2.11)* y *(1.4)*), esto puede deberse a que el número de observaciones no es suficiente y a que el número de dimensiones de los datos es relativamente alto. Este problema también puede aparecer en el caso de intentar ajustar los datos sobredimensionando I.

Para evitar este problema existen diferentes soluciones: pre-procesar los datos o adaptar el algoritmo para que todas las Gaussianas utilicen la misma matriz de covarianza [5], [12]. Otra opción es la regularización de las matrices de covarianza utilizando un parámetro de regularización. Este parámetro se añadiría en la diagonal de la matriz de covarianza que esté mal condicionada en cada iteración que sea necesario como se muestra a continuación

$$\boldsymbol{\Sigma}_i^{(n+1)'} = c \cdot \boldsymbol{I}_D + \boldsymbol{\Sigma}_i^{(n+1)} \qquad (2.13)$$

Donde $\boldsymbol{I}_D$ es la matriz identidad de tamaño DxD y c es el parámetro de regularización. Su valor se puede estimar a partir de los propios datos como explican Schäfer y Strimmer en [13]. En ocasiones, al sobredimensionar I puede ocurrir que alguna Gaussiana quede centrada en un "outlaier" y con las iteraciones los valores de la diagonal de $\boldsymbol{\Sigma}_i$ converjan a cero [12]. En estas situaciones, una posible solución es reducir el valor de I. Otra solución adicional es intentar aplicar sobre los datos técnicas de reducción de dimensiones [12], [14].

En el algoritmo EM, como en cada iteración la verosimilitud *(1.11)* aumenta, es necesario utilizar un criterio de parada, un umbral para garantizar la convergencia. En los problemas y aplicaciones planteadas en este trabajo mientras no se diga lo contrario, el umbral utilizado será $1 \cdot 10^{-8}\%$, cuando la variación relativa de la verosimilitud en una iteración con respecto a la anterior es inferior a ese umbral, el algoritmo terminará.

2.2 Algoritmo K-means

El algoritmo K-means no tiene por objetivo resolver *(1.11)*, presenta un planteamiento diferente, resuelve un problema geométrico. Se trata de uno de los algoritmos de agrupamiento batch ampliamente utilizados para resolver problemas de clústering. Un clúster es un conjunto/grupo de datos (observaciones) que presentan propiedades comunes entre sí y diferentes a las de otros clústers. Se define clústering a la tarea de agrupar los datos en grupos (clústers) de acuerdo a algún/os criterios. El clústering puede ser utilizado para descubrir estructuras en los datos, como se distribuyen entre sí, sin una interpretación o explicación previa. Es utilizado en diversos campos como el aprendizaje máquina, reconocimiento de patrones, análisis de imágenes, recuperación de información y bioinformática.

En el algoritmo K-means se fija a priori un número de clústers K [15], [16], [17]. Para cada clúster se define el centroide como el punto medio (la media) del clúster $\boldsymbol{\mu}_i \epsilon \mathbb{R}^D \; i\epsilon\{1, \dots, K\}$.

Dado un conjunto de observaciones $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_J\}$, el algoritmo K-means tiene por objetivo agrupar las J observaciones en $K \leq J$ grupos en los que cada observación pertenece al centroide del clúster más cercano

$$\boldsymbol{C} = \{\boldsymbol{C}_1, \boldsymbol{C}_2, \dots, \boldsymbol{C}_K\} \qquad (2.14)$$

Donde $\boldsymbol{C}_i \; i\epsilon\{1, \dots, K\}$ es el conjunto de observaciones pertenecientes al clúster i-ésimo. Para minimizar la suma de los cuadrados en el clúster (la distancia euclídea, distancia de cada observación al centroide de cada clúster) se utiliza una función objetivo $\mathcal{J}$

$$\mathcal{J} = argmin_s \sum_{i=1}^{K} \sum_{\boldsymbol{y}_j \in \boldsymbol{C}_i} \left\| \boldsymbol{y}_j - \boldsymbol{\mu}_i \right\|^2 \qquad (2.15)$$

Dado un conjunto inicial de centroides $\boldsymbol{\mu}^{(0)} = (\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \dots, \boldsymbol{\mu}_K^{(0)})$, el algoritmo K-means alterna dos pasos que se repiten iterativamente hasta que se cumple un criterio de convergencia.

Paso 0 inicialización: se eligen de forma aleatoria K observaciones y se utilizan como centroides iniciales de los clústers. En la *Figura 2.1* se muestra como ejemplo un grupo de observaciones representadas en gris, y se han representado en rojo, verde y azul los $K = 3$ centroides elegidos de forma aleatoria como inicialización.

Paso 1 asignación: se asigna cada observación a un clúster de forma que la suma de los cuadrados del clúster sea la menor

$$\boldsymbol{C}_i^{(n)} = \left\{ \boldsymbol{y}_j : \left\| y_j - \boldsymbol{\mu}_i^{(n)} \right\|^2 \leq \left\| y_j - \boldsymbol{\mu}_z^{(n)} \right\|^2 \forall_z, 1 \leq z \leq K \right\} \qquad (2.16)$$

En cada iteración n cada observación $\boldsymbol{y}_j$ se asigna a un único cluster $\boldsymbol{C}_i^{(n)}$ (Como ejemplo del paso de asignación se muestra la *Figura 2.2*, se ha realizado la asignación de las observaciones al centroide más cercano). Se calcula la distancia euclídea de cada observación a cada centroide y cada observación será asignada al clúster $\boldsymbol{C}_i$ más cercano (distancia euclídea menor).

Paso 2 actualización: Se recalculan las medias de los clústers teniendo en cuenta las observaciones asignadas a cada clúster por medio de

$$\boldsymbol{\mu}_i^{(n+1)} = \frac{1}{|\boldsymbol{C}_i^{(n)}|} \sum_{\boldsymbol{y}_j \in \boldsymbol{C}_i^{(n)}} \boldsymbol{y}_j \qquad (2.17)$$

Siendo $|\boldsymbol{C}_i^{(n)}|$ el número de observaciones pertenecientes al clúster i-ésimo (En la *Figura 2.3* se muestra un ejemplo del paso de actualización, se recalculan las medias de los clústers). Una vez obtenidos esos nuevos centroides se realiza de nuevo la asignación de las observaciones a los centroides más cercanos. Los pasos 1 y 2 se repiten iterativamente produciendo el cambio de la posición de los centroides en cada iteración (en la *Figura 2.4* se muestra el resultado final tras haber repetido los pasos 1 y 2 hasta que no se han producido cambios de posición de los centroides). El algoritmo se dice que ha terminado o convergido cuando en una iteración no se producen nuevas asignaciones y por tanto $\boldsymbol{\mu}_i^{(n+1)} = \boldsymbol{\mu}_i^{(n)}$. El algoritmo converge a un mínimo aunque no siempre será el mínimo global.

Nótese que el número de clústers K es el equivalente al número de clases I en el algoritmo EM, por tanto de aquí en adelante se utilizará K o I indistintamente.

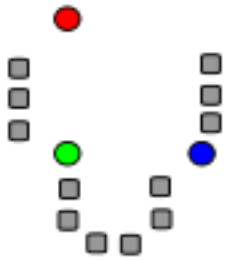

Figura 2.1: K-means, Paso 0 inicialización

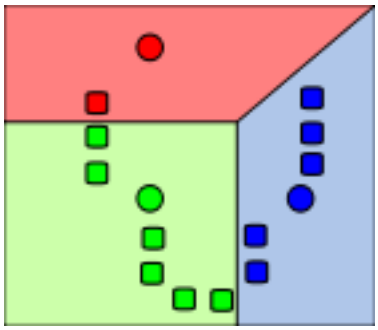

Figura 2.2: K-means, Paso 1 asignación

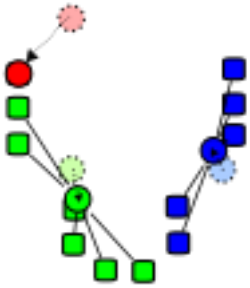

Figura 2.3: K-means,Paso 2 actualización

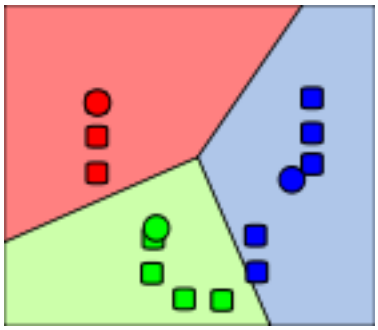

Figura 2.4: K-means, Paso 3 iteración (repetición pasos 1 y 2)

2.2.1 El problema de la inicialización y la determinación del número de clústers

Habitualmente se utilizan dos métodos de inicialización, el de Forgy y el de Partición Aleatoria [15], [16]. El método de Forgy es el más habitual y se corresponde con la inicialización explicada en el "Paso 0" del apartado anterior.

La convergencia al óptimo global dependerá de la posición de los centroides iniciales asignados en el paso de inicialización. Habitualmente suele ejecutarse el algoritmo varias veces con diferentes centroides iniciales y quedándose con las estimas que minimicen en mayor medida *(2.15)*.

En ocasiones a la hora de clusterizar los datos, el número de clústers I no es conocido a priori o no es obvio. En estos casos existen diferentes técnicas para obtener el I óptimo, una opción es la utilización del criterio de información de Akaike [6]. Otra posible opción es evaluar el algoritmo K-means para varios valores de I y posteriormente analizar la silueta (Silhouette) y quedarse con el valor óptimo como explica Peter en [18]. Esta técnica sirve para obtener una idea de cómo de bien cada observación se encuentra dentro de su clúster. Sin embargo no hay un método que siempre funcione mejor para cualquier conjunto de datos. Dependiendo del problema, el método a utilizar para encontrar el valor de I óptimo será determinado de forma experimental.

2.3 Análisis comparativo algoritmos batch

La explicación del algoritmo K-means en el apartado anterior podría dar la impresión de que no encaja mucho con el planteamiento del trabajo y con el modelo de mezclas Gaussianas. Sin embargo, el algoritmo EM aplicado al modelo de mezclas Gaussianas puede verse como un método de clústering probabilístico soft que asigna a cada observación la probabilidad de pertenecer a un clúster por medio de

$$N(z_{ij} = 1|\boldsymbol{y}_j, \boldsymbol{\beta}_i) = \frac{N(\boldsymbol{y}_j|C = i, \boldsymbol{\beta}_i) \cdot \alpha_i}{\sum_{i=1}^{I} \alpha_i \, N(\boldsymbol{y}_j|C = i, \boldsymbol{\beta}_i)} \qquad (2.18)$$

Además, sobre *(2.18)* es posible tomar una decisión hard clasificando las observaciones de forma determinista, asignándolas a la clase que mayor probabilidad obtiene.

En contraposición, el algoritmo K-means es un método de hard clústering ya que asigna cada observación a un clúster de forma determinista. En el caso del algoritmo K-means al utilizar métricas de distancias para asignar las observaciones a los clústers, no puede distinguir clústers que compartan la misma media o clústers superpuestos, es decir, clústers no separados geométricamente. El modelo de mezclas Gaussianas aplicado al algoritmo EM por el contrario permite distinguir clústers superpuestos. Esta es una de las grandes diferencias entre ambos métodos. Para ilustrar mejor la definición de clústers superpuestos se muestra un ejemplo a continuación

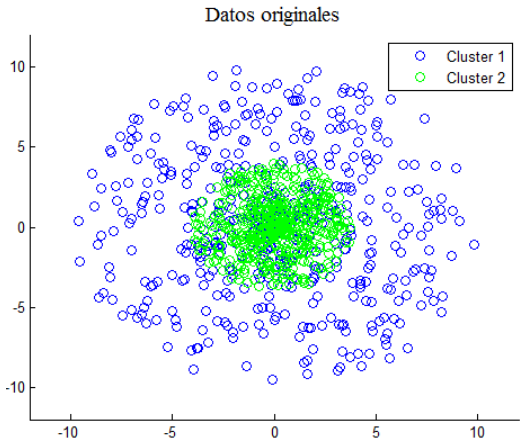

Figura 2.5: Ejemplo clústers superpuestos

En el ejemplo de la figura anterior se tiene un conjunto de datos en el que $D = 2$, $I = 2$ y se distingue un clúster con las observaciones más dispersas y otro con observaciones más concentradas. En el *Ejemplo 1, Apartado 4.3* se muestran los resultados obtenidos con ambos algoritmos a modo de comparación.

Una limitación clave del algoritmo K-means es su modelo de formación de clústers. Para un problema determinado en el que las observaciones no pertenezcan al centroide del clúster más cercano, no sería el algoritmo óptimo a utilizar. El algoritmo EM aplicado al modelo de mezclas Gaussianas permite distinguir clústers con formas Gaussianas y estima *(1.2)*, el algoritmo K-means únicamente estima los centroides de los clústers. El algoritmo K-means no es muy útil en casos en los que la I no es conocida a priori, cuando los observaciones no son linealmente separables o cuando existe ruido (observaciones muy alejadas de los centroides de los clústers y

dispersas). En estos casos el algoritmo EM funciona notablemente mejor. En el *Apartado 4.1* en la aplicación de ruido impulsivo se mostrarán estos problemas.

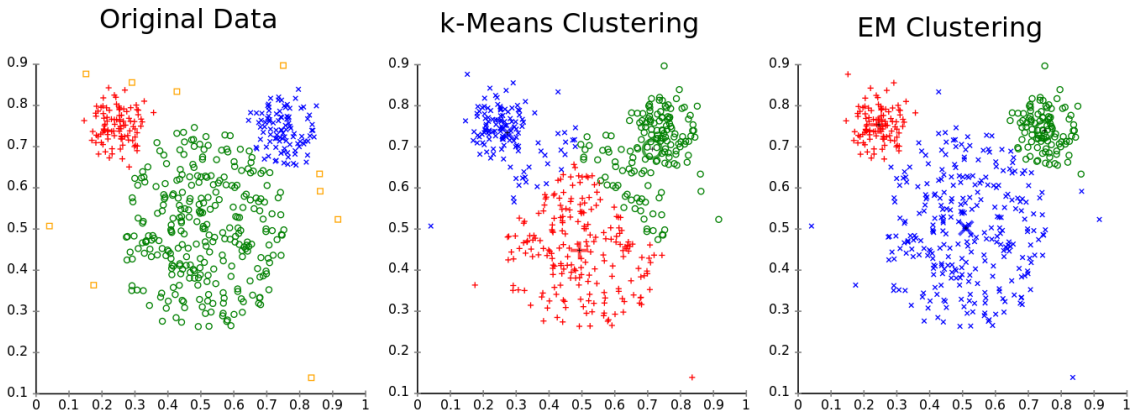

Figura 2.6: Comparación clustering K-means vs EM data set "mouse"

En la *Figura 2.6* se muestra la ventaja del algoritmo EM ya que se beneficia de la forma Gaussiana de los datos mientras que en el caso del algoritmo K-means, tiende a formar grupos con tamaños parecidos lo que lleva a obtener malos resultados.

El algoritmo EM aplicado al modelo de mezclas Gaussianas no es conveniente aplicarlo sobre datos discretos ya que se obtendrán matrices de covarianza mal condicionadas. Para los casos en los que las bases de datos toman valores discretos sin embargo existen ciertas técnicas para pre-procesarlas y evitar este tipo de problemas [12]. Por otro lado el algoritmo EM no es útil en los casos en los que no hay variables ocultas. No es la mejor elección a la hora de trabajar con problemas en los que hay muchos mínimos locales ya que tarda en converger y su convergencia es muy sensible a la inicialización utilizada en estos casos. El algoritmo EM aplicado al modelo de mezclas Gaussianas no es conveniente utilizarlo cuando los datos no se han generado por medio de una mezcla de Gaussianas como por ejemplo

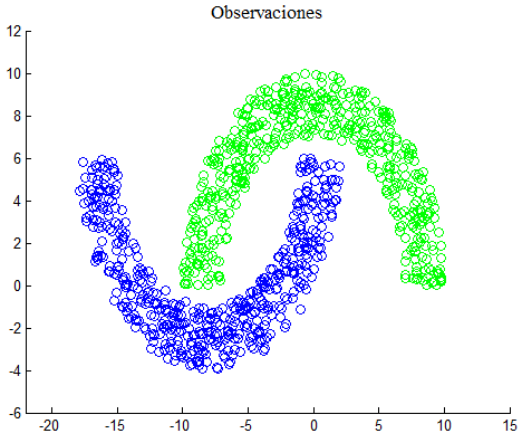

Ilustración 2.1: Ejemplo de datos no generados por medio de una mezcla Gaussiana

En la *Figura 2.7* se muestra un ejemplo típico en el que ninguno de los dos algoritmos obtiene buenos resultados. En el *Apartado 4.3* se mostarán varios ejemplos incluyendo el anterior para ver los resultados de los algoritmos de una forma más representativa.

En lo referente a la convergencia de los algoritmos, en la versión estándar del algoritmo K-means, el algoritmo termina cuando no cambian las medias de los clústers con respecto a la

iteración anterior. En el algoritmo EM para garantizar la convergencia se utiliza un umbral de parada.

Capítulo 3: Algoritmos online

Los algoritmos online a diferencia de los algoritmos bloque procesan un dato o subconjunto de datos en cada iteración del algoritmo. Se suele utilizar en casos en los que los datos son disponibles de forma secuencial. En cada iteración del algoritmo se obtiene una nueva estima de los parámetros a optimizar. Por tanto, la diferencia principal entre un algoritmo bloque y uno online es que en el segundo caso la actualización de las estimas se produce después de la llegada de cada nueva observación o subconjunto de observaciones mientras que en los algoritmos bloque se actualizan cuando se han procesado todas las observaciones [20], [21].

En alguna aplicación concreta en la que los datos son disponibles de forma secuencial y no pueden ser almacenados, se debe utilizar un algoritmo online.

3.1 Descenso por gradiente estocástico (SGD)

El algoritmo SGD es un método iterativo de optimización de funciones con la siguiente forma

$$Q(\boldsymbol{\theta}) = \sum_{j=1}^{J} q_j(\boldsymbol{\theta}) \qquad (3.1)$$

Donde cada función $q_j(\boldsymbol{\theta})$ es una función diferenciable y $Q(\boldsymbol{\theta})$ es la función objetivo ó función de coste a optimizar. Para minimizar $Q(\boldsymbol{\theta})$ la iteración j-ésima del algoritmo es

$$\widehat{\boldsymbol{\theta}}^{(j)} = \widehat{\boldsymbol{\theta}}^{(j-1)} - \gamma^{(j)} \nabla_{\boldsymbol{\theta}} q_j\left(\widehat{\boldsymbol{\theta}}^{(j-1)}\right), \qquad 0 < \gamma^{(j)} \leq 1 \qquad (3.2)$$

Dónde $\gamma^{(j)}$ es la constante de paso ó constante de aprendizaje. En el caso de buscar la maximización de la función de coste, la expresión *(3.2)* se modificaría cambiando el signo negativo delante de la constante de paso por un signo positivo.

El algoritmo se inicializa partiendo un vector inicial de parámetros $\widehat{\boldsymbol{\theta}}^{(0)}$ y unos valores de $\gamma^{(j)}$ que deben ser determinados a priori.

3.1.1 Algoritmo SGD aplicado a la estimación de los parámetros del modelo de mezclas Gaussianas

A continuación se detalla cómo estimar los parámetros de una mezcla de Gaussianas por medio del algoritmo SGD para el caso de $D = 1$. En el modelo de mezclas Gaussianas para encontrar

$\boldsymbol{\theta}$ se buscan los parámetros que maximizan el logaritmo de la función verosimilitud *(1.11)*. $\log L(\boldsymbol{\theta})$ puede verse como una suma de funciones diferenciables que dependen del conjunto de parámetros a estimar $\boldsymbol{\theta}$. Esto permite la aplicación del algoritmo SGD.

