Taller Modelos de Mezclas y Árboles de Decisión

Julián D. Arias Londoño
Departamento de Ingeniería de Sistemas
Universidad de Antioquia, Medellín, Colombia
julian.ariasl@udea.edu.co

March 28, 2016

1 Marco teórico

1.1 Modelos de Mezcla de funciones Gaussinas (GMM)

Un modelo de mezcla de funciones Gausianas (En inglés: Gaussian Mixture Model - GMM) es una función de densidad de probabilidad paramétrica representada como una suma sopesada de componentes Gausianas [1]. Formalmente un GMM puede ser expresado como:

$$p\left(\mathbf{x}|\Theta\right)=\sum_{i=1}^{M}w_i\mathcal{N}\left(\mathbf{x}|\mu_i,\Sigma_i\right) \tag{1}$$

donde $\mathbf{x}$ es una ϱ-dimensional vector de datos continuos (i.e. mediciones o características), $w_i, i=1,...,M$ son los pesos de las mezclas, and $\mathcal{N}\left(\mathbf{x}|\mu_i,\Sigma_i\right)$ son las componentes Gausianas. Cada componente es una función Gausiana multivariada de dimensión ϱ, de la forma:

$$\mathcal{N}\left(\mathbf{x}|\mu_i,\Sigma_i\right)=\frac{1}{\left(2\pi\right)^{\varrho/2}\left|\Sigma_i\right|^{1/2}}\exp\left\{-\frac{1}{2}\left(\mathbf{x}-\mu_i\right)^T\Sigma_i^{-1}\left(\mathbf{x}-\mu_i\right)\right\} \tag{2}$$

con vector de medias μ_i y matriz de covarianza Σ_i. Los pesos de las mezclas satisfacen la condición $w_i\geq 0$ y $\sum_{i=1}^{M}w_i=1$. El GMM completo está parametrizado por los vectores de medias, las matrices de covarianza y los pesos de las mezclas para todas las funciones de densidad que componen la mezcla. Estos parámetros son representados por la notación:

$$\Theta=\left\{w_i,\mu_i,\Sigma_i\right\},\quad i=1,\ldots,M \tag{3}$$

Los GMMs son modelos parámetricos útiles cuando el conjunto de datos que queremos modelar están agrupados en diferentes conglomerados (ver Fig. 1).
De acuerdo a lo mostrado en la figura 1b, cada conglomerado estaría representado por un vector de medias y una matriz de covarianza. Sin embargo, si

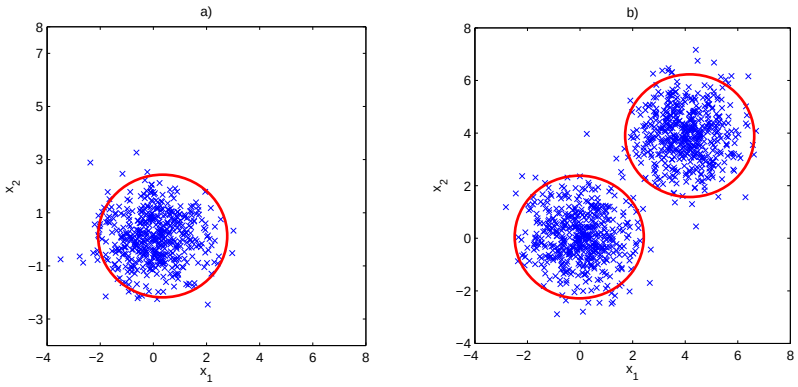

Figure 1: Distribución de datos bidimensionales a) un conglomerado b) dos conglomerados

deseamos saber la probabilidad de un determinado $\mathbf{x}$ de acuerdo a la distribución completa, de acuerdo al teorema de Bayes deberíamos estimar:

$$p(C_i|\mathbf{x}) = \frac{p(\mathbf{x}|C_i)p(C_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|C_i)p(C_i)}{\sum\limits_{\forall i} p(\mathbf{x}|C_i)p(C_i)} \tag{4}$$

para lo cual necesitaríamos conocer el valor de $p(C_i)$. Si conocemos el conglomerado al cual pertenece cada uno de los $\mathbf{x}_i$'s, la estimación de máxima verosimilitud es muy simple. Podríamos hacer estimaciones de máxima verosimilitud de cada componente Gausiana independientemente y posteriormente maximizar la función de verosimilitud con respecto a los $p(C_i)$; sin embargo, ¿cómo sabemos cuáles $\mathbf{x}_i$ usamos para estimar los parámetros de la primera componente (μ_1 y Σ_1) y cuáles para estimar los parámetros de la segunda componente (μ_2 y Σ_2)?[1]