Calculando el logaritmo de la función verosimilitud se obtiene

$$Q(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \sum_{j=1}^{J} \log \sum_{i=1}^{I} \alpha_i N(y_j|\boldsymbol{\beta}_i) = \sum_{j=1}^{J} q_j(\boldsymbol{\theta}) \qquad (3.3)$$

Siendo q_j

$$q_j(\boldsymbol{\theta}) = \log\left(\sum_{i=1}^{I} \alpha_i N(y_j|\boldsymbol{\beta}_i)\right) = \log p(y_j|\boldsymbol{\theta}) \qquad (3.4)$$

Y su gradiente,

$$\nabla q_j(\boldsymbol{\theta}) = \frac{1}{p(y_j|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} p(y_j|\boldsymbol{\theta}) \qquad (3.5)$$

Donde *j* hace referencia al índice de las observaciones. Se considera una iteración al procesado de una única observación. En la aplicación directa del algoritmo pueden obtenerse estimas que no cumplan las restricciones del modelo de mezclas Gaussianas (*Apartado 1.2.1*). Para resolver este problema se plantean unas transformaciones. Para cumplir la primera restricción *(1.6)* se ha utilizado la siguiente transformación de parámetros

$$\alpha_i = \frac{e^{w_i}}{\sum_{s=1}^{I} e^{w_s}} \qquad (3.6)$$

y para que se cumpla la segunda restricción *(1.7)* se propone la siguiente transformación

$$\sigma_k^2 = e^{-z_k} \qquad (3.7)$$

El nuevo conjunto de parámetros a estimar ahora es

$$\boldsymbol{\theta} = \{\omega_1, \dots, \omega_I, \mu_1, \dots, \mu_I, z_1, \dots, z_I\} \qquad (3.8)$$

En cada iteración se actualizarán los valores de $\boldsymbol{\theta}$ por medio de

$$\begin{aligned}
\widehat{\boldsymbol{\omega}}^{(j)} &= \widehat{\boldsymbol{\omega}}^{(j-1)} + \gamma_{\boldsymbol{\omega}}^{(j)} \nabla_{\boldsymbol{\omega}} q_j\left(\widehat{\boldsymbol{\omega}}^{(j-1)}\right), \qquad \widehat{\boldsymbol{\omega}} = (\widehat{\omega}_1, \dots, \widehat{\omega}_I)^T \\
\widehat{\boldsymbol{\mu}}^{(j)} &= \widehat{\boldsymbol{\mu}}^{(j-1)} + \gamma_{\boldsymbol{\mu}}^{(j)} \nabla_{\boldsymbol{\mu}} q_j\left(\widehat{\boldsymbol{\mu}}^{(j-1)}\right), \qquad \widehat{\boldsymbol{\mu}} = (\hat{\mu}_1, \dots, \hat{\mu}_I)^T \\
\hat{\boldsymbol{z}}^{(j)} &= \hat{\boldsymbol{z}}^{(j-1)} + \gamma_{\boldsymbol{z}}^{(j)} \nabla_{\boldsymbol{z}} q_j\left(\hat{\boldsymbol{z}}^{(j-1)}\right), \qquad \hat{\boldsymbol{z}} = (\hat{z}_1, \dots, \hat{z}_I)^T
\end{aligned} \qquad (3.9)$$

Y los gradientes respecto a los nuevos parámetros se calculan de la siguiente manera

$$\begin{aligned}
\frac{\partial p(y_j|\boldsymbol{\theta})}{\partial \omega_k} &= \sum_{i=1}^{I} \alpha_i N(y_j|C=i) \frac{\partial \alpha_i}{\partial \omega_k}, \\
\frac{\partial \alpha_i}{\partial \omega_k} &= \alpha_k (I_A(i=k) - \alpha_i), \quad k, i \in \{1, \dots, I\}
\end{aligned} \qquad (3.10)$$

Siendo I_A la función indicador

$$I_A(i=k)\begin{cases}1, & i=k\\ 0, & i\neq k\end{cases} \qquad (3.11)$$

Se obtiene

$$\frac{\partial p(y_j|\boldsymbol{\theta})}{\partial \omega_k} = \alpha_k\left(N(y_j|C=k) - p(y_j|\boldsymbol{\theta})\right) \qquad (3.12)$$

Teniendo en cuenta (3.7)

$$N(y_j|C=k) = \frac{1}{\sqrt{2\pi}} e^{\frac{z_k}{2}} \cdot \exp\left(-\frac{1}{2} e^{z_k}(y_j-\mu_k)^2\right) \qquad (3.8)$$

El gradiente respecto de z_k es

$$\frac{\partial N(y_j|C=k)}{\partial z_k} = \frac{1}{2} N(y_j|z_k)\left[1 - \frac{(y_j-\mu_k)^2}{\sigma_k^2}\right] \qquad (3.9)$$

Y por último el gradiente respecto de μ_k

$$\frac{\partial p(y_j|\boldsymbol{\theta})}{\partial \mu_k} = \alpha_k N(y_j|\boldsymbol{\beta}_k)\frac{(y_j-\mu_k)}{\sigma_k^2} \qquad (3.10)$$

Nótese que la constante de paso no tiene por qué tomar el mismo valor para cada conjunto de parámetros a optimizar. Los valores de $\gamma^{(j)}$ serán diferentes para cada problema y determinarán en qué medida se actualizarán las estimas de los parámetros. Si toman unos valores muy grandes, el segundo término de *(3.2)* tendrá un mayor peso en la expresión, por lo que en cada iteración se actualizará el valor de $\boldsymbol{\theta}_j$ en mayor medida. Si por el contrario los valores de $\gamma^{(j)}$ son muy pequeños, afectarán en menor medida en las actualizaciones de $\boldsymbol{\theta}_j$.

Para garantizar la convergencia, los valores de la constante de aprendizaje se irán disminuyendo en cada iteración de forma que

$$\lim_{j\to\infty} \gamma^{(j)} = 0 \qquad (3.11)$$

Para ello es habitual disminuir γ_j en cada iteración de la siguiente forma

$$\gamma^{(j)} = \frac{1}{j^m}, \quad j=1,\dots,J, \quad 0\leq m\leq 1 \qquad (3.12)$$

Siendo m una constante a determinar y j representa el índice de la iteración. Por tanto para un número de iteraciones infinito se cumple *(3.16).* Lo mostrado en este apartado hace referencia al caso de $D=1$, es posible su generalización a D dimensiones sin embargo las matrices de

covarianza serían diagonales. En una línea futura de este trabajo se buscaría una transformación para las matrices de covarianza.

3.1.2 Algoritmo SGD mini-batch

Este algoritmo es una variante del algoritmo SGD. La diferencia es que se procesa en cada iteración un bloque de b observaciones en lugar de una observación. El parámetro b es conocido como tamaño del batch [24], [25]. Se calculan los gradientes para todo todas las observaciones de un mismo bloque y se promedia el resultado obtenido. Con este resultado se procede a la actualización del parámetro a estimar de la siguiente forma

$$\widehat{\boldsymbol{\theta}}^{(k)} = \widehat{\boldsymbol{\theta}}^{(k-1)} + \frac{1}{b}\gamma^{(k)} \sum_{j=(k-1)b+1}^{k\cdot b} \nabla_{\boldsymbol{\theta}} q_j(\widehat{\boldsymbol{\theta}}^{(k-1)}) \qquad (3.13)$$

El tamaño del bloque adecuado es un parámetro a determinar de forma experimental, típicamente $2 \leq b \leq 100$ [24]. Una ventaja a destacar del algoritmo SGD mini-batch es que puede ser más rápido que el algoritmo SGD debido a que se puede implementar de forma vectorizada y utilizar procesadores en paralelo para realizar los cálculos. Nótese que para $b = 1$ el problema se reduce al caso del algoritmo SGD convencional y para $b = J$ al Descenso por Gradiente [26].

3.1.3 El problema de la inicialización y la convergencia

Los algoritmos SGD y SGD mini-batch son sensibles a la inicialización como pasa con los demás algoritmos vistos anteriormente, el número de Gaussianas o clases de partida habitualmente es desconocido y se deberá recurrir a la utilización de alguna técnica como las comentadas en el *Apartado 2.1.1*. Los algoritmos online requieren habitualmente un elevado número de observaciones para obtener buenos resultados. Los problemas principales del algoritmo SGD y del SGD mini-batch, son encontrar el número de clases I para cada problema y cómo ajustar $\gamma^{(j)}$ para garantizar la convergencia. Para comprobar la convergencia y el valor adecuado de la constante de aprendizaje lo habitual es representar la evolución de las estimas de los parámetros en función del número de observaciones procesadas como se muestra en un ejemplo a continuación

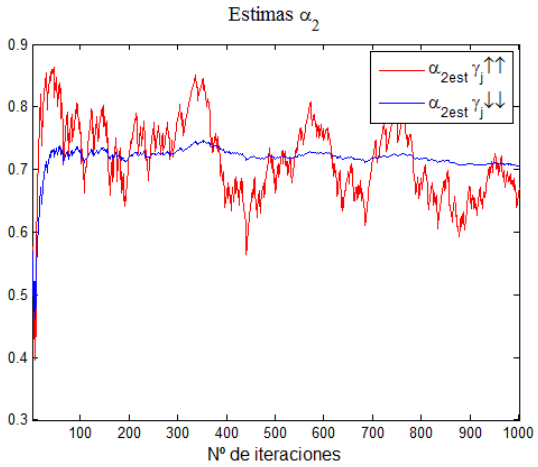

Figura 3.1: Curva de convergencia estima de α_2, algoritmo SGD, Ap. Ruido impulsivo, I=2

La figura anterior ha sido tomada de los resultados obtenidos en el *Apartado 4.1*. En la *Figura 3.1* se ha representado la evolución de la estima de un parámetro, concretamente α_2, para dos valores distintos de $\gamma^{(j)}$. En rojo se representa la evolución de la estima utilizando una constante de aprendizaje mayor que la utilizada en la representada en la curva azul. En la curva representada en rojo se muestra una gran variabilidad en las estimas de α_2 (cambios abruptos en las estimas del parámetro), esto es debido a que para este problema en concreto se ha elegido un valor de $\gamma^{(j)}$ muy elevado, lo que conlleva a una notable variación de las estimas en cada iteración. En azul se ha representado la evolución del mismo parámetro habiendo reducido $\gamma^{(j)}$, en este caso se muestra cómo la estima de α_2 converge de forma más suave y presenta una menor variabilidad en las estimas. En el caso de que la evolución de las estimas no converja se debería reducir la constante de aprendizaje o realizar lo siguiente que se explica a continuación.

En ciertos problemas se disponen de pocas observaciones y la evolución de las estimas de los parámetros puede no llegar a converger. En estos casos puede resultar interesante la opción de dar varias pasadas. Una pasada ($n = 1$) consta en la aplicación del algoritmo SGD con una ordenación aleatoria de los datos. Al dar varias pasadas, el algoritmo procesa sucesivas veces los datos en diferente orden. En este caso es necesario almacenar los datos en memoria y se producirá el aumento de las iteraciones (las iteraciones totales serán igual al número de pasadas por el nº de iteraciones en el caso de dar una sola pasada $nº\ iteraciones_{totales} = n * J$). En el algoritmo SGD mini-batch el planteamiento es el mismo.

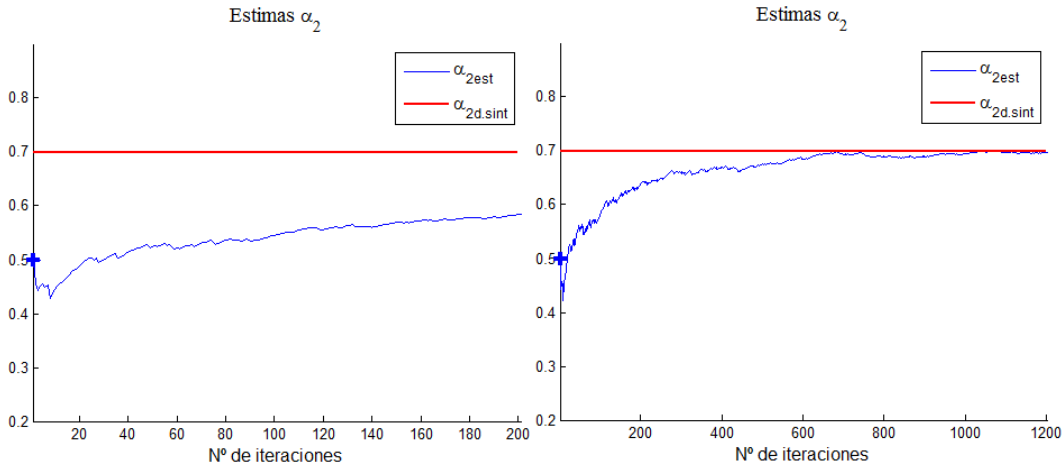

Figura 3.2: Curva de convergencia estima de α_2, algoritmo SGD n=1 vs n=6, Ap. Ruido impulsivo

Las figuras anteriores han sido tomadas de otro ejemplo de la aplicación del *Apartado 4.1*. En rojo se ha representado el valor utilizado de α_2 para la generación de la base de datos utilizada en esta aplicación. Por medio de un aspa azul se ha representado la estima inicial de la que parte el algoritmo SGD y a continuación la evolución de las estimas en cada iteración. En la imagen de la izquierda se ha representado la evolución de las estimas de α_2 para $n = 1$ y en la figura de la derecha para $n = 6$. En ambos casos se han utilizado los mismos valores de $\gamma^{(j)}$. Como se muestra en la figura de la derecha, al dar varias pasadas, la mejora de los resultados es notable. Sin embargo, llega un momento en el que la evolución de las estimas de α_2 converge, en torno a las 800 iteraciones, esto quiere decir que no por dar un número muy elevado de pasadas se obtendrá una mejora significativa en los resultados. El número de pasadas óptimo dependerá de cada problema en concreto y se determinará por medio del análisis de la convergencia de las estimas de los parámetros. En el *Apartado 4.1* se explicará con más detalle mediante la aplicación de Ruido impulsivo.

En el algoritmo SGD mini-batch es necesario determinar el tamaño de bloque óptimo *b* de forma experimental [24] mediante el análisis de la convergencia de las estimas de los parámetros. Cuanto mayor es *b* mejor es la aproximación del gradiente a la función de coste en un punto determinado, sin embargo el número de iteraciones se reducen. A continuación se muestra por medio de un ejemplo

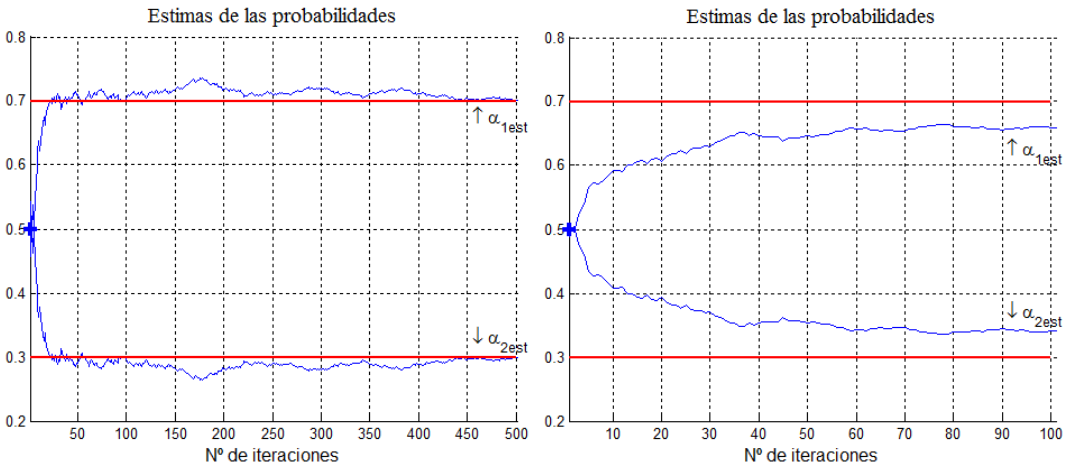

Figura 3.3: Curva de convergencia estima de $\boldsymbol{\alpha}$, algoritmo SGD b=2 vs b=10, Ap. Ruido impulsivo

Las figuras anteriores han sido tomadas de la aplicación del *Apartado 4.1*. Se ha representado la evolución de la curva de convergencia de la estima de los parámetros $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T$. En ambas figuras se ha utilizado la misma constante de aprendizaje de partida y como se muestra en los resultados, al utilizar un tamaño de bloque muy grande ($b = 10$ frente a $b = 2$), el número de iteraciones se reducen y los resultados empeoran. En este problema en concreto es preferible utilizar $b = 2$ como se muestra en la figura de la izquierda.

Capítulo 4: Resultados, experimentos y aplicaciones

4.1 Modelado de ruido impulsivo

El ruido impulsivo se caracteriza por la aparición esporádica de valores de ruido muy elevados. La fdp de ruido impulsivo puede escribirse como una mezcla de $I = 2$ gaussianas

$$p(y_j|\boldsymbol{\theta}) = \frac{\alpha_1}{\sqrt{2\pi\sigma_1}} e^{\left(-\frac{{y_j}^2}{2\sigma_1^2}\right)} + \frac{\alpha_2}{\sqrt{2\pi\sigma_2}} e^{\left(-\frac{{y_j}^2}{2\sigma_2^2}\right)} = \sum_{i=1}^{2} \alpha_i N(\boldsymbol{y}_j|C = i, \boldsymbol{\beta}_i) \qquad (4.1)$$

Donde $D = 1$, $\mu_i = 0\ \forall i$ y, α_2 controla la proporción de espurios o muestras de ruido de amplitud elevada $\sigma_2 \gg \sigma_1$. En los casos extremos $\alpha_2 = 0$ y $\alpha_2 = 1$ el ruido impulsivo se reduce a ruido Gaussiano. El conjunto de parámetros a estimar es

$$\boldsymbol{\theta} = [\alpha_1, \alpha_2, \beta_1, \beta_2]^T, \quad \beta_i = \sigma_i^2 \qquad (4.2)$$

La estimación de los parámetros del modelo de ruido impulsivo no es un problema sencillo debido a que las dos Gaussianas se encuentran centradas en el origen.

4.1.1 Base de datos ruido impulsivo

A continuación se muestran los resultados obtenidos por cada algoritmo utilizando una base de datos sintéticos. Para la obtención de la base de datos se han generado $J = 1000$ observaciones independientes de forma aleatoria por medio de una mezcla de Gaussianas con los siguientes parámetros

Parámetros	Clase 1	Clase 2
Probabilidades	$\alpha_1 = 0.3$	$\alpha_2 = 0.7$
Varianzas	$\sigma_1^2 = 100$	$\sigma_2^2 = 2$

Tabla 4.1: Parámetros utilizados para la generación base de datos Ap. Ruido impulsivo

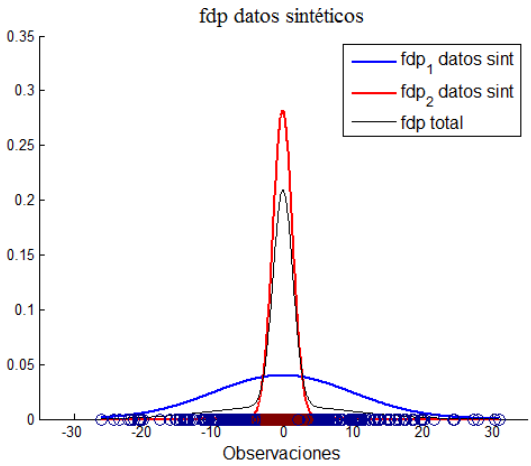

Figura 4.1: Datos sintéticos Ap. Ruido impulsivo

En azul se han representado las observaciones pertenecientes a la clase 1 y en rojo las pertenecientes a la clase 2. Si se realiza el histograma de los datos se obtiene

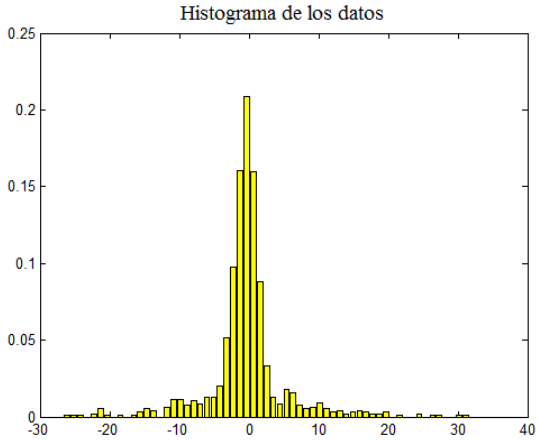

Figura 4.2: Ruido impulsivo, histograma

En la figura anterior se ha representado el histograma de los datos. A continuación se muestran los resultados que se obtienen con cada algoritmo.

4.1.1.1 Algoritmo EM

Se ha utilizado una inicialización aleatoria. Los resultados obtenidos son los siguientes

Parámetros	Clase 1	Clase 2	Error relativo
Probabilidades	$\alpha_1 = 0.3040$	$\alpha_2 = 0.6960$	0.0104 %
Varianzas	$\sigma_1^2 = 99.5889$	$\sigma_2^2 = 2.0935$	

Tabla 4.2: Resultados algoritmo EM, Ap. Ruido impulsivo

El error relativo es la variación en valor absoluto de la verosimilitud obtenida con los parámetros estimados y la que se obtendría con los parámetros que se han utilizado para generar los datos sintéticos (*Tabla 4.1*). A continuación se muestra la evolución de las estimas de los parámetros con el número de iteraciones

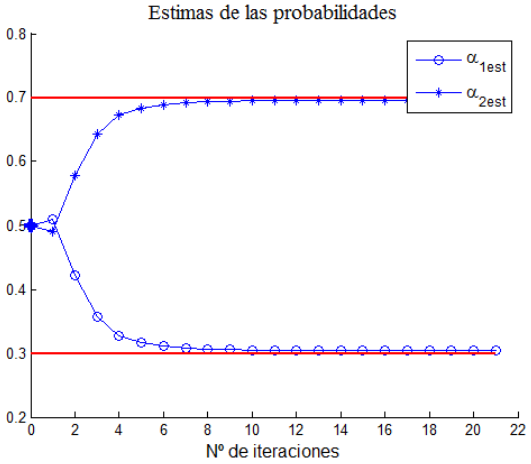

Figura 4.3: EM, Ruido impulsivo: Curva de convergencia en la estima de las probabilidades

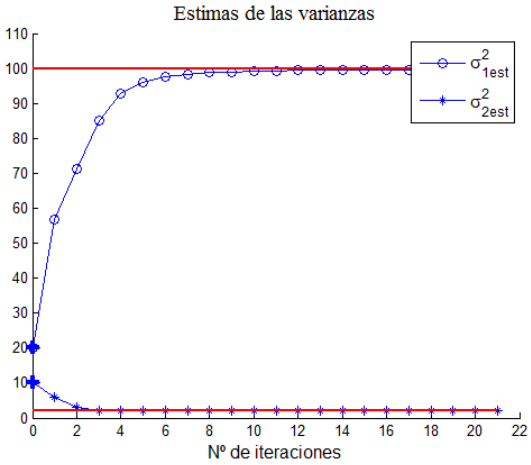

Figura 4.4: EM, Ruido impulsivo: Curva de convergencia en la estima de las varianzas

En las figuras anteriores, se representa en color azul la evolución en la estima de los parámetros a estimar y en rojo los valores con los que se han generado los datos sintéticos. Las aspas azules representan los valores iniciales de los parámetros.

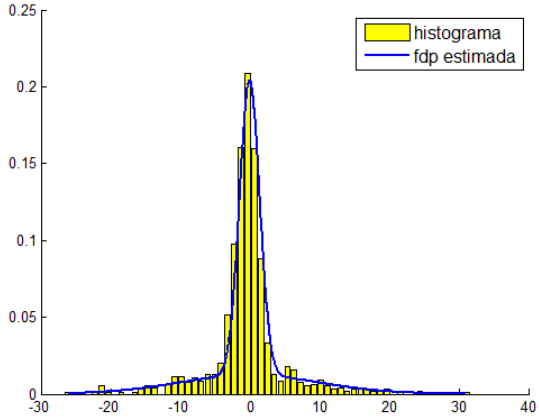

Figura 4.5: EM, Ruido impulsivo: Comparativa fdp estimada vs histograma

Como se muestra en la figura anterior, la fdp estimada (mezcla de dos Gaussianas) se ajusta al histograma de los datos observados.

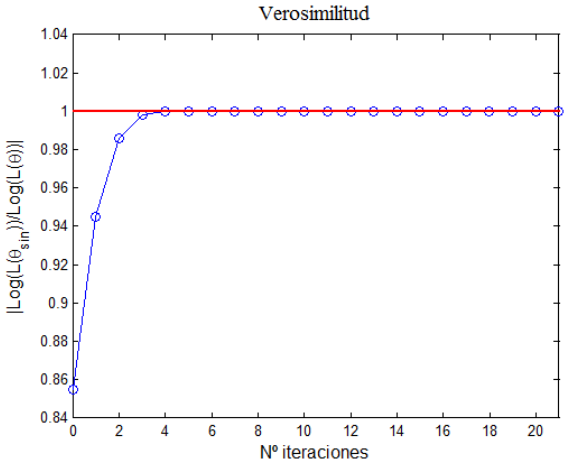

Figura 4.6: EM, Ruido impulsivo: Curva de convergencia de la función verosimilitud

En rojo se ha representado el valor de la función log-verosimilitud *(1.11)* que se obtendría con los parámetros que se han utilizado para la generación de los datos sintéticos normalizada a 1. En azul se ha representado la evolución de la verosimilitud con la estima de los parámetros en cada iteración.

A continuación se utilizan únicamente $J = 200$ observaciones de la base de datos anterior y se muestran los resultados obtenidos.

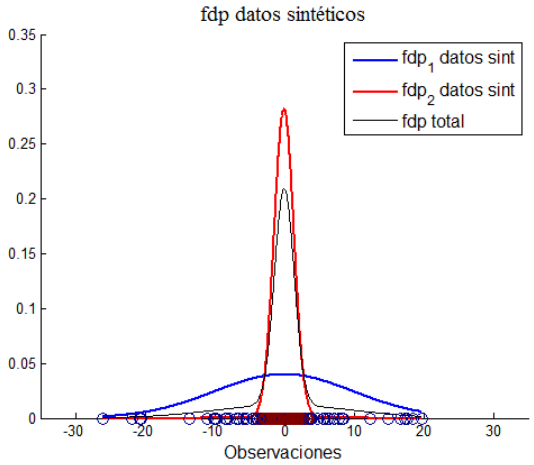

Figura 4.7: Datos sintéticos, J=200 Ap. Ruido impulsivo

En la *Figura 4.7* se puede apreciar que hay muy pocas observaciones de la clase 1 en comparación con la *Figura 4.1*.

Los resultados obtenidos son los siguientes

Parámetros	Clase 1	Clase 2	Error relativo
Probabilidades	$\alpha_1 = 0.2751$	$\alpha_2 = 0.7249$	0.1872 %
Varianzas	$\sigma_1^2 = 102.4356$	$\sigma_2^2 = 2.5773$	

Tabla 4.3: Resultados algoritmo EM, Ap. Ruido impulsivo J=200

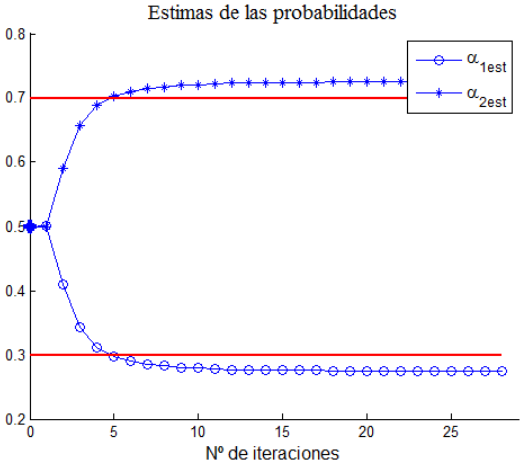

Figura 4.8: EM, Ruido impulsivo: Curva de convergencia en la estima de las probabilidades

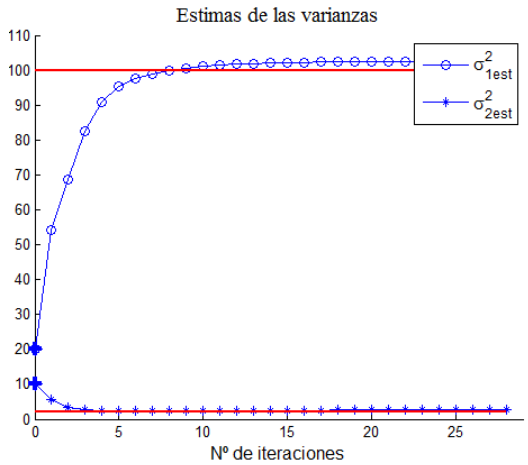

Figura 4.9: EM, Ruido impulsivo: Curva de convergencia en la estima de las varianzas

Los resultados obtenidos en las estimas de los parámetros convergen sin la necesidad de muchas iteraciones pese a la reducción de J (con respecto al problema original) y con un error relativo cercano al 0,2%. A continuación se muestra la fdp estimada sobre el histograma de los datos y la evolución de la verosimilitud.

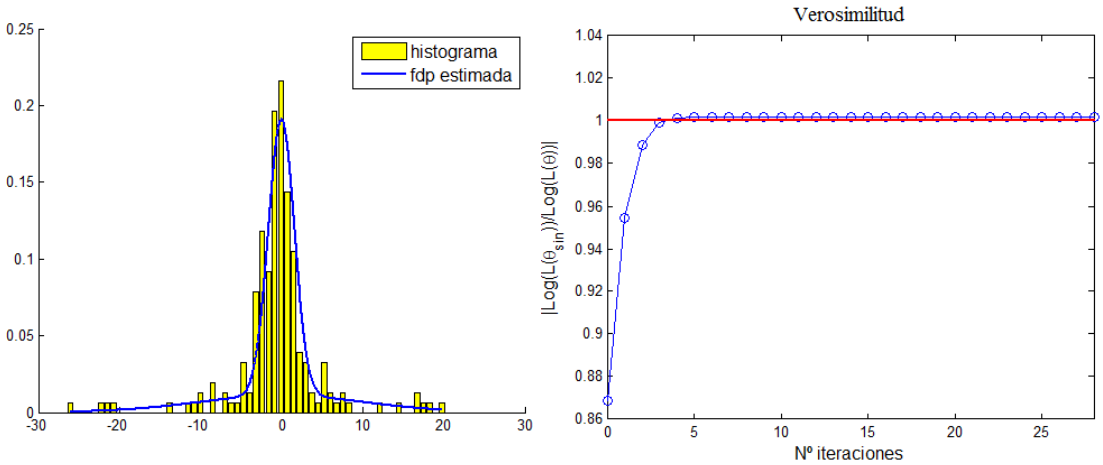

Figura 4.10: EM Ruido impulsivo: histograma vs fdp estimada y evolución verosimilitud

4.1.1.2 Algoritmo SGD

Se ha utilizado la misma inicialización que en el algoritmo EM. Los resultados obtenidos con el algoritmo SGD para la base de datos sintéticos inicial son los siguientes

Parámetros	Clase 1	Clase 2	Error relativo
Probabilidades	$\alpha_1 = 0.2940$	$\alpha_2 = 0.7060$	0.0543 %
Varianzas	$\sigma_1^2 = 98.0343$	$\sigma_2^2 = 2.3972$	

Tabla 4.4: Resultados algoritmo SGD, Ap. Ruido impulsivo

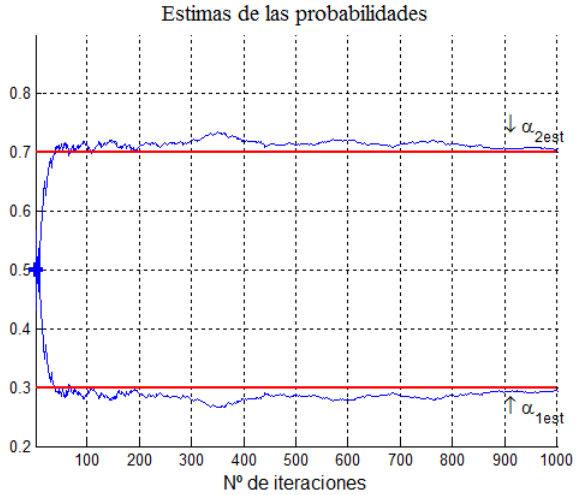

Figura 4.11: SGD, Ruido impulsivo: Curva de convergencia en la estima de las probabilidades

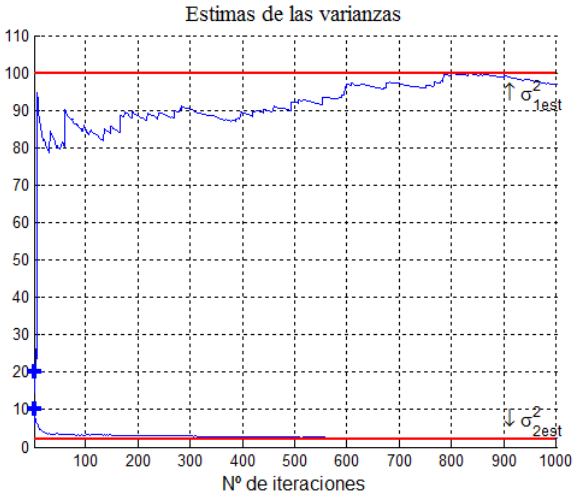

Figura 4.12: SGD, Ruido impulsivo: Curva de convergencia en la estima de las varianzas

A partir de 600 iteraciones las estimas convergen muy cerca de los valores utilizados para la generación de los datos sintéticos. El error relativo es superior al obtenido con el algoritmo EM, a pesar de ello, como se muestra a continuación la fdp estimada se ajusta al histograma de los datos

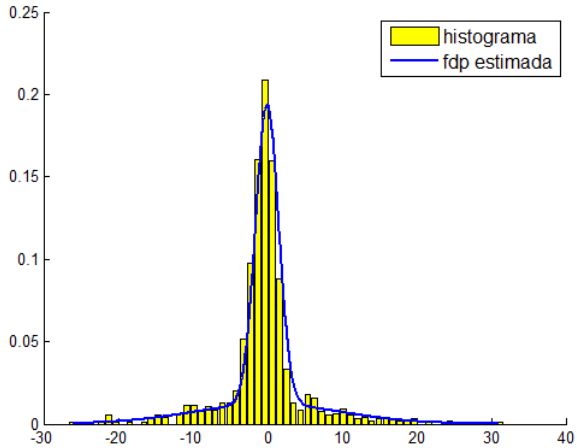

Figura 4.13: SGD, Ruido impulsivo: Comparativa fdp estimada vs histograma

Como se muestra en la figura anterior, la fdp estimada se ajusta bien al histograma de los datos.

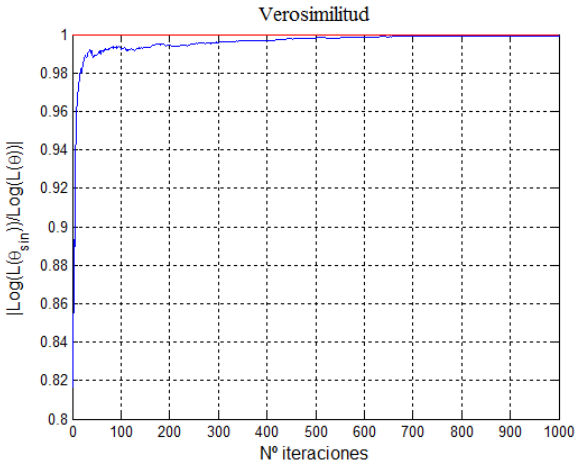

Figura 4.14: SGD, Ruido impulsivo: Curva de convergencia de la función verosimilitud

A continuación se muestran los resultados en el caso de utilizar menos observaciones $J = 200$, las mismas que se utilizaron en el algoritmo EM. Se obtienen los siguientes resultados

Parámetros	Clase 1	Clase 2	Error relativo
Probabilidades	$\alpha_1 = 0.3647$	$\alpha_2 = 0.6353$	3.2734 %
Varianzas	$\sigma_1^2 = 64.0994$	$\sigma_2^2 = 6.1922$	

Tabla 4.5: Resultados algoritmo SGD, Ap. Ruido impulsivo J=200

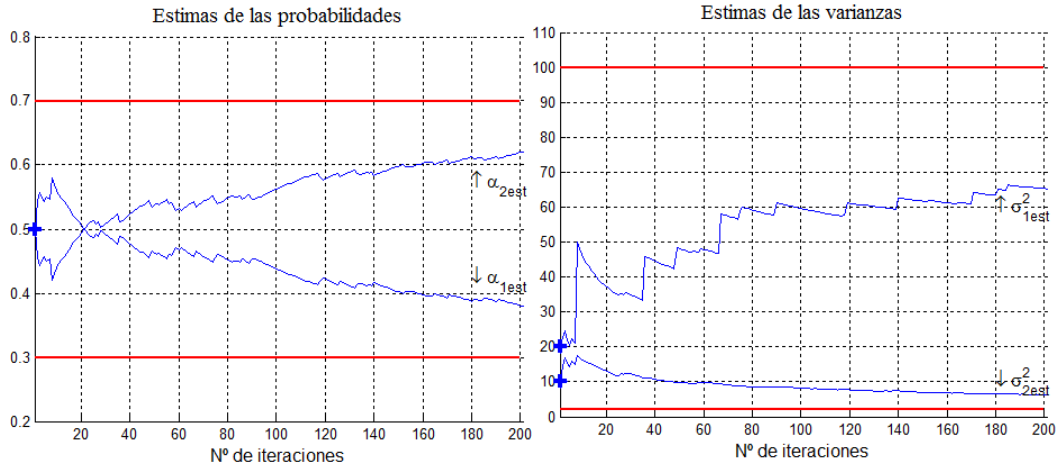

Figura 4.15: SGD, Ruido impulsivo: Curva de convergencia en la estima de los parámetros, J=200

A pesar de haber intentado encontrar el valor óptimo de la constante de aprendizaje, las estimas de los parámetros divergen debido al reducido número de observaciones para este problema.

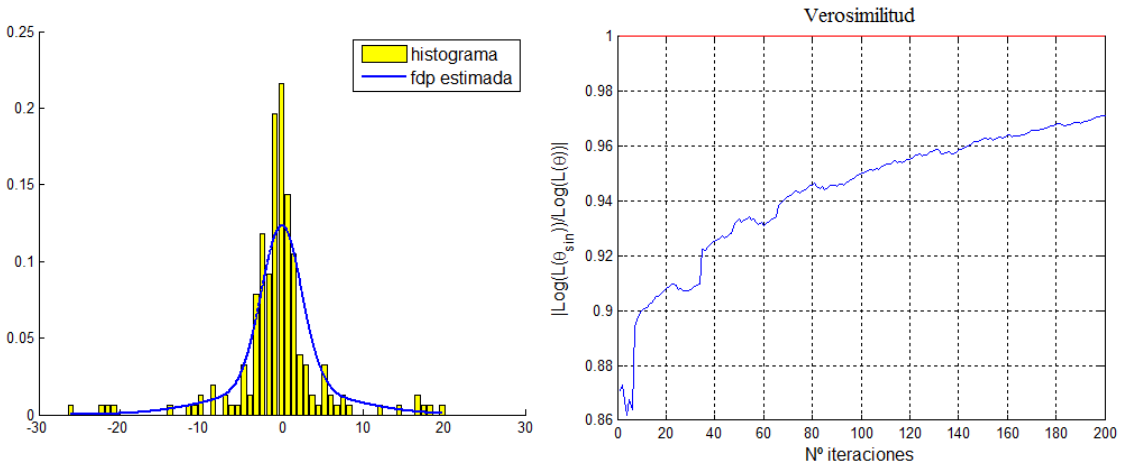

Figura 4.16: SGD, Ruido impulsivo: histograma vs fdp estimada y evolución verosimilitud, J=200

Los resultados de las estimas de los parámetros influyen en la fdp estimada y en la convergencia de la verosimilitud como se muestra en la figura superior.

En casos como estos en los que tenemos pocas observaciones puede resultar interesante aplicar varias pasadas a los datos como se ha explicado en el *Apartado 3.1.3*.