El criterio de máxima verosimilitud visto en las clases anteriores puede aplicarse también en este caso:

$$\begin{aligned} \max_{\Theta} \log \prod_{j=1}^{N} p\left(\mathbf{x}_j|\Theta\right) &= \max_{\Theta} \log \prod_{j=1}^{N} \sum_{i=1}^{M} w_i \mathcal{N}\left(\mathbf{x}_j|\mu_i, \Sigma_i\right) \\ &= \max_{\Theta} \sum_{j=1}^{N} \log \sum_{i=1}^{M} w_i \mathcal{N}\left(\mathbf{x}_j|\mu_i, \Sigma_i\right) \end{aligned} \tag{5}$$

donde $w_i = p(C_i)$ que para lo concerniente al problema de estimación es equivalente a $w_i = p(C_i|\mathcal{X})$. Derivando la función de verosimilitud con respecto a μ_i

[1] En términos generales el número de componentes es un parámetro del modelo y corresponde al valor de M en la ecuación (1).

obtenemos:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mu_i} &= \sum_{t=1}^{n} \frac{1}{p\left(\mathbf{x}_t|\Theta\right)} \frac{\partial}{\partial \mu_i} \sum_{i=1}^{M} p\left(\mathbf{x}_t|C_i,\Theta_i\right) p(C_i) \\ &= \sum_{t=1}^{n} \frac{p(C_i)}{p\left(\mathbf{x}_t|\Theta\right)} \frac{\partial}{\partial \mu_i} p\left(\mathbf{x}_t|C_i,\Theta_i\right) \end{aligned} \tag{6}$$

Por el teorema de Bayes y teniendo en cuenta que

$$p\left(\mathbf{x}_t|\Theta\right) = \sum_{i=1}^{M} p\left(\mathbf{x}_t|C_i,\Theta_i\right) p(C_i) \tag{7}$$

la ecuación (6) se convierte en:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mu_i} &= \sum_{j=1}^{N} \frac{p(C_i)}{p\left(\mathbf{x}_j|\Theta\right)} p\left(\mathbf{x}_j|C_i,\Theta_i\right) \frac{\partial}{\partial \mu_i} \log p\left(\mathbf{x}_j|C_i,\Theta_i\right) \\ &= \sum_{j=1}^{N} p(C_i|\mathbf{x}_j,\Theta) \frac{\partial}{\partial \mu_i} \log p\left(\mathbf{x}_j|C_i,\Theta_i\right) \end{aligned} \tag{8}$$

Reemplazando $p\left(\mathbf{x}_t|C_i,\Theta_i\right)$ por una función normal univariada obtenemos:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mu_i} &= \sum_{j=1}^{N} p(C_i|x_j,\Theta) \frac{\partial}{\partial \mu_i} \log \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-\left(x_j-\mu_i\right)^2}{2\sigma_i^2}\right) \\ &= \sum_{j=1}^{N} p(C_i|x_j,\Theta) \frac{\partial}{\partial \mu_i} \left(\frac{-\left(x_j-\mu_i\right)^2}{2\sigma_i^2}\right) \\ &= \sum_{j=1}^{N} p(C_i|x_j,\Theta) \left(-\frac{1}{2\sigma_i^2} \frac{\partial}{\partial \mu_i} \left(x_j-\mu_i\right)^2\right) \\ &= \sum_{t=1}^{n} p(C_i|x_j,\Theta) \frac{\left(x_j-\mu_i\right)}{\sigma} = 0 \end{aligned} \tag{9}$$

despejando obtenemos,

$$\hat{\mu}_i = \frac{\sum_{j=1}^{N} p(C_i|x_j,\Theta) x_j}{\sum_{j=1}^{N} p(C_i|x_j,\Theta)} \tag{10}$$

Sin embargo $p(C_i|x_j,\Theta)$ es desconocida. El algoritmo para la solución de este problema de estimación se conoce como EM (Esperanza-Maximización), éste es un algoritmo iterativo en el cual primero se estiman las probabilidades $p(C_i|x_j,\Theta)$ de acuerdo al conjunto de parámetros Θ actuales y posteriormente

se encuentran los parámetros maximizando la verosimilitud.