En el caso de realizar 10 pasadas se obtiene

Parámetros	Clase 1	Clase 2	Error relativo
Probabilidades	$\alpha_1 = 0.2730$	$\alpha_2 = 0.7270$	0.3814 %
Varianzas	$\sigma_1^2 = 99.9463$	$\sigma_2^2 = 2.6665$	

Tabla 4.6: Resultados algoritmo SGD, Ap. Ruido impulsivo J=200, 10 pasadas

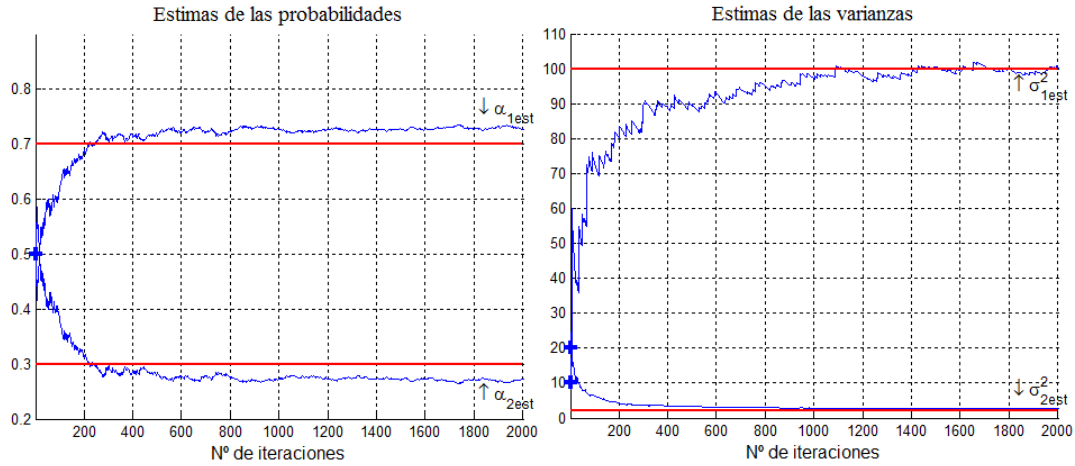

Figura 4.17: SGD, Ruido impulsivo: Curva de convergencia en la estima de los parámetros, J=200, 10 pasadas

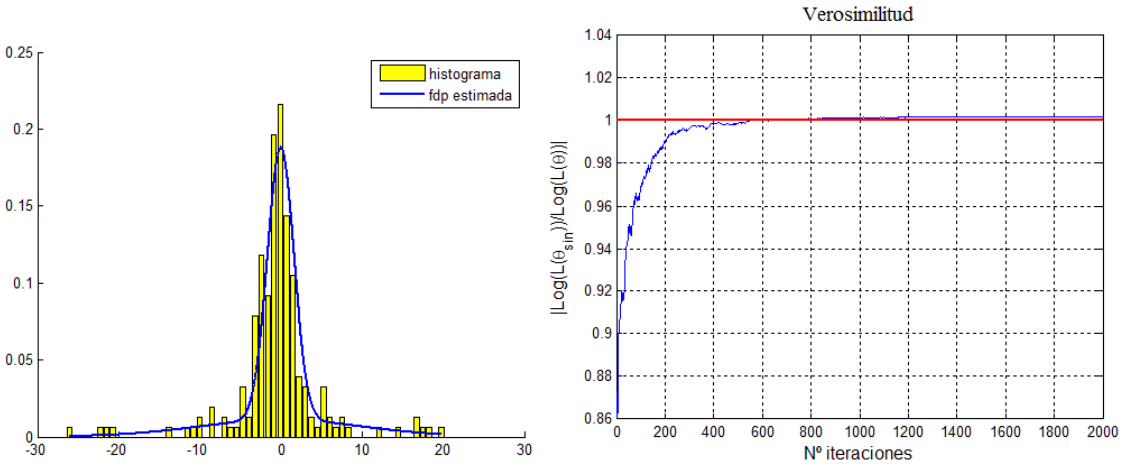

Figura 4.18: SGD, Ruido impulsivo: histograma vs fdp estimada y evolución verosimilitud, J=200, 10 pasadas

Como se muestra en los resultados, se obtiene una mejora sustancial en los resultados. A continuación se adjunta una tabla comparativa aplicando un número diferente de pasadas y los resultados obtenidos

Nº de pasadas	1	3	5	10	20
Error relativo	3.2734 %	0.9323 %	0.4518 %	0.3814 %	0.3859 %

Tabla 4.7: Error relativo algoritmo SGD, en función del número de pasadas

Se muestra que aunque se realicen muchas pasadas llega un momento en el que el error relativo converge a un valor (debido a la convergencia en la estima de los parámetros) y aunque se realicen más pasadas, no se obtiene una mejora notable en los resultados.

4.1.1.3 Algoritmo SGD mini-batch

En este caso es necesario determinar el tamaño óptimo del batch. Para la base de datos utilizada, se ha determinado de forma experimental que el valor de b óptimo es $b = 2$. Los resultados obtenidos son los siguientes

Parámetros	Clase 1	Clase 2	Error relativo
Probabilidades	$\alpha_1 = 0.2992$	$\alpha_2 = 0.7008$	0.0241 %
Varianzas	$\sigma_1^2 = 98.6851$	$\sigma_2^2 = 2.2303$	

Tabla 4.8: Resultados algoritmo SGD mini-batch, Ap. Ruido impulsivo

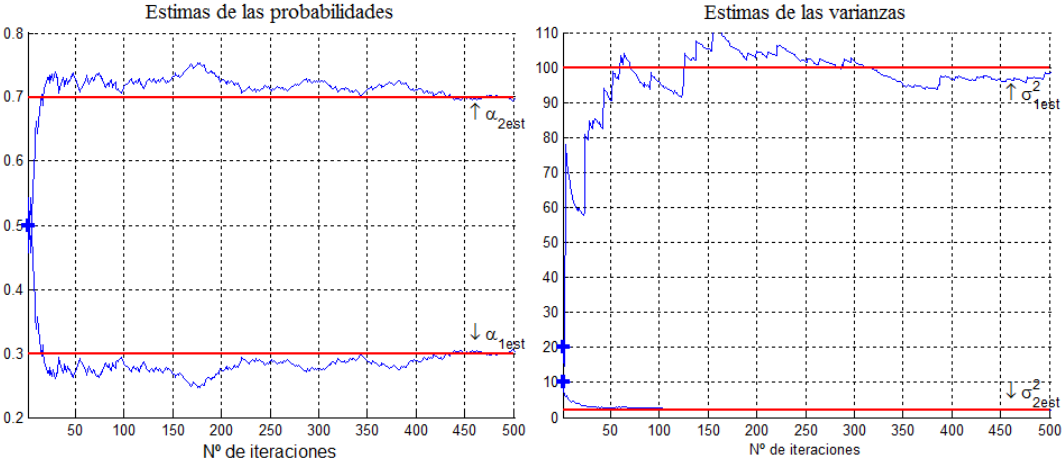

Figura 4.19: SGD mini-batch, Ruido impulsivo: Curva de convergencia en la estima de los parámetros

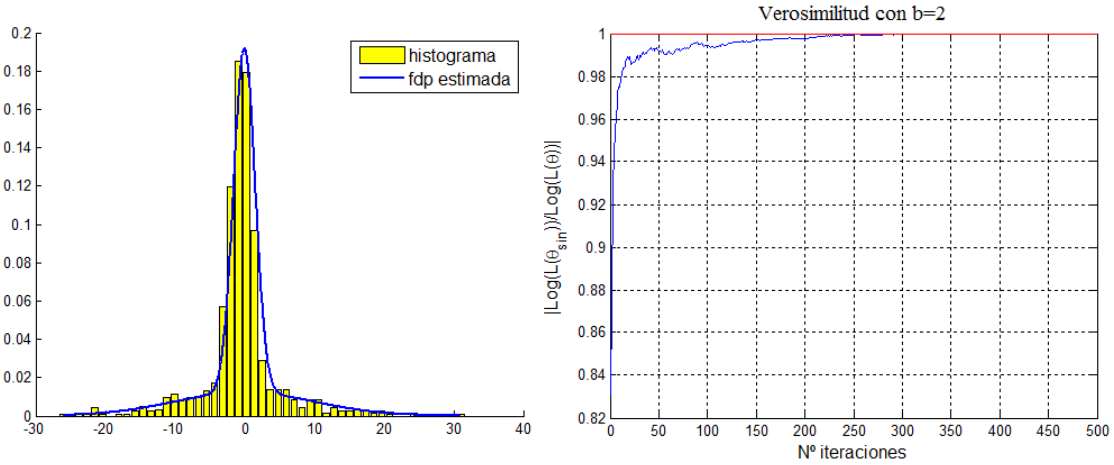

Figura 4.20: SGD, Ruido impulsivo: histograma vs fdp estimada y evolución verosimilitud

Los resultados mejoran en gran medida los obtenidos con el SGD, la fdp estimada se ajusta mejor y el error relativo es inferior. En el caso de utilizar diferentes tamaños de bloque a modo de comparación se adjuntan los resultados

Tamaño bloque	$b = 1$	$b = 2$	$b = 5$	$b = 10$
Error relativo	0.0543 %	0.0241 %	0.1706 %	3.5593 %

Tabla 4.9: Error relativo algoritmo SGD mini-batch, en función de b

La tabla anterior muestra que un tamaño de bloque mayor no implica mejores resultados, esto es debido a que al utilizar un tamaño de bloque mayor, el número de iteraciones se reduce. Al utilizar uno más pequeño, las actualizaciones de las estimas, no siempre van en la dirección del máximo pero en su conjunto poco a poco puede acabar convergiendo a un valor cercano al óptimo. Esto depende también en gran medida con el valor de γ_j ya que un valor cercano a 1 presentará mayor variabilidad en las estimas que uno cercano a 0.

En el caso de utilizar las $J = 200$ observaciones que se utilizaron para los algoritmos anteriores los resultados que se obtienen para b=2 son los siguientes

Parámetros	Clase 1	Clase 2	Error relativo
Probabilidades	$\alpha_1 = 0.3186$	$\alpha_2 = 0.6814$	1.2547 %
Varianzas	$\sigma_1^2 = 73.7704$	$\sigma_2^2 = 4.4426$	

Tabla 4.10: Resultados algoritmo SGD mini-batch, Ap. Ruido impulsivo J=200, b=2

Los resultados son algo mejores que los obtenidos con el algoritmo SGD, en el caso de utilizar tamaños de bloque mayor los resultados empeoran. En el caso de utilizar varias pasadas

	Error relativo %			
Nº pasadas	$b = 1$	$b = 2$	$b = 5$	$b = 10$
1	3.2734	1.2547	2.6568	3.5307
3	0.9323	0.5288	0.6237	1.2411
5	0.4518	0.3914	0.4726	0.5843
10	0.3814	0.2549	0.3968	0.4116

Tabla 4.11: Error relativo algoritmo SGD mini-batch, en función de b, J=200

Al realizar varias pasadas los resultados obtenidos con el algoritmo SGD mini-batch mejoran notablemente y superan los que se obtuvieron con el algoritmo SGD.

4.1.1.4 K-means

En este problema no tiene sentido aplicar el algoritmo K-means ya que los clústers son concéntricos (comparten la misma media) y están superpuestos. Si se fuerza a analizar dos clústers $k = 2$ los resultados que se obtienen son los siguientes

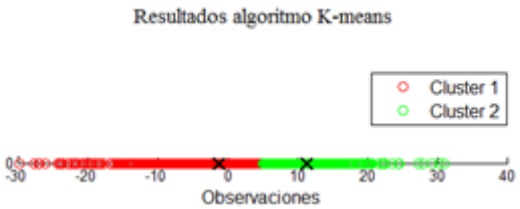

Figura 4.21: Resultados K-means, Ap. Ruido impulsivo

En la *Figura 4.21* se han representado las observaciones clusterizadas. El algoritmo K-means no es capaz de detectar clústers superpuestos y por tanto ha situado las medias en dos puntos separados una cierta distancia, las observaciones más cercanas a la primera media las ha asignado al clúster 1 y las demás al clúster 2. Las medias de los clústers se encuentran representadas por medio de una 'x' en negro.

4.2 Modelado de distribuciones de variables y vectores aleatorios

En esta aplicación se pretende modelar la fdp de una distribución arbitraria por medio de una mezcla de Gaussianas utilizando los algoritmos estudiados.

Ejemplo 1

Para este ejemplo se ha utilizado la tercera dimensión $d = 3$ de la base de datos *breast_cancer* [26] utilizada en otro experimento en el *Apartado 4.6* (para una explicación de la base de datos consultar este apartado). El histograma de los datos es el siguiente

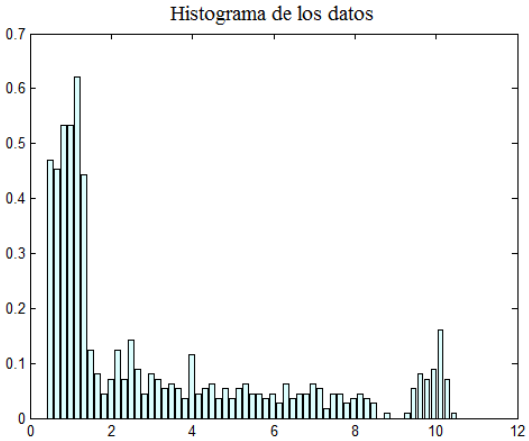

Figura 4.22: Histograma de los datos ejemplo 1

A continuación se intenta modelar la distribución de los datos anteriores $J = 680$ por medio del algoritmo EM y el algoritmo SGD (tanto la versión normal como la mini-batch) utilizando el modelo de mezclas Gaussianas. Se analizarán los resultados para distintos valores de I intentando encontrar el número de Gaussianas de la mezcla óptimo que mejor se ajuste al histograma de los datos. Para $I = 2$ se obtiene

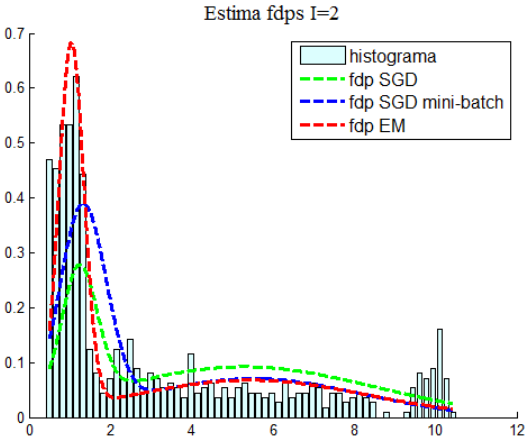

Figura 4.23: Ejemplo 1, análisis comparativo en el ajuste al histograma de los datos I=2

Los resultados pueden mejorarse en el caso de aumentar el número de Gaussianas de la mezcla como se muestra a continuación

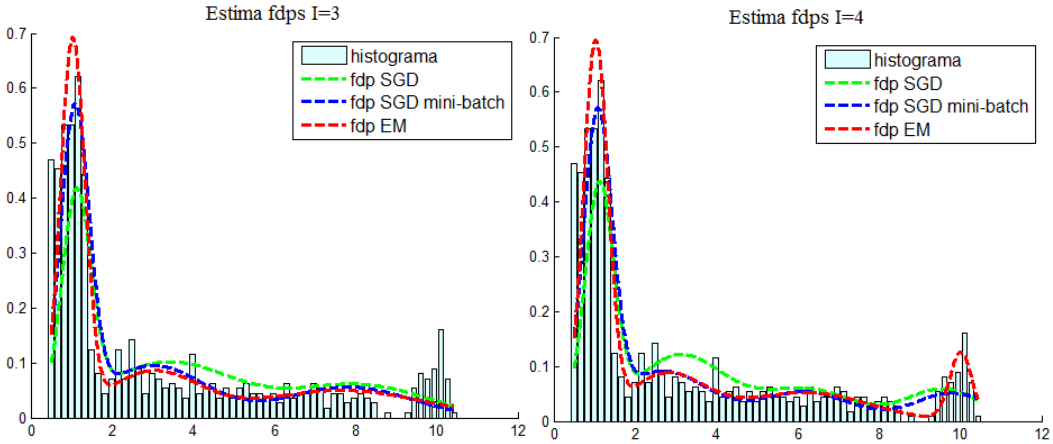

Figura 4.24: Ejemplo 1, análisis comparativo en el ajuste al histograma de los datos utilizando mezcla de 3 y 4 Gaussianas

En el caso de $I = 4$ la fdp estimada obtenida con el algoritmo EM se ajusta mejor al histograma de los datos en comparación con $I = 2$ *y* $I = 3$, sin embargo el algoritmo SGD y el SGD mini-batch lo hacen peor. El tamaño de bloque utilizado en el algoritmo SGD mini-batch es de $b = 10$. En este problema en el que se disponen de pocos datos, es interesante aplicar varias pasadas para mejorar los resultados. En el caso de dar varias pasadas se obtiene

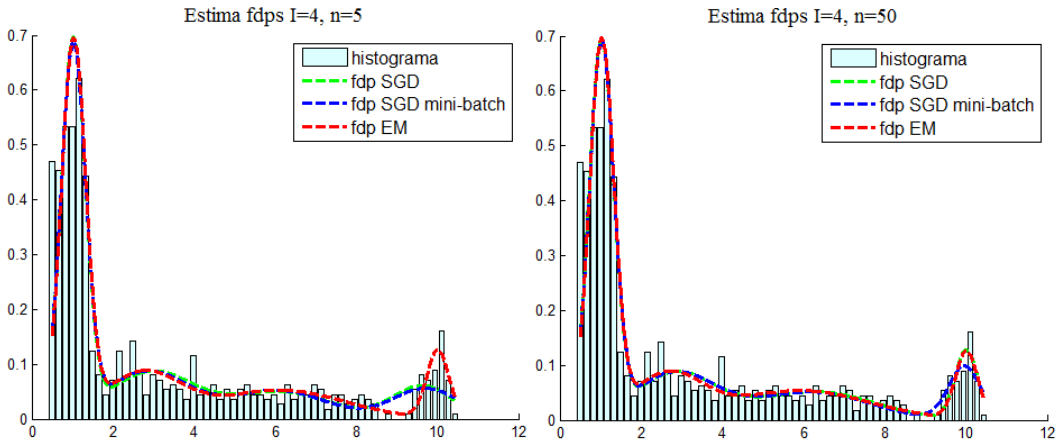

Figura 4.25: Ejemplo 1, análisis comparativo en el ajuste al histograma de los datos utilizando I=4 y n=5 vs n=50

Cuanto mayor es el número de pasadas, las fdps estimadas obtenidas con el agoritmo SGD y el SGD mini batch se acercan más a la obtenida con el algoritmo EM.