El Pase E del algoritmo consiste entonces en calcular las "clases esperadas" de todos los puntos para cada conglomerado, en la iteración t. Considerando una función Gaussiana multivariada este paso correspondería a estimar:

$$p(C_i|x_t, \Theta(k)) = r_{ji} = \frac{\omega_i^{(t-1)} \mathcal{N}(\mathbf{x}_j|\mu_i^{(t-1)}, \Sigma_i^{(t-1)})}{\sum_{k=1}^{M} \omega_k^{(t-1)} \mathcal{N}(\mathbf{x}_j|\mu_k^{(t-1)}, \Sigma_k^{(t-1)})} \tag{11}$$

El Paso M correspondería a calcular los parámetros Θ que maximizan la verosimilitud de los datos dadas las membresías anteriores:

$$\omega_i^{(t)} = \frac{1}{N} \sum_{j=1}^{N} r_{ji} = \frac{r_i}{N}$$

Recordando el resultado de la ecuación (10), entonces

$$\hat{\mu}_i^{(t)} = \frac{\sum_{j=1}^{N} r_{ij} \mathbf{x}_j}{r_i}$$

y realizando la derivada de la función de verosimilitud con respecto Σ obtendremos:

$$\hat{\Sigma}_i^{(t)} = \frac{\sum_{j=1}^{N} r_{ij} \mathbf{x}_j \mathbf{x}_j^T}{r_i} - \mu_i \mu_i^T$$

Las ecuaciones anteriores no constituyen una solución cerrada para los parámetros del modelo de mezclas porque las probabilidades condicionales $p(C_i|x_t, \Theta)$ dependen de dichos parámetros de una manera compleja. Sin embargo, este resultado constituye un esquema iterativo simple para encontrar una solución al problema de máxima verosimilitud.
Un modelo GMM por sí sólo también es una herramienta que permite agrupar un conjunto de datos en diferentes subgrupos, razón por la cual el modelo GMM puede ser usado también como algoritmo de agrupamiento no supervisado. El método de agrupamiento (clustering) más usado en problemas de Minería de Datos y Aprendizaje de Máquina, es el algoritmo k-means, el cual será descrito a continuación:

	Algoritmo: *K-means*
1	Inicializar μ_i: Se puede hacer escogiendo M puntos aleatorios.
2	**Repita**
3	Asigne cada punto al cluster con valor medio más cercano: $z_j = \arg\max_i \|\mathbf{x}_j - \mu_i\|^2$
4	Actualice el centro de cada cluster calculando la media de todos los puntos asignados a cada uno: $\mu_i = \frac{1}{N_i} \sum_{j:z_j=i} \mathbf{x}_j$
5	**Hasta**: Convergencia

El algoritmo K-means y el modelo GMM ajustado a través del algoritmo EM tienen muchas similitudes. En el primer caso se selecionan centroides aleatorios mientras que en el segundo, debido a la suposición de la forma de la distribución, se deben asumir no solo los centroides (en este caso las medias de cada componente Gaussiana), sino también las varianzas y los pesos de cada componente. En el segundo paso del algoritmo K-means se mide la distancia de cada muestra a cada uno de los centroides para establecer la pertenencia de cada muestra, la cual se decide únicamente a través de la distancia más corta. En el caso del modelo GMM la pertenencia se decide calculando la probabilidad de cada muestra en cada una de las componentes. Por último, el algoritmo K-means recalcula el centroide de cada grupo como la media de las muestras que pertenecen al mismo grupo, mientras que por otro lado en el modelo GMM es necesario recalcular no sólo las medias, sino las covarianzas y los pesos. En este último caso, como cada muestra no pertenece a un solo grupo sino que tiene una probabilidad de pertenecer a cada grupo, todos los parámetros del modelo para cada componente se estiman asignando a cada muestra un peso proporcional a dicho valor de probabilidad.