Ejemplo 2

En este segundo ejemplo se consideran los siguientes datos sintéticos, $J = 1000, D = 2,\ I = 2$

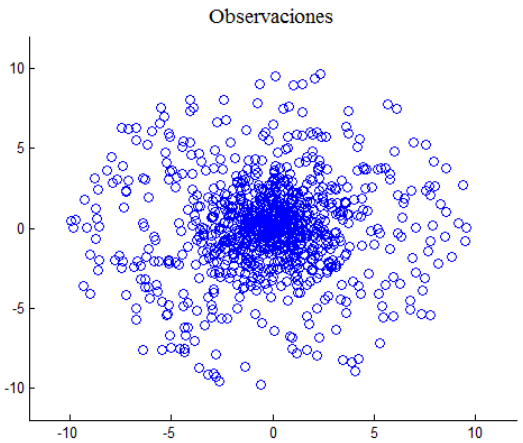

Figura 4.26: Ejemplo 2, distribución de los datos

En la figura anterior se distinguen dos clases, una de ellas presenta las observaciones más concentradas que la otra. A continuación se muestra el histograma para cada componente de los datos

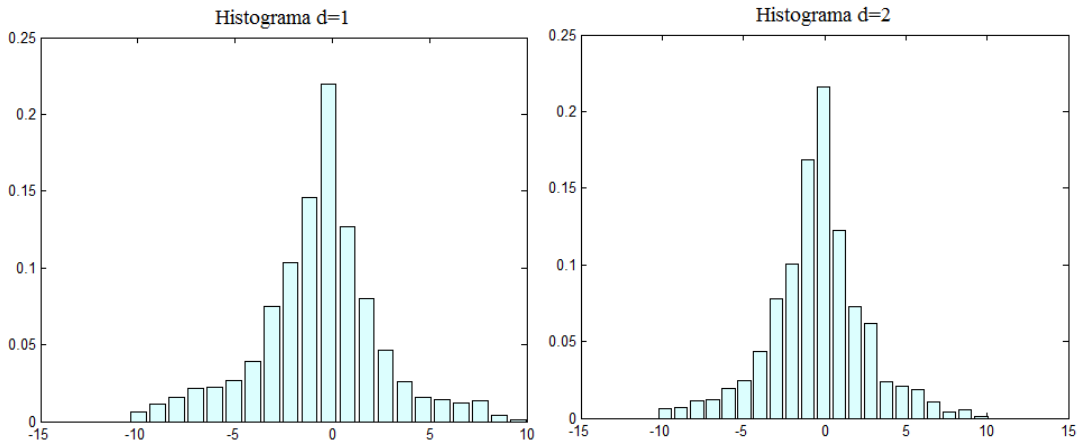

Figura 4.27: Ejemplo 2, histograma de los datos d=1 y d=2

El histograma de los datos es similar al que se obtenía en la Ap. Ruido impulsivo, presenta una forma Gaussiana. Se intentará modelar este problema mediante una mezcla Gaussiana con $I = k = 2$.

A continuación se muestra el ajuste de las fdps estimadas al histograma de los datos a modo de comparación

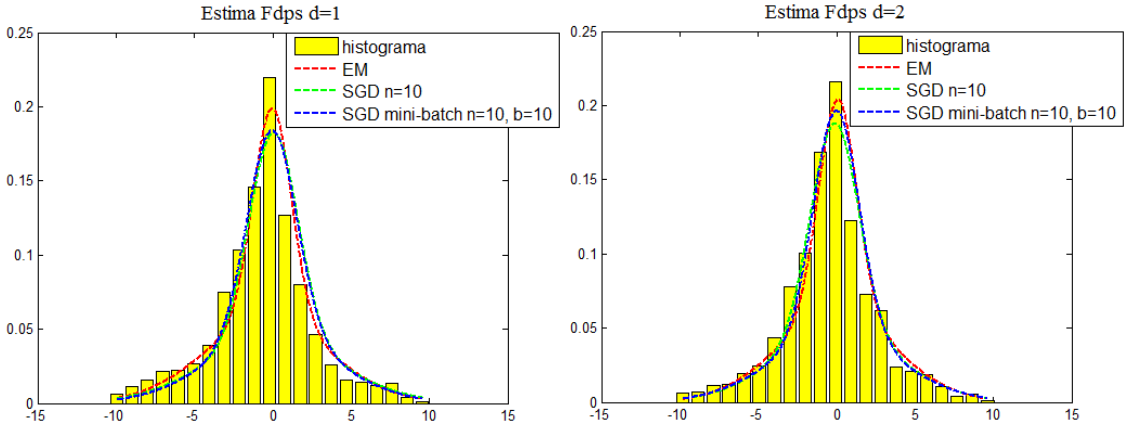

Figura 4.28: Ejemplo 2, análisis comparativo de los algoritmos en el ajuste al histograma de los datos

Como se muestra en las figuras anteriores, los resultados obtenidos son mejores en el caso de utilizar el algoritmo EM, sin embargo, no hay diferencias significativas al utilizar el algoritmo SGD y el SGD mini-batch en este problema.

Ejemplo 3

En este tercer ejemplo se plantea un caso en el que las observaciones se disponen distribuidas a lo largo de dos rectángulos del mismo tamaño. Las observaciones dentro de cada rectángulo no se encuentran distribuidas de forma uniforme, sino que existe una mayor concentración en diferentes partes.

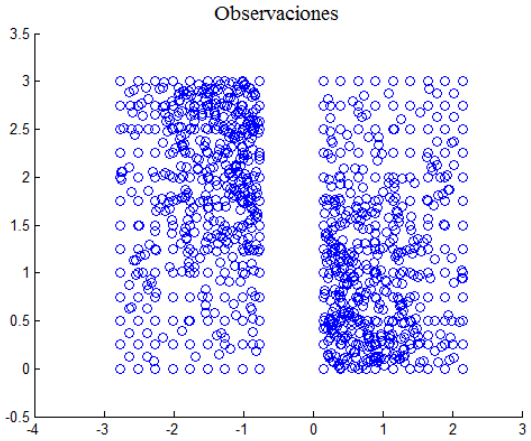

Figura 4.29: Ejemplo 3, distribución de los datos

Los resultados que se obtienen para $I = 2$ son los siguientes

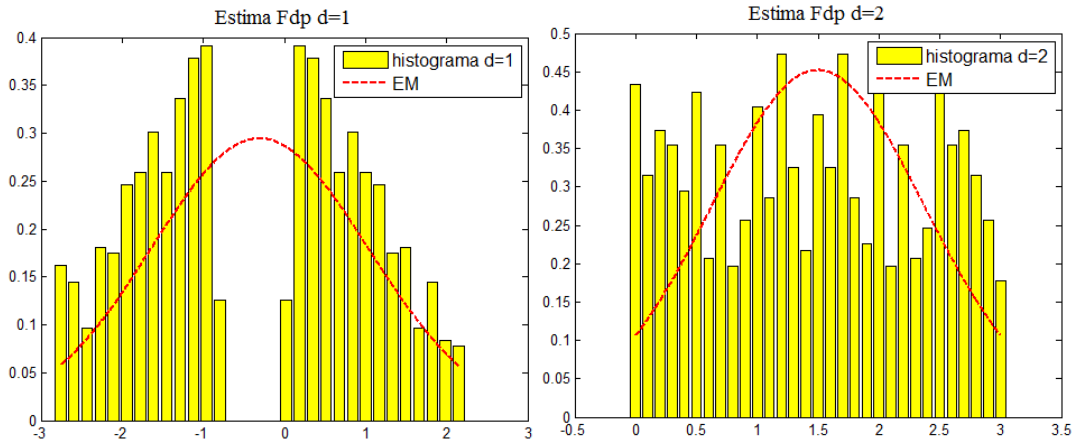

Figura 4.30: Ejemplo 3, ajuste al histograma de los datos algoritmo EM

Como se muestra en los resultados, el ajuste de las fdps estimadas al histograma de los datos no se realiza correctamente ya que no se ha utilizado un valor de I óptimo. Si se aumenta I es posible mejorar el ajuste de la fdp estimada como se muestra a continuación

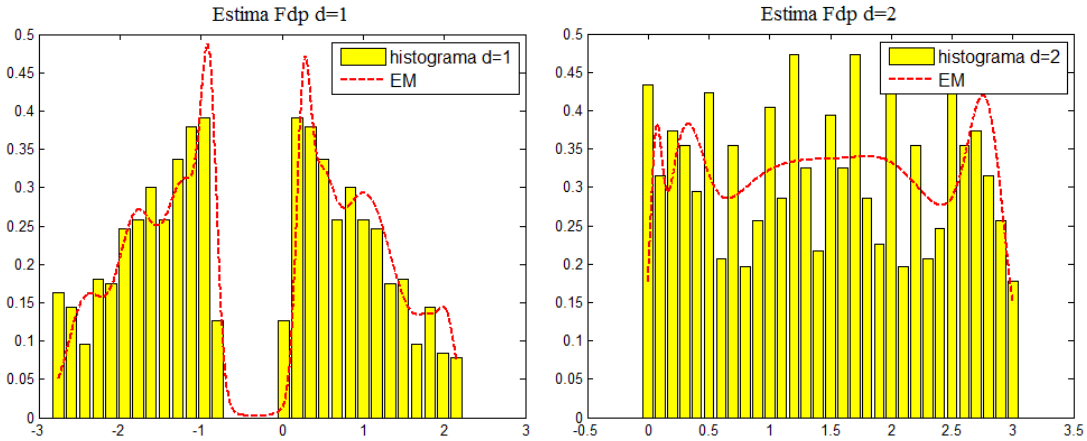

Figura 4.31: Ejemplo 3, ajuste al histograma de los datos I=20, algoritmo EM

Al utilizar un valor de I mayor ($I = 20$) el modelo de mezclas Gaussianas se adapta mejor para $d = 1$, sin embargo para $d = 2$, a pesar de obtenerse un mejor ajuste, los resultados no son tan buenos.

Ejemplo 4

En este ejemplo los datos utilizados se distribuyen de la siguiente forma

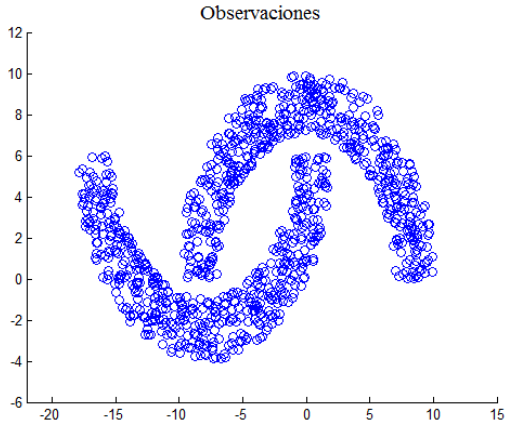

Figura 4.32: Ejemplo 4, distribución de los datos

Al intentar ajustas el histograma de los datos por una mezcla de Gaussianas con $I = 2$ se obtienen los siguientes resultados

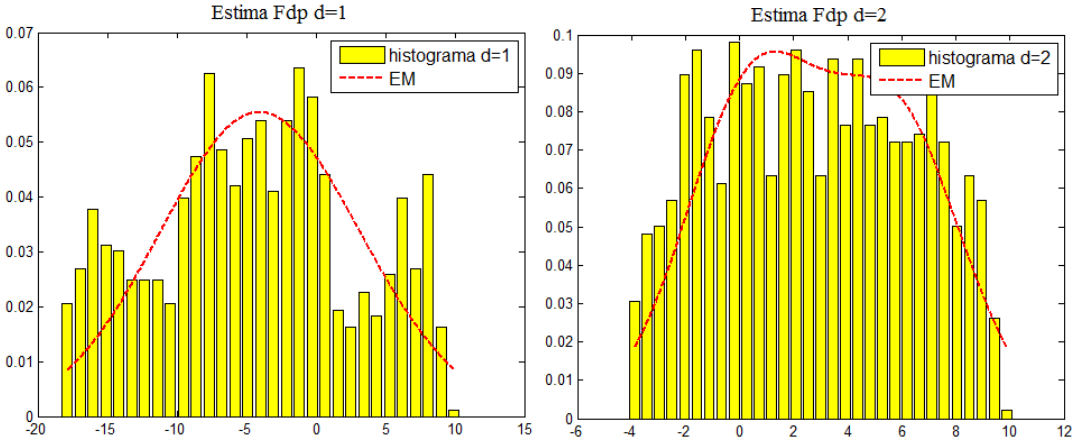

Figura 4.33: Ejemplo 4, ajuste al histograma de los datos I=2, algoritmo EM

El ajuste al histograma de los datos puede mejorarse al incrementar I como se muestra a continuación

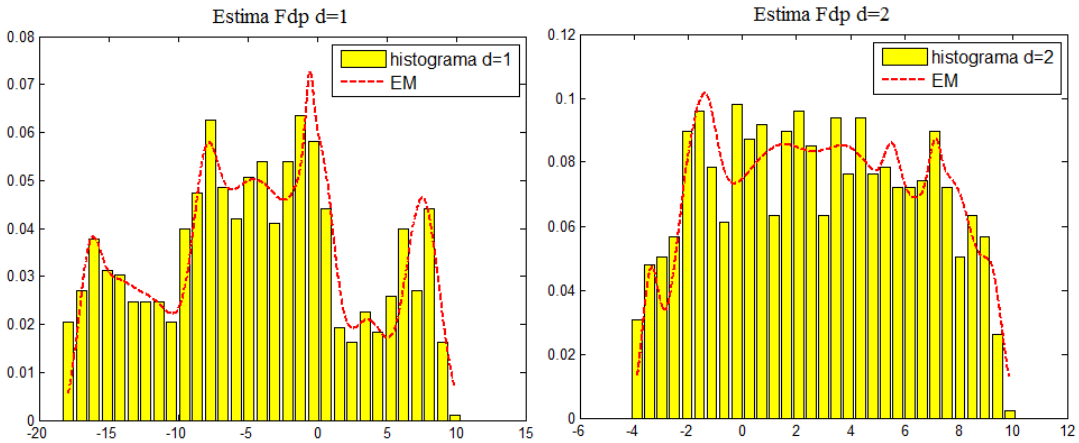

Figura 4.34: Ejemplo 4, ajuste al histograma de los datos I=20, algoritmo EM

4.3 Clústering

En este apartado se mostrarán los resultados obtenidos con los diversos algoritmos en problemas de clústering. Para ello se utilizarán algunas de las bases de datos del *Apartado 4.2*.

Ejemplo 1

Se considera la siguiente base de datos explicada en el apartado anterior

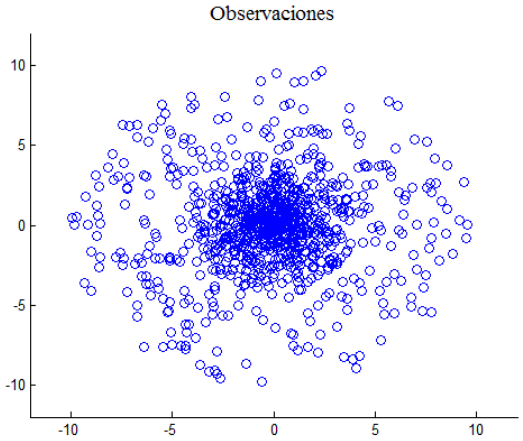

Figura 4.35: Ejemplo 1 clústering, distribución de los datos

Los resultados obtenidos son los siguientes

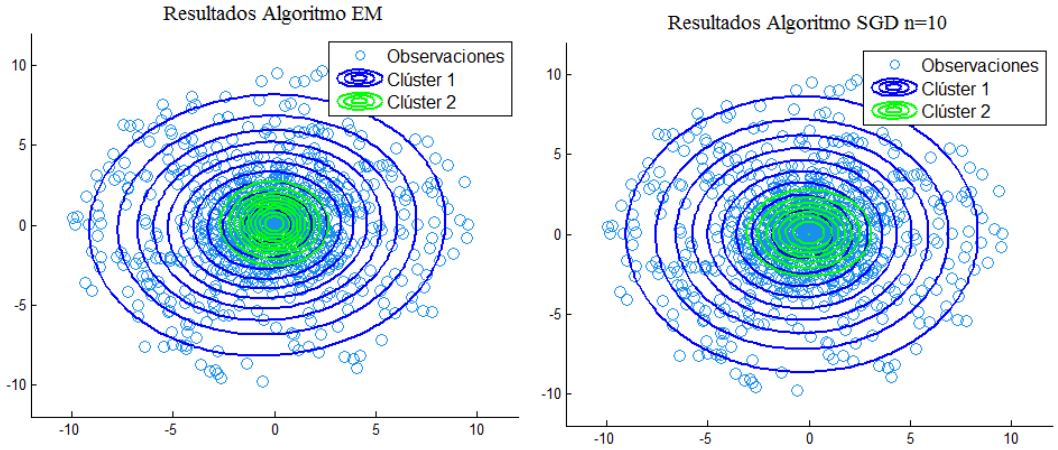

Figura 4.36: Ejemplo 1 clústering, resultados algoritmo EM y SGD n=10

En la figura anterior se ha representado cada Gaussiana de la mezcla como un clúster diferente. Para ello se han representado las curvas de nivel de cada Gaussiana centradas en las medias obtenidas.

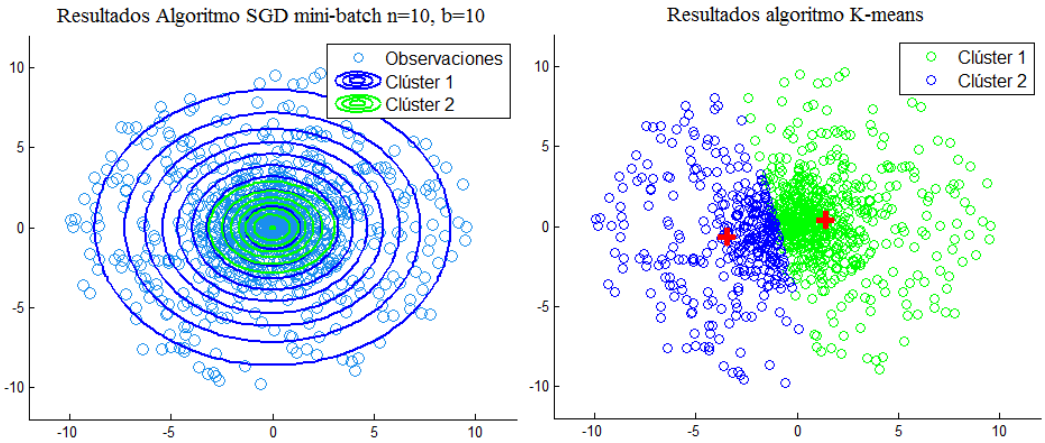

Figura 4.37: Ejemplo 1 clústering, resultados algoritmo SGD mini-batch n=10, b=10 y K-means

Los resultados obtenidos utilizando el modelo de mezclas Gaussianas son similares. El algoritmo K-means no es capaz de distinguir los dos clústers debido a que están superpuestos. En rojo se ha representado las medias de los clústers obtenidas con el algoritmo K-means. Para esta base de datos los resultados son mejores en el caso de utilizar el modelo de mezclas Gaussianas.

Ejemplo 2

Se considera la siguiente base de datos explicada anteriormente

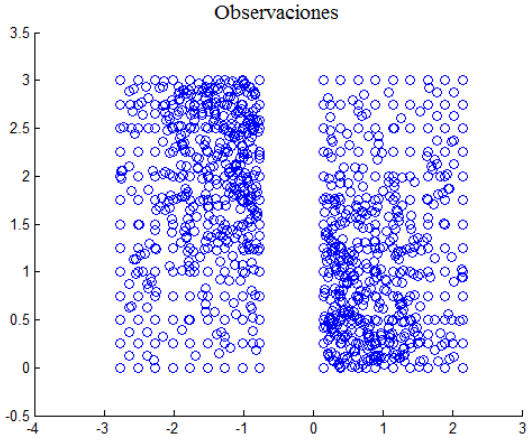

Figura 4.38: Ejemplo 2 clústering, distribución de los datos

Los resultados obtenidos por medio del algoritmo EM y el K-means se muestran a continuación

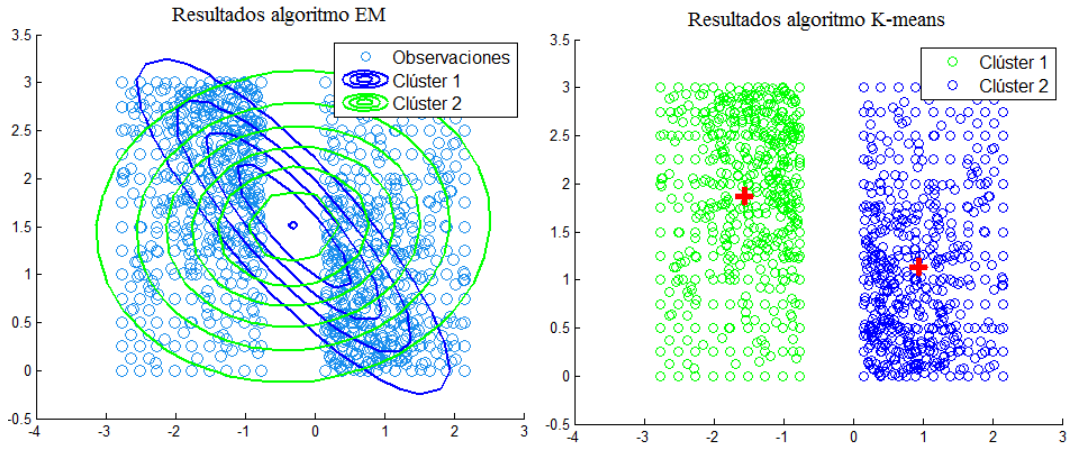

Figura 4.39: Ejemplo 2, resultados algoritmo EM y K-means

El algoritmo K-means resuelve el problema debido a que los clústers se encuentran bien separados (diferenciados). En rojo se ha representado las medias obtenidas con el algoritmo. Sin embargo el algoritmo EM no lo resuelve correctamente debido a que distingue como un clúster (clase) a los dos conjuntos de observaciones más concentrados de las esquinas de los rectángulos y otro clúster al resto de observaciones que presentan una concentración menor.