1.2 Árboles de decisión

Los árboles de decisión son métodos naturales e intuitivos de realizar la predicción de una variable (clasificación o regresión), a través de una secuencia de preguntas que van subdividiendo y a la vez simplificando el espacio en el cual se encuentra el conjunto de datos, en pocas palabras, es un aproximación en la cual una decisión compleja se rompe en una serie de decisión más simples (ver fig 2).

El algoritmo básico de aprendizaje de árboles de decisión conocido como ID3 [2], construye la estructura en árbol de arriba hacia abajo. Los aspecto básicos a determinar dentro del algoritmo son:

- ¿cuál atributo debería ser evaluado en cada nodo?
- Seleccionar una regla de partición para cada nodo interno (determinación de características y umbrales a ser usados en cada nodo)
- Determinar cuáles son los nodos terminales, es decir, para cada nodo se debe decidir si se sigue partiendo o si se convierte en nodo terminal.
- Finalmente, se debe generar un procedimiento para realizar la predicción en los nodos terminales.

Para poder determinar cual atributo usar es necesario definir una medida que cuantifique la calidad de la partición. Teniendo en cuenta que para el caso de clasificación el nodo ideal sería uno en el que sólo hubiesen muestras de una misma clase (un nodo puro), una de las medidas más usadas es precisamente una que trata de medir la impureza de un nodo.

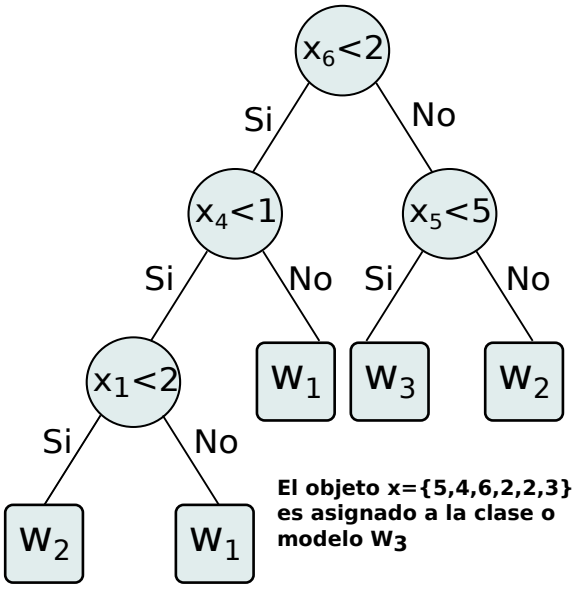

Figure 2: Árbol de decisión construido para un problema de clasificación de 3 clases, en el cual cada muestra está representada por 6 características.

La medida clásica de impureza de una partición U está basada en la medida de entropía:

$$I(U) = -\sum_j P(w_j) \log_2 P(w_j)$$

donde $P(w_j)$ es la probabilidad de la clase j en el espacio U. Esta medida será cero si todas las muestras de la partición pertenecen a una misma clase (nodo puro) y será igual a uno si existe un número igual de muestras de cada clase en la participación (máxima incertidumbre).

Teniendo en cuenta la medida anterior, se puede definir una medida de calidad de partición proporcionada por un atributo conocida como *ganancia de información*. La ganacia de información es la reducción esperada en la impureza de información debida a la partición del conjunto de muestras de acuerdo a un atributo a. La ganacia de información se puede expresar como:

$$Gain(U, a) = I(U) - \sum_{v \in valores(a)} \frac{|U_v|}{|U|} I(U_v)$$

donde $valores(a)$ es el conjunto de todos los posibles valores de a y U_v es el subconjunto de U para el cual el atributo a toma el valor v. $Gain(U, a)$ es en pocas palabras la reducción esperada en la entropía causada por el conocimiento del atributo a.

La medida de ganancia de información requiere establecer el número de posibles valores que toma el atributo a evaluar, sin embargo es muy común encontrar atributos que toman valores reales y por ende no pueden ser evaluados a través de la medida anterior.