Ejemplo 3

Se considera la siguiente base de datos explicada en el apartado anterior

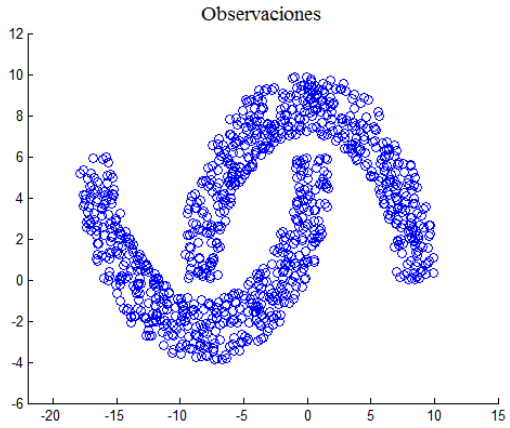

Figura 4.40: Ejemplo 3 clústering, distribución de los datos

A continuación se muestran los resultados obtenidos por medio del algoritmo EM y el K-means para $I = 2$

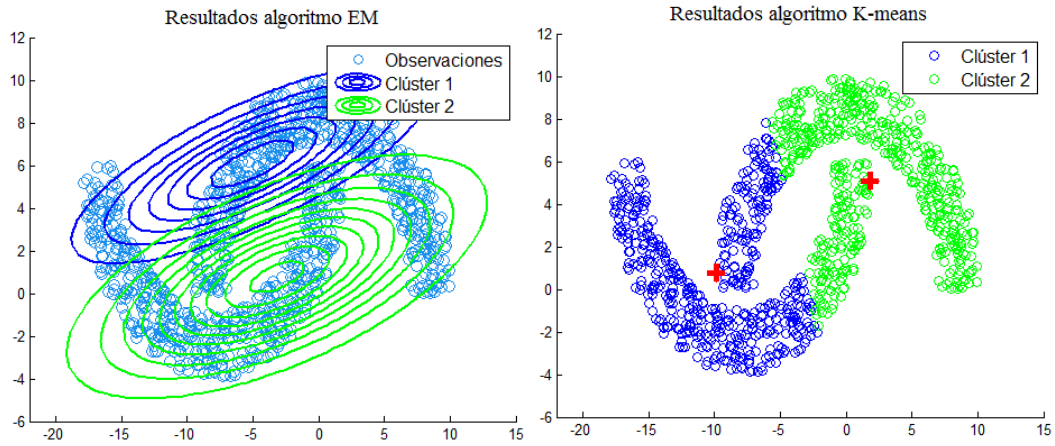

Figura 4.41: Ejemplo 3, resultados algoritmo EM y K-means

Como se muestra en los resultados, se trata de un ejemplo en el que ninguno de los algoritmos distingue correctamente los clústers.

4.4 Segmentación de imágenes

Se define a segmentación de imágenes al proceso de dividir una imagen digital en varios objetos, partes o grupos de píxeles. El objetivo es simplificar y/o cambiar la representación de una imagen en otra más significativa y más fácil de analizar. Se puede usar tanto para localizar objetos como para encontrar los límites de estos dentro de una imagen. Mediante el proceso de segmentación se asigna una etiqueta (clase o clúster) a cada píxel de la imagen de forma que los píxeles que compartan la misma etiqueta también tendrán ciertas características visuales similares.

En este apartado se propone el uso de algoritmos de agrupamiento para segmentar imágenes. En concreto se reducirá el número de colores de una imagen para reconocer mejor las formas de los objetos, o simplemente permitir la compresión de la misma como se explicará en el *Apartado 4.5*.

Se utilizará la siguiente imagen (llamada Lenna) para realizar las pruebas

Figura 4.42: Segmentación de imágenes, Imagen Lenna

Las imágenes están formadas por un conjunto de píxeles, el número de estos dependerá de la resolución de las mismas. Una imagen con resolución 20x40 contendrá 800 píxeles. Se va a trabajar con el modelo RGB, por tanto a la hora de procesar una imagen el conjunto de datos será el conjunto de píxeles y cada píxel/observación tendrá 3 valores, 3 dimensiones ($D = 3$). Los valores de cada dimensión con los que se trabajarán $\epsilon\mathbb{R}, [0,1]$.

La primera prueba que se realizará será la de reducir los colores de la imagen. En el caso de distinguir únicamente dos colores $I = 2$ los resultados que se obtuvieron fueron los siguientes

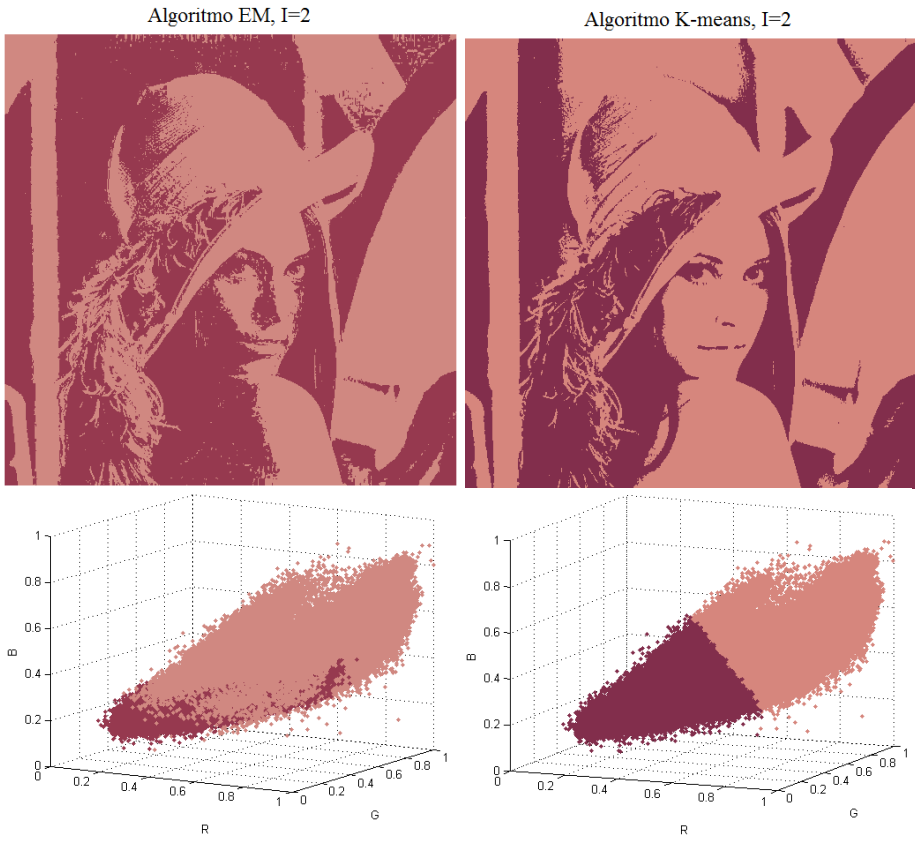

Figura 4.43: Segmentación de imágenes, análisis comparativo clustering I=2

En la figura superior se han representado los resultados obtenidos mediante cada algoritmo y en la parte inferior los píxeles tras la segmentación. Los píxeles tras la segmentación se han coloreado con la tonalidad asociada al valor de la media de la clase a la que pertenece. El algoritmo K-means realiza hard clústering y los píxeles de los 2 clústers se encuentran claramente separados. En el caso del algoritmo EM se ha realizado hard clústering mediante *(2.18)* como se explica en el *Apartado 2.3* y los clústers pueden estar superpuestos como se muestra en los píxeles tras la segmentación *Figura 4.43*.

Aumentar el número de colores I, supondría aumentar el número de clústers a determinar, los resultados obtenidos se asimilarían cada vez más a la imagen original como se muestra a continuación

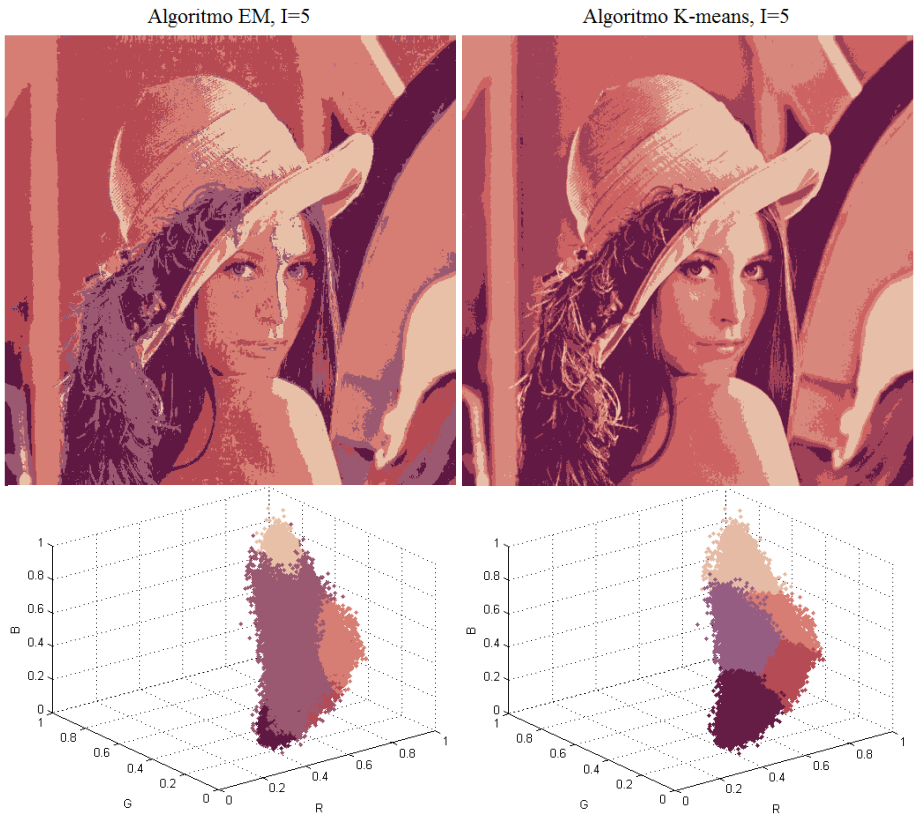

Figura 4.44: Segmentación de imágenes, análisis comparativo clústering I=5

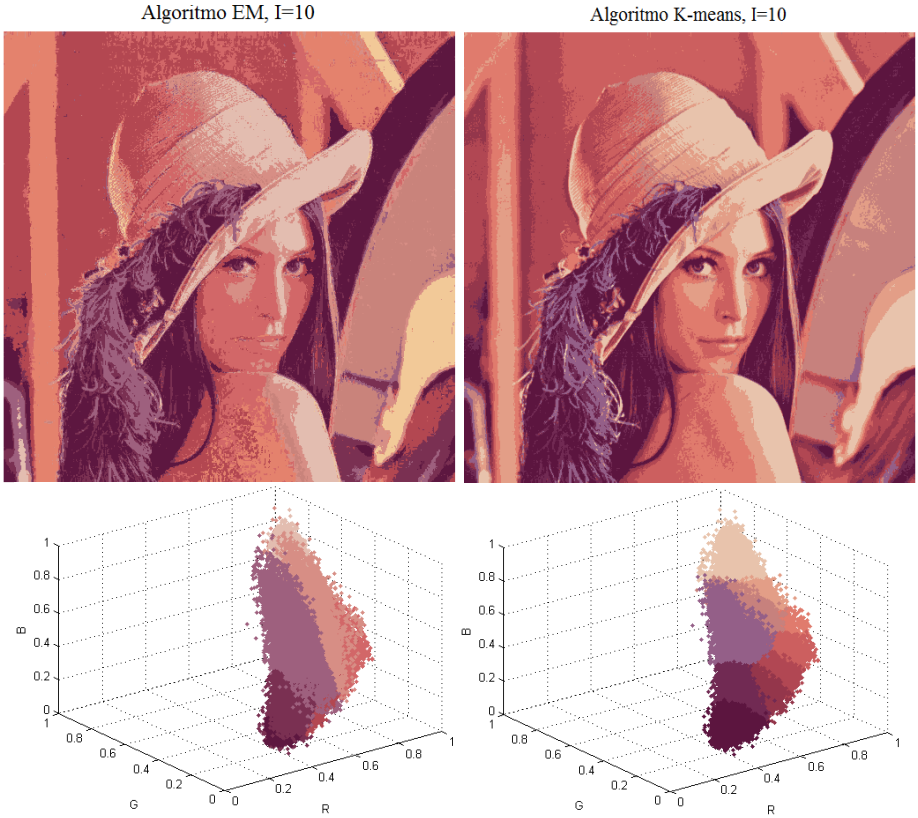

Figura 4.45: Segmentación de imágenes, análisis comparativo clústering I=10

Como se muestra en los resultados, al ir aumentando *I* el algoritmo K-means obtiene mejores resultados frente al modelo de mezclas Gaussianas aplicado al algoritmo EM.

Ambos algoritmos pueden usarse también para convertir imágenes a blanco y negro o a escala de grises. Para convertirlas a blanco y negro el procedimiento sería el mismo que el utilizado en los resultados anteriores, ya que en el caso de inicializar los algoritmos con dos clases, se van a obtener los dos colores más representativos de la imagen. Una vez realizado el clústering de los píxeles, se asignaría al color más claro el blanco y al más oscuro el negro. En el caso de escala de grises el procedimiento sería igual, sin embargo, hay que asignar los colores de forma adecuada, es decir, a las zonas más oscuras asignarlas grises más oscuros y a las zonas más claras, grises más claros. Los resultados obtenidos han sido los siguientes

Figura 4.46: Segmentación de imágenes, análisis comparativo clústering, escala de grises I=2

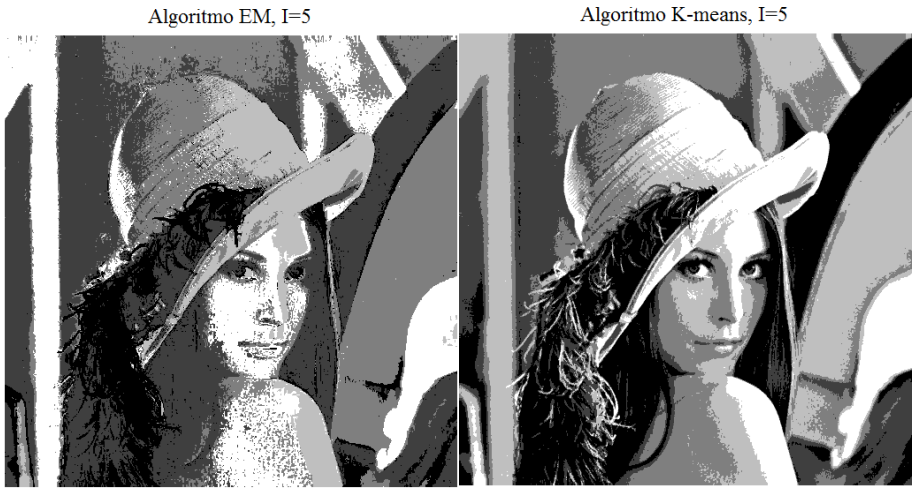

Figura 4.47: Segmentación de imágenes, análisis comparativo clústering, escala de grises I=5

Figura 4.48: Segmentación de imágenes, análisis comparativo clústering, escala de grises I=10

En este problema el algoritmo K-means obtiene mejores resultados. Sin embargo, en el caso del algoritmo EM, no se ha aprovechado la incertidumbre de los píxeles de pertenecer a cada clúster *(2.18)* ya que se ha realizado hard clústering. Esa incertidumbre quizás podría explotarse de alguna forma en la segmentación de imágenes, por ejemplo, los píxeles que tienen una probabilidad cercana a 0.5 de pertenecer a dos clases, podrían englobarse en nueva clase en la que el color fuera el color medio de ambas clases.

Las razones por las que el algoritmo EM no funciona tan bien como el algoritmo K-means en esta aplicación son debidas a una serie de detalles a tener en cuenta. En primer lugar el modelo de mezclas Gaussianas utilizado en el algoritmo EM no es muy apropiado para trabajar con datos de este tipo debido a que en una imagen suele haber muchos píxeles con el mismo color, el modelo RGB implica la existencia de datos discretos. Esto obliga a utilizar regularización de las matrices de covarianza en algunas iteraciones del algoritmo (no ocurre en todas las imágenes, es más probable cuanto menor sea el nº de colores en una imagen). En segundo lugar la forma en la que se distribuyen los clústers en el algoritmo K-means es más apropiada ya que se encuentran claramente diferenciados, no existe superposición de los mismos como se muestra a continuación

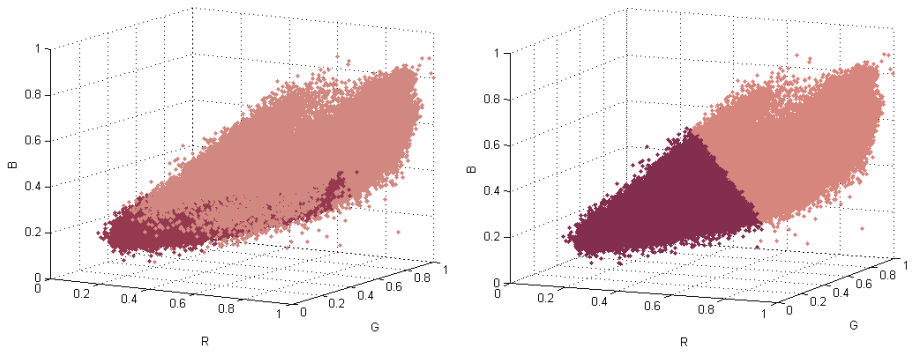

Figura 4.49: Segmentación de imágenes, análisis comparativo clustering

A la izquierda se han representado los resultados obtenidos con el algoritmo EM tras la segmentación con $I = 2$ y a la derecha los obtenidos con el algoritmo K-means. Como se muestra, el algoritmo EM asigna al mismo clúster píxeles con tonalidades más diferentes entre sí mientras que el algoritmo K-means separa los clústers con tonalidades más similares.

4.5 Compresión y reconstrucción de datos

En esta aplicación se explica la posibilidad de comprimir una base de datos y cómo reconstruirla posteriormente. La compresión de datos puede resultar muy interesante en sistemas con capacidad de almacenamiento limitada, cuanta mayor sea la cantidad de datos mayor será la fiabilidad de la reconstrucción y más útil resulta la compresión de los datos.

Dependiendo del grado de compresión y la cantidad de datos de la que se disponga se plantean dos opciones.