En este caso es posible definir una malla de umbrales y evaluar la reducción en la impureza debida a la partición:

$$Gain(U, a) = I(U) - (I(U_L)P_L + I(U_R)P_R)$$

donde U_L corresponde al subconjunto asignado al nodo hijo izquierdo y U_R el subconjunto asignado al nodo hijo derecho. P_L y P_R corresponden a las probabilidades de cada nodo hijo de acuerdo al conjunto de entrenamiento. En el algoritmo de aprendizaje básico se realiza la partición de las muestras hasta alcanzar impureza igual a 0. Sin embargo esto puede conducir a problemas de sobre ajuste (cada nodo conteniendo una única muestra de entrenamiento). Una forma alternativa es definir un límite deceado de impureza y permitir que el árbol crezca hasta alcanzar un nivel de impureza inferior al deseado. Sin embargo, muchas aproximaciones permiten que el árbol crezca hasta alcanzar una impureza de cero y posteriormente se aplica un procedimiento de podado.

Existen un sinnúmero de algoritmos para reducir el número de nodos y evitar el sobre ajuste del modelo. Uno de los más básicos es llamado *podado de error reducido* [2]. En éste todos los nodos de decisión son candidatos a ser reducidos y reemplazados por nodos terminales. La clase asignada en el nuevo nodo es la clase a la cual pertenezcan la mayoría de las muestras asociadas al nodo.

Un nodo de decisión es suprimido únicamente si el árbol podado tiene el mismo desempeño que el árbol original sobre un conjunto de validación. el podado se realiza sobre un nodo a la vez y se realiza hasta que no pueda ser eliminado ningún nodo sin perjudicar el desempeño del sistema.

1.3 Comité de Máquinas

Es usual en el aprendizaje de máquina no dejar la responsabilidad de la decisión a un solo modelo, sino por el contrario combinar la decisión de varios para tomar una decisión final. Existen muchas maneras de combinar clasificadores, una de ellas es conocida como Bagging "Bootstrap Agreggating", consiste en realizar B muestreos (con substitución) a partir del conjunto de entrenamiento y entrenar un clasificador diferente a partir de cada uno de los conjuntos de muestras.

La decisión final se toma dependiendo de si el problema es de clasificación o de regresión, si el problema es de clasificación se utiliza la regla del mayor voto (moda) y si por el contrario el problema es de regresión se usa el promedio de las predicción de cada clasificador.

1.4 Random Forest

En el caso particular de los árboles de regresión o clasificación, existe una modificación del método anterior conocida como Random Forest, la cual consiste no sólo en crear un grupo de árboles (bosque) B, sino también en incluir un componente aleatorio en la partición que se realiza en cada nodo.

En un Random Forest el conjunto de variables que se evalúan en cada nodo se escoge de manera aleatoria del conjunto de variables originales, es decir, antes de analizar cuál variable usar para hacer la partición se escogen de manera aleatorioa m variables y la escogencia de la partición se realiza únicamente usando dicho subconjunto. La decisión final se toma a partir de la combinación de las decisiones de los B árboles entrenados.

2 Ejercicios

1. Adjunto a este taller encontrará los archivos: `Main.m, entrenarGMM.m, entrenarTREE.m, entrenarFOREST.m, testGMM.m`, `testTREE.m` y `testFOREST.m`. El archivo `Main.m` es el script principal, el cual lleva a cabo un experimento de clasificación utilizando los datos de la base de datos Vertebral Column disponible en el UCI Machine Learning Repository. La base de datos consta de 310 muestras pertenecientes a 3 clases. Cada muestra está descrita por 6 atributos. Una vez corra el Script principal se le solicitará ingresar el numeral del punto que desea resolver (es decir 4,5 ó 6). Analice con cuidado el script y comprenda como esta construido.

2. Descargue el toolbox Netlab desde el siguiente enlace

   `http://www.aston.ac.uk/eas/research/groups/ncrg/resources/netlab/`

   Como parte del toolbox se incluyen una serie de Scripts `demgmm1.m, demgmm2.m,..., demgmm5.m`, en los cuales se usan GMMs para modelar diferentes conjuntos de datos, haciendo especial énfasis en las restricciones que se imponen sobre la matriz de covarianza del modelo (completa, diagonal y esférica), lo cual cambia la forma que la funciones de densidad pueden tomar.