La primera opción aporta un mayor grado de compresión y puede aplicarse si utilizamos el modelo de mezclas Gaussianas (utilizado en el algoritmo EM, SGD y SGD mini-batch) ya que presenta la ventaja de poder estimar la probabilidad de pertenencia a cada clúster de las observaciones. Esto permitirá un grado de compresión enorme, de aquí en adelante, se designará a este tipo de compresión compresión hard. Para este tipo de compresión es necesario un número de observaciones elevado (cuanto mayor sea el número de observaciones, más fiable será la posterior reconstrucción). Supóngase que se dispone de una base de datos sintéticos que contiene $J = 10000$ observaciones generadas a partir de $I = 3$ Gaussianas con los siguientes parámetros

Parámetros	Clase 1	Clase 2	Clase 3
Probabilidades	$\alpha_1 = 0.2$	$\alpha_2 = 0.3$	$\alpha_3 = 0.5$
Medias	$\boldsymbol{\mu}_1 = [4\ 1]'$	$\boldsymbol{\mu}_2 = [-4\ -2]'$	$\boldsymbol{\mu}_3 = [0\ 0]'$
Matrices covarianza	$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$	$\boldsymbol{\Sigma}_2 = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}$	$\boldsymbol{\Sigma}_3 = \begin{pmatrix} 1 & -0.5 \\ -0.5 & 1 \end{pmatrix}$

Tabla 4.12: Parámetros utilizados para la generación base de datos Ap. Compresión y reconstrucción de datos

Las observaciones se distribuyen de la siguiente forma

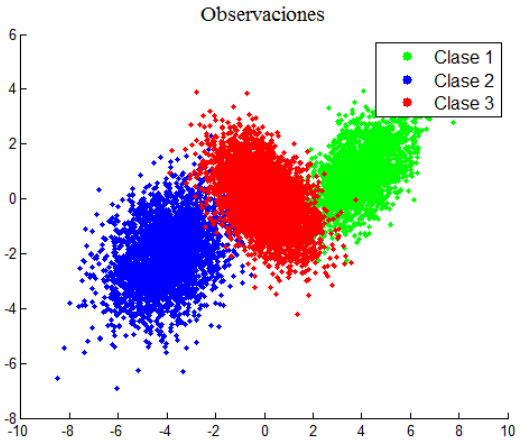

Figura 4.50: Compresión y reconstrucción de datos, distribución de los datos sintéticos

Este primer tipo de compresión consiste en aplicar el aplicar el algoritmo EM, el SGD o el SGD mini-batch al problema para el modelo de mezclas Gaussianas. Por ejemplo, supóngase que se utiliza el algoritmo EM para realizar el clústering de los datos. Se obtuvieron los siguientes resultados

Parámetros	Clase 1	Clase 2	Clase 3
Probabilidades	$\alpha_{1\,est} = 0.2056$	$\alpha_{2\,est} = 0.3020$	$\alpha_{3\,est} = 0.4924$
Medias	$\boldsymbol{\mu}_{1\,est} = [-3.9759\ 0.9809]'$	$\boldsymbol{\mu}_{2\,est} = [-3.9875 - 1.9917]'$	$\boldsymbol{\mu}_{3\,est} = [-0.0113\ 0.001]'$
Matrices covarianza	$\boldsymbol{\Sigma}_{1\,\text{est}} = \begin{pmatrix} 1.04 & 0.52 \\ 0.52 & 1.04 \end{pmatrix}$	$\boldsymbol{\Sigma}_{2\,\text{est}} = \begin{pmatrix} 1.47 & 0.48 \\ 0.48 & 1.48 \end{pmatrix}$	$\boldsymbol{\Sigma}_{3\,\text{est}} = \begin{pmatrix} 0.91 & -0.47 \\ -0.47 & 0.99 \end{pmatrix}$

Tabla 4.13: Resultados algoritmo EM, Ap. Compresión y reconstrucción de datos

Este tipo de compresión se basa en almacenar únicamente las estimas de $\alpha, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ y el número de observaciones J. Se pasaría de tener una base de datos de números reales de tamaño JxD a almacenar únicamente los parámetros anteriormente indicados. Si se realiza esto, para poder reconstruir la base de datos, únicamente se deberían generar J observaciones de forma aleatoria siendo el 20.56% de ellas realizaciones de una v.a. Gaussiana con media $\boldsymbol{\mu}_{1\,est}$ y matriz de covarianza $\boldsymbol{\Sigma}_{1\,\text{est}}$, el 30.20% realizaciones de una v.a. Gaussiana con media $\boldsymbol{\mu}_{2\,est}$,… etc. De esta forma se perdería algo de información ya que la base de datos reconstruida no sería exactamente igual a la original. Conforme aumenta J para un problema concreto, la efectividad aumenta y las diferencias entre la base de datos original y la reconstruida disminuyen. Este tipo de compresión también se puede realizar por medio del algoritmo K-means, sin embargo para reconstruir la base de datos sería necesario obtener las matrices de covarianza de los clústers y α (para obtener $\boldsymbol{\Sigma}$ se calcularía la covarianza de las observaciones pertenecientes a cada clúster y para obtener α, la media muestral). Si se reconstruye la base de datos del problema anterior se obtiene

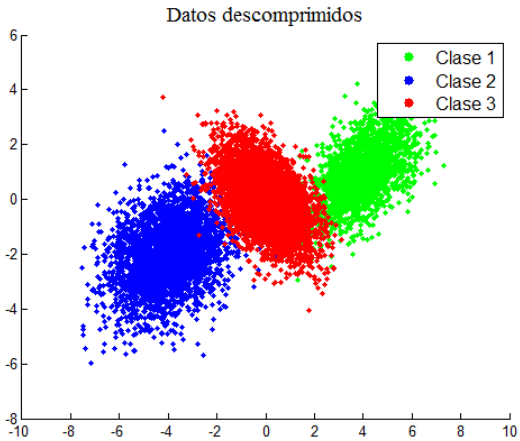

Figura 4.51: Compresión y reconstrucción de datos, reconstrucción de las observaciones

En el caso de realizar una compresión teniendo en cuenta a que clúster pertenece cada observación individualmente (ya sea por tener pocas observaciones o por una mayor fiabilidad) y el orden de las observaciones, se deberán almacenar los mismos parámetros que antes, excepto J. Adicionalmente es necesario almacenar un vector de números naturales de dimensiones $Jx1$ en el que cada posición se indicará con un número natural a que Gaussiana pertenece esa observación. A este tipo de compresión se la designará con el nombre de compresión soft y podrá realizarse con todos los algoritmos presentados en el este trabajo. Este tipo de compresión permite reconstruir la base de datos con una mayor fiabilidad. Cabe mencionar que para una base de datos de dimensiones muy grandes, cuanto mayor sea D mayor será el factor de compresión, puesto que cada observación pasará de ser un vector de números reales de tamaño $1xD$ a ser representada por un número natural que indicará a que clúster pertenece. En el caso

del algoritmo EM es necesario realizar una clasificación hard de las observaciones para poder almacenar el vector de número naturales.

A continuación se adjunta una tabla comparativa en la que se muestra el grado de compresión de la base de datos del problema planteado anteriormente utilizando el algoritmo K-means (con los algoritmos EM, SGD y SGD mini-batch, el factor de compresión coincidiría)

J=10000, D=2, I=3	Datos sin comprimir	Datos comprimidos	Factor compresión
Compresión hard	156.25kB	75.25B	99.95%
Compresión soft	156.25kB	2573.5B	98.39%

Tabla 4.14: Resultados de compresión mediante el algoritmo K-means, Ap. Compresión y reconstrucción de datos

La compresión y reconstrucción de datos puede resultar interesante en imágenes. En el caso de aplicarlo a la imagen del experimento de segmentación de imágenes anterior, sería necesario utilizar una compresión soft debido a que el orden de las observaciones es imprescindible ya que es lo que permitirá reconstruir los píxeles en el orden adecuado. En el caso de utilizar el algoritmo K-means, el EM o el SGD únicamente será necesario almacenar el vector de números naturales que indica a que clúster pertenece cada píxel y las medias que representan los colores de cada clúster. Los resultados de compresión obtenidos con cualquiera de los algoritmos utilizando 10 colores/clases son los siguientes:

Parámetros iniciales	Píxeles sin comprimir	Píxeles comprimidos	Factor de compresión
J=262144 D=3 I=10	6MB	128.23kB	97.91%

Tabla 4.15: Resultados de compresión mediante el algoritmo EM, Ap. Compresión y reconstrucción de datos

Como se muestra en la tabla anterior, el grado de compresión es significativo.

Y al reconstruir la imagen se obtiene

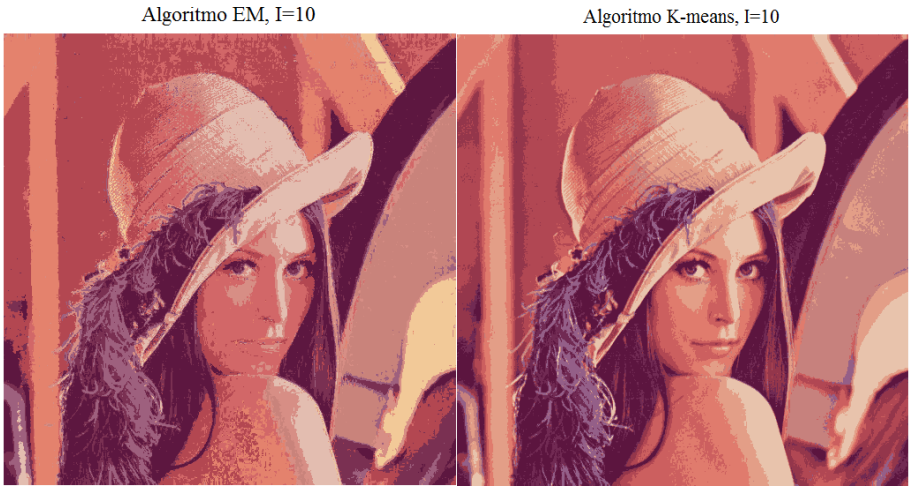

Figura 4.52: Reconstrucción de la imagen Lenna, Ap. Compresión y reconstrucción de datos

Como se muestra en la figura anterior, el resultado es el mismo que se obtuvo al reducir la imagen a 10 colores en el *Apartado 4.4*. Esto es debido a que al guardar el vector de números naturales que indica a que clúster pertenece cada píxel y $\boldsymbol{\mu}$, se recupera la imagen tal cual quedó tras su segmentación.

4.6 Base de datos breast_cancer

En este apartado se pretende mostrar la utilidad de la aplicación de algoritmos de agrupamiento en problemas reales.

Los datos analizados en este apartado han sido proporcionados por el Dr. William H. Wolberg, de la Universidad de Wisconsin y pueden encontrarse en [26]. La base de datos contiene los resultados de biopsias realizadas a un conjunto de pacientes para determinar si tienen o no un cáncer de mama. Una biopsia es un procedimiento diagnóstico que consiste en la extracción de una muestra total o parcial de tejido y posteriormente su análisis en un laboratorio. La base de datos analizada contiene muestras de tejido de 683 pacientes. A cada paciente se le ha extraído una muestra de tejido y posteriormente se la han realizado 9 tests para determinar si ese paciente presenta un tumor maligno o benigno.

Los datos analizados se encuentran etiquetados, una vez obtenidos los resultados se puede comprobar por tanto que tal ha resuelto el problema cada algoritmo. De los 683 pacientes, 444 presentan un tumor benigno y 239 un tumor maligno. Cada paciente se representa por los 9 resultados de los tests que se han aplicado sobre su muestra de tejido, por tanto se dispone de $J = 683$ observaciones de dimensión $D = 9$. A priori se considerará la existencia de dos clases diferentes únicamente $I = 2$, pacientes con tumores malignos y pacientes con tumores benignos.

4.6.1 Algoritmo EM

En primer lugar se analizará la base de datos para el modelo de mezclas Gaussianas utilizando el algoritmo EM. Se han realizado las simulaciones utilizando una inicialización aleatoria y considerando $I = 2$ clases. Los resultados obtenidos son los siguientes

Clase 1: Benignos									
α_1	0.5449								
μ_1	2.8426	1.1124	1.2648	1.1677	1.9339	1.0702	1.9759	1.0745	1.0134
Σ_1	2.6798	0.0309	0.1430	0.1777	0.0919	−0.0218	−0.0014	0.0722	0.0154
	0.0309	0.2646	0.0466	0.0082	0.0182	0.0095	0.0043	0.0197	0.0061
	0.1430	0.0466	0.4482	0.0223	0.0124	−0.0086	−0.0030	−0.0030	0.0010
	0.1777	0.0082	0.0223	0.3486	0.0066	0.0062	−0.0181	0.0004	0.0146
	0.0919	0.0182	0.0124	0.0066	0.2241	0.0098	−0.0088	−0.0006	0.0055
	−0.0218	0.0095	−0.0086	0.0062	0.0098	0.1248	0.0327	0.0029	−0.0056
	−0.0014	0.0043	−0.0030	−0.0181	−0.0088	0.0327	0.7132	0.0219	0.0149
	0.0722	0.0197	−0.0030	0.0004	−0.0006	0.0029	0.0219	0.1679	−0.0022
	0.0154	0.0061	0.0010	0.0146	0.0055	−0.0056	0.0149	−0.0022	0.0883
Clase 2: Malignos									
α_2	0.4551								
μ_2	6.3501	5.5485	5.5658	4.8026	4.7941	6.5164	5.2108	5.0359	2.3229
Σ_2	7.7144	3.6131	3.7689	1.6792	1.6211	2.9282	2.2718	2.5467	1.3086
	3.6131	9.6662	7.7680	4.8699	4.2754	3.5054	4.6119	5.0605	2.2511
	3.7689	7.7680	9.2726	4.3699	3.8929	4.1453	4.2903	5.0720	1.9801
	1.6792	4.8699	4.3699	10.7792	2.7221	4.5655	3.8979	3.8843	2.0025
	1.6211	4.2754	3.8929	2.7221	6.2673	1.8138	2.4191	3.1986	2.0469
	2.9282	3.5054	4.1453	4.5655	1.8138	12.8631	3.5964	2.3474	0.7652
	2.2718	4.6119	4.2903	3.8979	2.4191	3.5964	6.7863	3.8877	0.9635
	2.5467	5.0605	5.0720	3.8843	3.1986	2.3474	3.8877	11.9358	2.3427
	1.3086	2.2511	1.9801	2.0025	2.0469	0.7652	0.9635	2.3427	5.6737

Tabla 4.16: Resultados mediante el algoritmo EM, I=2, base de datos breast_cancer

Se diferencian claramente dos clases. Si se comparan los valores de α obtenidos con lo que se obtendría si se realizara la media muestral de la base de datos etiquetada se obtendría

$$\alpha_1 = \frac{444}{683} = 0.6501 \quad y \quad \alpha_2 = \frac{239}{683} = 0.3499$$

Existe una ligera diferencia entre los resultados anteriores y los obtenidos con el algoritmo EM. A continuación se han utilizado los resultados del algoritmo EM para realizar hard clústering de las observaciones y analizar los errores que se producen. A continuación se define la notación que se utilizará de aquí en adelante para la clasificación de los tumores

	Hipótesis H_1: Tumor Maligno	Hipótesis H_0: Tumor Benigno
Test positivo	Verdadero positivo (VP), $H_1\|H_1$	Falso positivo (FP), $H_1\|H_0$
Test negativo	Falso negativo (FN), $H_0\|H_1$	Verdadero negativo (VN), $H_0\|H_0$

Tabla 4.17: Notación a utilizar en los resultados de clasificación, base de datos breast_cancer

La P_d representa la probabilidad de que un sujeto con un tumor maligno tenga un resultado positivo en el test

$$P_d = \frac{VP}{VP + FN} = P(H_1|H_1) \qquad (4.3)$$

La P_{fa} representa la probabilidad de que un sujeto sano tenga un resultado positivo en la prueba

$$P_{fa} = \frac{FP}{VN + FP} = P(H_1|H_0) \qquad (4.4)$$

Los resultados obtenidos mediante la clasificación son los siguientes

	Número de pacientes	Porcentaje (%)
VP	239	34,99%
FP	72	10,54%
FN	0	0%
VN	372	54,47%

P_d	P_{fa}
100%	16,22%

Tabla 4.18: Resultados clasificación algoritmo EM, base de datos breast_cancer

La P_d obtenida utilizando el algoritmo como clasificador es del 100%, esto es debido a que no se ha producido ningún FN. Sin embargo, existe una P_{fa} no nula ya que se ha producido un FP en el 10,54% de los pacientes. Para esta base de datos en concreto todos los pacientes con tumor maligno han sido detectados, sin embargo, se han producido 10.54% FP. Los resultados son muy buenos puesto que se considera más grave no detectar un tumor maligno que la aparición de FP.

Si se realiza un pre-procesado de los datos calculando los coeficientes de correlación de la base de datos se puede observar cómo de correlados se encuentran los test (consideramos 1 test a cada dimensión) entre sí

Coef =

1.0000	0.6349	0.6495	0.4909	0.5171	0.5870	0.5512	0.5344	0.3521
0.6349	1.0000	0.8978	0.7002	0.7431	0.6787	0.7482	0.7083	0.4600
0.6495	0.8978	1.0000	0.6752	0.7176	0.7015	0.7285	0.7076	0.4371
0.4909	0.7002	0.6752	1.0000	0.5903	0.6630	0.6564	0.6013	0.4152
0.5171	0.7431	0.7176	0.5903	1.0000	0.5759	0.6152	0.6214	0.4766
0.5870	0.6787	0.7015	0.6630	0.5759	1.0000	0.6708	0.5742	0.3319
0.5512	0.7482	0.7285	0.6564	0.6152	0.6708	1.0000	0.6559	0.3476
0.5344	0.7083	0.7076	0.6013	0.6214	0.5742	0.6559	1.0000	0.4384
0.3521	0.4600	0.4371	0.4152	0.4766	0.3319	0.3476	0.4384	1.0000

Tabla 4.19: Coeficientes de correlación, base de datos breast_cancer

En los resultados anteriores se muestra un elevado grado de correlación entre los test. A continuación se analizarán los resultados eligiendo un menor número de ellos. Para ello, se elegirán los test que estén muy correlados con los demás pero a su vez los menos incorrelados entre ellos. Tras varias pruebas se ha llegado a la conclusión de que lo óptimo sería utilizar los test 3, 4 y 6. Con ellos se obtienen los siguientes resultados

Clase 1: Benignos	
α_1	0.5101
μ_1	1.1953 1.0569 1.0436
Σ_1	0.2938 0.0213 -0.0101 0.0213 0.1353 -0.0007 -0.0101 -0.0007 0.1043
Clase 2: Malignos	
α_2	0.4890
μ_2	5.3405 4.6666 6.1670
Σ_2	9.4241 4.3524 4.8861 4.3524 10.3218 4.8458 4.8861 4.8458 13.6307

Tabla 4.20: Resultados mediante el algoritmo EM, I=2, D=3, base de datos breast_cancer

Y desde el punto de vista de clasificación se obtiene

	Número de pacientes	Porcentaje (%)
VP	239	34,99%
FP	94	13,76%
FN	0	0%
VN	350	51,24%

P_d	P_{fa}
100%	21,17%

Tabla 4.21: Resultados clasificación algoritmo EM, D=3, base de datos breast_cancer

Desde el punto de vista de clasificación, la P_d sigue siendo del 100% aunque parezca difícil de creer, sin embargo, la P_{fa} ha aumentado. Estos resultados tienen una gran utilidad, ya que a la hora de hacer los test a las muestras de los tejidos, podrían realizarse en primer lugar únicamente los test 3, 4 y 6. Esto permitiría (para esta base de datos) ahorrar tiempo y dinero realizando una primera clasificación para detectar el número máximo de tumores benignos. Por otro lado hay que tener en cuenta que por medio del modelo de mezclas Gaussianas, se puede calcular la probabilidad de pertenencia de cada observación a cada clase. Tras esa primera fase con los test 3,4 y 6 se podrían realizar los test que se crean convenientes a los pacientes en los que haya una cierta incertidumbre de tener un tumor maligno o benigno. En los casos en los que la probabilidad de tener un tumor maligno sea muy cercana a 1, se podría asegurar casi con certeza que ese paciente tiene un tumor maligno, sin embargo en los casos en los que la probabilidad de pertenecer a una u otra clase no sea muy decisiva, sería conveniente realizar

otros test posteriores. Otra opción sería realizar un clústering suponiendo 3 clases $I = 3$, una clase para los malignos, otra para los benignos y la tercera para pacientes dudosos (pacientes en los que no está muy claro el tipo de tumor). En este caso se introduciría una nueva hipótesis H_2: $Tumores\ dudosos$. Al realizar una clasificación utilizando todos los test se obtiene

		Hipótesis		
		$H_0\|$	$H_1\|$	$H_2\|$
Real	$\|H_0$	359	8	77
	$\|H_1$	0	127	112

Tabla 4.22: Resultados clasificación algoritmo EM, I=3, base de datos breast_cancer

En la tabla anterior se muestra para cada tipo de tumor etiquetado la clasificación obtenida en diferentes hipótesis. Es decir de los 444 pacientes con tumor benigno, 359 se han clasificado como benignos, 8 como malignos y 77 como tumores dudosos.

Los resultados tienen un gran interés, se detectan 359 VN, 127 VP y 0 FN. Por tanto, los pacientes clasificados con un tumor dudoso podrían someterse a otros test o análisis posteriormente.

Si se realiza la clasificación utilizando únicamente los test 3,4 y 6 se obtiene

		Hipótesis		
		$H_0\|$	$H_1\|$	$H_2\|$
Real	$\|H_0$	304	43	97
	$\|H_1$	0	236	3

Tabla 4.23: Resultados clasificación algoritmo EM, I=3, D=3, base de datos breast_cancer

Se detectan 304 VN, 236 VP y 0 FN. Tras los resultados mostrados se diferencian dos posibilidades. La primera es realizar todos los test a todos los pacientes para tener una mayor fiabilidad en general y la segunda es realizar menos test en una primera fase y posteriormente otros test a los que queden clasificados como dudosos o malignos.

4.6.2 K-means

En el caso de la utilización del algoritmo K-means, en los resultados directamente se obtendrá a qué clase clasifica cada paciente. En el caso de utilizar los 9 test se obtiene

	Número de pacientes	Porcentaje (%)
VP	222	32,50%
FP	8	1,17%
FN	17	2,49%%
VN	436	63,84%

P_d	P_{fa}
92,89%	1,80%

Tabla 4.24: Resultados clasificación algoritmo K-means, base de datos breast_cancer

Se obtiene en este problema una P_{fa} inferior respecto a la obtenida con el algoritmo EM, sin embargo, la P_d es inferior. Ese tipo de error es intolerable. En el caso de quitar otros tests, tras varias pruebas se ha llegado a la conclusión de que no es conveniente ya que los resultados que se obtienen empeoran la P_d. En el caso de utilizar únicamente los test 3, 4 y 6 como en el caso del algoritmo EM, los resultados son

	Número de pacientes	Porcentaje (%)
VP	207	30,31%
FP	9	1,32%
FN	32	4,69%
VN	435	63,69%

P_d	P_{fa}
86,61%	2,02%

Tabla 4.25: Resultados clasificación algoritmo K-means D=3, base de datos breast_cancer

La P_d disminuye y la P_{fa} aumenta respecto al caso de utilizar todos los test. Al utilizar $I = 3$ clases (malignos, benignos, dudosos) se obtienen los siguientes resultados

		Hipótesis		
		$H_0\|$	$H_1\|$	$H_2\|$
Real	$\|H_0$	433	0	11
	$\|H_1$	7	135	97

Tabla 4.26: Resultados clasificación algoritmo K-means I=3, base de datos breast_cancer

Se muestra una mejora en los resultados, en este caso se reducen los FN y se detectan 0 FP. Al introducir la clase de tumores dudosos, se consigue incluir en ella los FP y disminuir el número de FN. En el caso de utilizar únicamente los test 3, 4 y 6 se obtiene

		Hipótesis		
		$H_0\|$	$H_1\|$	$H_2\|$
Real	$\|H_0$	431	6	7
	$\|H_1$	17	163	59

Tabla 4.27: Resultados clasificación algoritmo K-means I=3, D=3, base de datos breast_cancer

En este caso los resultados empeoran, aumenta el número de FN y FP.

Capítulo 5: Conclusiones y líneas futuras

En este trabajo se presentan algoritmos batch y algoritmos online para la estima de máxima verosimilitud de modelos de mezclas Gaussianas. En concreto se presenta un algoritmo batch basado en el algoritmo EM y uno online basado en el algoritmo SGD. Estos algoritmos son muy flexibles, tienen la ventaja de ser aplicados a una gran variedad de problemas prácticos: se han utilizado para el modelado de ruido impulsivo, modelado de distribuciones de variables y vectores aleatorios, problemas de clústering, segmentación de imágenes, compresión y reconstrucción de datos e incluso en una base de datos real de ámbito clínico.

Por otro lado se ha explicado y utilizado otro algoritmo batch, el algoritmo K-means. Este algoritmo resuelve un problema geométrico, no tiene por objetivo la estima de máxima verosimilitud de modelos de mezclas Gaussianas. Se ha utilizado a modo de comparación en las diversas aplicaciones mostradas en el trabajo. Por ejemplo, en la aplicación de segmentación de imágenes (*Apartado 4.4*) y en el *Ejemplo 2* del apartado de clústering (*Apartado 4.3*) es preferible la utilización del algoritmo K-means. Sin embargo, se han mostrado ejemplos en los que la utilización del modelo de mezclas Gaussianas obtiene mejores resultados como en el caso del modelado de ruido impulsivo (*Apartado 4.1*), *Ejemplo 1* del apartado de clústering y en la base de datos *breast_cancer* (*Apartado 4.6*). La conclusión que se obtiene de lo anterior es que no siempre es preferible la utilización de un mismo algoritmo para resolver cualquier problema, en función del problema y del tipo de datos el algoritmo óptimo a utilizar será diferente.

El algoritmo EM presentado en el trabajo sirve para un número arbitrario de dimensiones D. En el caso del algoritmo online SGD (tanto la versión convencional como la mini-batch) la transformación de parámetros planteada sirve para el caso de $D = 1$ y su generalización a D dimensiones con matrices de covarianza diagonales (dimensiones independientes entre sí). En una línea futura se buscaría una transformación de parámetros para el caso de un número arbitrario de dimensiones en el que las dimensiones no tengan por qué ser independientes entre sí. También resultaría interesante la aplicación del modelo de mezclas Gaussianas a problemas de otro ámbito o incluso la utilización de un modelo de mezclas distinto.

Capítulo 6: Apéndices

6.1 Apéndice I: Desarrollo Paso E algoritmo EM aplicado al modelo de mezclas Gaussianas

El paso esperanza (paso E) consiste en obtener el valor esperado de la función verosimilitud para los datos completos. Dicho valor esperado puede ser calculado de la siguiente forma

$$\mathbb{E}_{\boldsymbol{z}}\left[\log L_c(\boldsymbol{\theta})|\boldsymbol{y},\, \widehat{\boldsymbol{\theta}}_i^{(n)}\right] = \mathbb{E}_{\boldsymbol{z}}\left[\sum_{j=1}^{J}\sum_{i=1}^{I} z_{ij}\,[\log\left(\hat{\alpha}_i^{(n)}\right) + \log N\left(\boldsymbol{y}_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)]\right] \qquad (A.1)$$

El término $\sum_{j=1}^{J}\sum_{i=1}^{I}[\log\left(\hat{\alpha}_i^{(n)}\right) + \log p\left(\boldsymbol{y}_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)]$ no depende de $\boldsymbol{Z}$ por lo que se puede sacar fuera

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}}&\left[\log L_c(\boldsymbol{\theta})|\boldsymbol{y},\widehat{\boldsymbol{\theta}}_i^{(n)}\right]\\ &= \sum_{j=1}^{J}\sum_{i=1}^{I}[\log\left(\hat{\alpha}_i^{(n)}\right) + \log N\left(\boldsymbol{y}_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)]\;\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|\boldsymbol{y},\widehat{\boldsymbol{\theta}}_i^{(n)}\right]\\ &= h(\boldsymbol{\theta})\end{aligned} \qquad (A.2)$$

Observando el término $\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|\boldsymbol{y},\widehat{\boldsymbol{\theta}}_i^{(n)}\right]$, solo hay una muestra j para la cual $z_{ij}=1$, por tanto se puede expresar como

$$\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|\boldsymbol{y},\widehat{\boldsymbol{\theta}}_i^{(n)}\right] = \mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|\boldsymbol{y}_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right] \qquad (A.3)$$

Por otro lado

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|\boldsymbol{y}_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right] &= N\left(z_{ij}=0\middle|\boldsymbol{y}_j,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)\cdot 0 + \cdots + N\left(z_{ij}=1\middle|\boldsymbol{y}_j,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)\cdot 1\\ &= N\left(z_{ij}=1\middle|\boldsymbol{y}_j,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)\end{aligned} \qquad (A.4)$$

Aplicando el teorema de Bayes

$$N\left(z_{ij}=1\middle|\boldsymbol{y}_j,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)=\frac{N\left(\boldsymbol{y}_j\middle|z_{ij}=1,\ \widehat{\boldsymbol{\beta}}_i^{(n)}\right)\cdot p\left(z_{ij}=1\middle|\widehat{\boldsymbol{\beta}}_i^{(n)}\right)}{N\left(\boldsymbol{y}_j\middle|\widehat{\boldsymbol{\beta}}_i^{(n)}\right)}$$

$$=\frac{N\left(\boldsymbol{y}_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)\cdot\hat{\alpha}_i^{(n)}}{\sum_{i=1}^{I}\hat{\alpha}_i^{(n)}N\left(\boldsymbol{y}_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)}=$$

$$=\frac{\hat{\alpha}_i^{(n)}\dfrac{1}{(2\pi)^{\frac{D}{2}}\left|\widehat{\boldsymbol{\Sigma}}_i^{(n)}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\boldsymbol{y}_j-\widehat{\boldsymbol{\mu}}_i^{(n)}\right)^T\widehat{\boldsymbol{\Sigma}}_i^{-1(n)}\left(\boldsymbol{y}_j-\widehat{\boldsymbol{\mu}}_i^{(n)}\right)\right)}{\sum_{i=1}^{I}\hat{\alpha}_i^{(n)}\dfrac{1}{(2\pi)^{\frac{D}{2}}\left|\widehat{\boldsymbol{\Sigma}}_i^{(n)}\right|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}\left(\boldsymbol{y}_j-\widehat{\boldsymbol{\mu}}_i^{(n)}\right)^T\widehat{\boldsymbol{\Sigma}}_i^{-1(n)}\left(\boldsymbol{y}_j-\widehat{\boldsymbol{\mu}}_i^{(n)}\right)\right)} \qquad (A.5)$$

6.2 Apéndice II: Obtención estimadores de los parámetros de un modelo de mezclas Gaussianas para el algoritmo EM y D=1

Optimización con respecto a α_i

$$\frac{dh(\boldsymbol{\theta})}{d\alpha_i}=\frac{d}{d\alpha_i}\left(\sum_{j=1}^{J}\sum_{i=1}^{I}\left[\log\left(\hat{\alpha}_i^{(n)}\right)+\log N\left(y_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)\right]\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right]\right) \qquad (A.6)$$

El término $\log N\left(y_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)$ puede desarrollarse como

$$\log N\left(y_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)=\log\left(\frac{1}{\sqrt{2\pi\hat{\sigma}_i^{2(n)}}}e^{\left(\frac{-\left(\boldsymbol{y}_j-\hat{\mu}_i^{(n)}\right)^2}{2\hat{\sigma}_i^{2(n)}}\right)}\right)=log\left(\frac{1}{\sqrt{2\pi\hat{\sigma}_i^{2(n)}}}\right)$$

$$-\frac{\left(y_j-\hat{\mu}_i^{(n)}\right)^2}{2\hat{\sigma}_i^{2(n)}}=-\frac{1}{2}\log(2\pi)-\frac{1}{2}\log\left(\hat{\sigma}_i^{2(n)}\right)-\frac{\left(y_j-\hat{\mu}_i^{(n)}\right)^2}{2\hat{\sigma}_i^{2(n)}} \qquad (A.7)$$

Aplicando los multiplicadores de Lagrange como indica Mortiz Blume [1] y teniendo en cuenta *(A.2)* se obtiene

$$\hat{\alpha}_i^{(n+1)}=\frac{1}{J}\sum_{j=1}^{J}\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right] \qquad (A.8)$$

Maximización con respecto μ_i

$$\frac{dh(\boldsymbol{\theta})}{d\mu_i} = \frac{d}{d\mu_i}\left(\sum_{j=1}^{J}\sum_{i=1}^{I}\left[\log\left(\hat{\alpha}_i^{(n)}\right) - \frac{1}{2}\log(2\pi) - \frac{1}{2}\log\left(\hat{\sigma}_i^{2\,(n)}\right) - \frac{\left(y_j - \hat{\mu}_i^{(n)}\right)^2}{2\hat{\sigma}_i^{2\,(n)}}\right]\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]\right)$$
$$= \sum_{j=1}^{J}\left(\frac{2(y_j - \hat{\mu}_i^{(n)})}{2\hat{\sigma}_i^{2\,(n)}}\right)\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]$$
$$= \frac{1}{\hat{\sigma}_i^{2\,(n)}}\sum_{j=1}^{J}\left(y_j - \hat{\mu}_i^{(n)}\right)\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right] = 0 \qquad (A.9)$$

$$\frac{1}{\hat{\sigma}_i^{2\,(n)}}\sum_{j=1}^{J}\left(y_j - \hat{\mu}_i^{(n)}\right)\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]$$
$$= \frac{1}{\hat{\sigma}_i^{2\,(n)}}\left(\sum_{j=1}^{J} y_j\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right] - \sum_{j=1}^{J}\hat{\mu}_i^{(n)}\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]\right) = 0 \qquad (A.10)$$
$$\sum_{j=1}^{J} y_j\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right] = \sum_{j=1}^{J}\hat{\mu}_i^{(n)}\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]$$

Finalmente se obtiene

$$\hat{\mu}_i^{(n+1)} = \frac{\sum_{j=1}^{J} y_j\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]}{\sum_{j=1}^{J}\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]} \qquad (A.11)$$

Maximización con respecto σ_i^2

$$\frac{dh(\boldsymbol{\theta})}{d\sigma_i^2} = \frac{d}{d\sigma_i^2}\left(\sum_{j=1}^{J}\sum_{i=1}^{I}\left[\log\left(\hat{\alpha}_i^{(n)}\right) - \frac{1}{2}\log(2\pi) - \frac{1}{2}\log\left(\hat{\sigma}_i^{2\,(n)}\right) - \frac{\left(y_j - \hat{\mu}_i^{(n)}\right)^2}{2\hat{\sigma}_i^{2\,(n)}}\right]\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right]\right) \qquad (A.12)$$
$$= \sum_{j=1}^{J}\left(-\frac{1}{2\hat{\sigma}_i^{2\,(n)}} + \frac{\left(y_j - \hat{\mu}_i^{(n)}\right)^2}{2\hat{\sigma}_i^{2\,(n)}\cdot\hat{\sigma}_i^{2\,(n)}}\right)\mathbb{E}_{\boldsymbol{z}}\left[z_{ij}|y_j, \widehat{\boldsymbol{\theta}}_i^{(n)}\right] = 0$$

$$\frac{1}{2\hat{\sigma}_i^{2(n)}}\sum_{j=1}^{J}\left(-1+\frac{\left(y_j-\hat{\mu}_i^{(n)}\right)^2}{\hat{\sigma}_i^{2(n)}}\right)\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right]$$

$$=\sum_{j=1}^{J}\left(-1+\frac{\left(y_j-\hat{\mu}_i^{(n)}\right)^2}{\hat{\sigma}_i^{2(n)}}\right)\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right] \qquad (A.13)$$

$$\frac{1}{\hat{\sigma}_i^{2(n)}}\sum_{j=1}^{J}\left(y_j-\hat{\mu}_i^{(n)}\right)^2\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right]=\sum_{j=1}^{J}\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right]$$

Despejando σ_i^2 se obtiene:

$$\hat{\sigma}_i^{2(n+1)}=\frac{\sum_{j=1}^{J}\left(y_j-\hat{\mu}_i^{(n+1)}\right)^2\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right]}{\sum_{j=1}^{J}\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right]} \qquad (A.14)$$

donde

$$\mathbb{E}_{\boldsymbol{z}}\left[\,z_{ij}|y_j,\widehat{\boldsymbol{\theta}}_i^{(n)}\right]=\frac{N\left(y_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)\cdot\hat{\alpha}_i^{(n)}}{\sum_{i=1}^{I}\hat{\alpha}_i^{(n)}\,N\left(y_j\middle|C=i,\widehat{\boldsymbol{\beta}}_i^{(n)}\right)} \qquad (A.15)$$

Referencias

[1] Moritz Blume, *Expectation maximization: A gentle introduction*, 2008

[2] G. McLachlan and K. Basford, *Mixture models: Inference and Application to Clustering*, New York: Marcel Dekker, 1988

[3] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: John Wiley & Sons, 2000

[4] D. Titterington, A. Smith, and U. Makov, *Statistical Analysis of Finite Mixture Distributions*. Chichester, U.K.: John Wiley & Sons, 1985

[5] Jeff A. Bilmes, *A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden markov models*, International computer science institute, Berkeley CA, 94704 and computer science division department of electrical engineering and computer science, 1997

[6] M. Whindham and A. Cutler, *Information Ratios for validating mixture analysis*, J. Am, Statistical Assoc, vol 87, pp 1188-1192, 1992

[7] J. Banfield and A. Raftery, *Model-Based Gaussian and Non-Gaussian Clustering*, Biometrics, vol. 49, pp 803-821, 1993

[8] C. Biernacki, G. Celeux, and G. Govaert, *Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models*, Comput. Stat. Data Anal., vol. 41, no. 3-4, pp. 561–575, Jan. 2003

[9] V. Melnykov and I. Melnykov, *Initializing the EM algorithm in Gaussian mixture models with an unknown number of components*, Computational Statistics & Data Analysis, Nov. 2011

[10] W. Kwedlo, *A New Method for Random Initialization of the EM Algorithm for Multivariate Gaussian Mixture Learning*, in Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013. Springer International Publishing, 2013, pp. 81–90

[11] D. Arthur and S. Vassilvitskii, *k-means++: the advantages of careful seeding*, in SODA, N. Bansal, K. Pruhs, and C. Stein, Eds. SIAM, 2007, pp. 1027–1035

[12] Mario A.T. Figueiredo, *Unsupervised Learning of Finite Mixture Models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no.3, 2002

[13] Schäfer and Strimmer, *A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics*, Juliane Schafer and Korbinian Strimmer, Statistical applications in genetics and molecular biology, vol. 4, issue 1, article 32, 2005

[14] N. Kambhatla and T. K. Leen, *Dimension reduction by local principal component analysis*. Neural Computation, vol. 9, no. 7:1793-1516, 1997

[15] Hartigan, J. A.; Wong, M. A.; *Algorithm AS 136: A K-means Clustering Algorithm*, Journal of the Royal Statistical Society Series C, vol. 28, pp. 100-108, 1979

[16] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; *An efficient K-means clustering algorithm: Analysis and implementation*, IEEE Trans.Pattern Analysis and Machine Intelligence, vol. 24, pp. 881-892, 2002

[17] Teukolsky, SA; Vetterling, WT; Flannery, BP; *Section 16.1. Gaussian Mixture Models and K-means Clustering*, Numerical Recipes: The Art of Scientific Computing. New York Cambridge University Press, 2007

[18] Peter 17] J. Rousseeuw, *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*. *Computational and Applied Mathematics* 20: 53–65, 1987

[19] McLachlan and Peel, *Finite Mixture Models*, Wiley-Interscience; 1 edition (October 2, 2000)

[20] S. Shalev-Shwartz, *Online learning: Theory, algorithms, and applications.* Technical report, The Hebrew University, PhD thesis, 2007

[21] Bottou, Léon, *Online Algorithms and Stochastic Approximations*, Online Learning and Neural Networks. Cambridge University Press, ISBN 978-0-521-65263-6, 1998

[22] Ferguson, Thomas S, *An inconsistent maximum likelihood estimate*, Journal of the American Statistical Association 77, pp. 831-834, 1982

[23] Mordecai, Avriel, Nonlinear Programming: Analysis and Methods, Dover Publishing, ISBN 0-486-43227-0, 2003

[24] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. *Better mini-batch algorithms via accelerated gradient methods*. In NIPS, volume 24, pages 1647–1655, 2011

[25] Li, Mu, et al. *Efficient mini-batch training for stochastic optimization.* Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014

[26] Frederic P. Miller, Agnes F. Vandome, John Mc Brewster, *Gradient Descent*, Alphascript Publishing, 2010

[26] Base de datos breast_cancer disponible en:
http://www.ics.uci.edu/~mlearn/MLRepository.html