   El toolbox utiliza una estructura particular para almacenar los modelos entrenados, por esa razón las funciones a tener en cuenta son las siguientes:

   - `gmm`: Crea una estructura con campos vacíos para almacenar el modelo
   - `gmminit`: Inicializa el modelo a partir del conjunto de datos. Pueden escoger dos alternativas: inicialización aletoria o a través del algoritmo K-means.
   - `gmmem`: Lleva a cabo el entrenamiento del modelo.
   - `gmmpost`: Estima la probabilidad de que una muestra (o un conjunto) pertenezca a cada una de las componentes del modelo, es decir que esta función devuelve un vector de valores por cada muestra a validar, donde la longitud del vector es igual al número de componentes en el modelo. Si se evalúan varias muestras la función devolverá una matriz.
   - `gmmprob`: Esta función calcula la probabilidad de una muestra en el modelo completo, Eq. (1), en lugar de evaluar las componentes individuales como en el caso anterior.

   Revise con cuidado la descripción de todas las funciones listadas, los parámetros de llamada y las variables retornadas. Utilice los demos para analizar el llamado de las funciones si tiene alguna duda al respecto.

   Para poder resolver el problema de clasificación con mezcla de funciones Gaussianas debe completar los archivos `entrenarGMM.m` y `testGMM.m`,

donde se deben incluir las lineas indicadas con las funciones para entrenar y validar el modelo.

Una vez haya completado el código, ejecute varias veces el proceso de entrenamiento y evaluación cambiando el número de componentes en la mezcla, y complete la siguiente tabla con los valores de la eficiencia y el intervalo de confianza:

Matriz de Covarianza	**Mezclas**	**Eficiencia**	**Intervalo de Confianza**
Completa	1		
	2		
	3		
	4		
	5		
	6		
Diagonal	1		
	2		
	3		
	4		
	5		
	6		
Esférica	1		
	2		
	3		
	4		
	5		
	6		

Responda las siguientes preguntas:

- ¿Por qué cree que se obtiene este resultado con un matriz de covarianza completa?

  R:/

- ¿Por qué el modelo GMM es un modelo generativo?

  R:/

3. Revise la ayuda de Matlab de las siguientes funciones relacionadas con la creación, evaluación, poda y visualización de un árbol de decisión: `fitctree, predict, prune, view`. Revise con cuidado la descripción de dichas funciones, los parámetros de llamada y las variables retornadas.

   Para poder resolver el problema de clasificación con arboles de decisión debe completar los archivos `entrenarTREE.m y testTREE.m`, donde se deben completar las lineas indicadas con las funciones para entrenar un modelo y generar la predicción de una muestra.

   Una vez haya completado el código, ejecute varias veces el proceso de entrenamiento y evaluación podando varios niveles del árbol, y complete la siguiente tabla con los valores de la eficiencia y el intervalo de confianza:

Nivel de poda	**Eficiencia**	**Intervalo de Confianza**
Sin poda		
1		
2		
3		
4		
5		

   * Utilice la función `prune` para graficar los árboles. Analice cual es el cambio que sufre un árbol cuando es podado.

   Responda las siguientes preguntas:

   - Analice el efecto del nivel de poda en el árbol y explique lo observado.

     R:/

4. Revise la ayuda de Matlab de la función `TreeBagger`, la cual permite entrenar un comité de árboles de decisión o Random Forest.

   Para poder resolver el problema de clasificación con Random Forest debe completar los archivos `entrenarFOREST.m y testFOREST.m`, donde se deben completar las lineas indicadas con las funciones para entrenar un modelo y generar la predicción de una muestra.

   Una vez haya completado el código, ejecute varias veces el proceso de entrenamiento y evaluación variando el número de árboles, y complete la siguiente tabla con los valores de la eficiencia y el intervalo de confianza:

Número de árboles	**Eficiencia**	**Intervalo de Confianza**
10		
20		
30		
40		
50		
500		

   Responda las siguientes preguntas:

   - ¿Cuál de los tres modelos utilizados requiere mayor carga computacional? Para responder la última pregunte haga uso de las funciones `tic, toc` a través de las cuales puede almacenar los tiempos de ejecución de algún segmento de código.

     R:/

5. * Cree de manera artificial tres grupos de muestras con distribución Gaussiana, con diferentes medias y lleve a cabo el agrupamiento de las misma a través del algoritmo K-means. Modifique la función para visualizar el ajuste de los centroides y la conformación de los clusters durante las iteraciones del algoritmo.

   - ¿Qué medida se puede usar para determinar el número óptimo de clusters en un algoritmo no supervisado?
     R:/

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. New Jersey, NY, USA: Wiley-Interscience, 2nd ed., 2000.