Taller de introducción a la programación: Trabajo Practico 2

Ph. D. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,

31 de mayo de 2022

El presente trabajo práctico pretende introducir al estudiante en el proceso de análisis, diseño e implementación de un sistema de software orientado objetos.

- **Modo de trabajo**: grupos de 3 personas
- **Fecha de entrega:** 5 de Junio del 2022
- **Tipo de entrega:** digital, por medio de la plataforma TEC-digital.

Parte I
Motivación

En la presente tarea programada, se implementará el algoritmo de *K-medias* para realizar el amontonamiento o *clustering* automático de un conjunto de datos. Los algoritmos de amontontonamiento son algoritmos de clasificación. Los algoritmos de clasificación tienen por finalidad asignar a un dato $\vec{x}_i \in \mathbb{R}^D$ cuya clase $k = 1, \ldots, K$ es desconocida (donde K es el total de clases existente en el dominio del problema), una etiqueta $t_i \in [1, \ldots, K]$. Los algoritmos de clasificación necesitan usualmente un número grande de N muestras, denotadas usualmente en una matriz de muestras de entrenamiento:

$$X = \begin{bmatrix} | & \cdots & | \\ \vec{x}_1 & \cdots & \vec{x}_N \\ | & \cdots & | \end{bmatrix}.$$

Cuando para esas muestras se conoce de antemano a cual clase pertenece cada una, se dice que las muestras están etiquetadas, por lo que se cuenta con el vector de etiquetas correspondiente:

$$T = \begin{bmatrix} t_1 & \ldots & t_N \end{bmatrix}.$$

Los algoritmos que necesitan del arreglo de etiquetas T para realizar la clasificación de una nueva muestra $\vec{v} \in \mathbb{R}^D$ se denominan **algoritmos de clasificación supervisados** y los que prescinden de tal arreglo de etiquetas, son referidos como **algoritmos de clasificación no supervisados**. El algoritmo a implementar en la presente tarea programada, correponde a un algoritmo de clasificación no supervisado.

Para ilustrar mejor el problema general de clasificación no supervisada, se presenta el siguiente problema: se desea estimar si un día es de verano ($k = 1$) o de invierno ($k = 2$) , por lo que existen entonces $K = 2$ clases, según los datos de humedad y temperatura (), con lo que cada muestra $\vec{x}_i \in \mathbb{R}^2$ tiene dos componentes. La siguiente es la tabla de las muestras:

Dada tal tabla de muestras, la matriz X de datos de entrenamiento está dada por:

$$X = \begin{bmatrix} 50 & 42 & 80 & 70 \\ 32 & 29 & 15 & 19 \end{bmatrix},$$

Observacion	Húmedad (%)	Temperatura (C°)	Clima
1	50	32	Verano
2	42	29	Verano
3	80	15	Invierno
4	70	19	Invierno

Cuadro 1: Muestras recolectadas en distintos días.

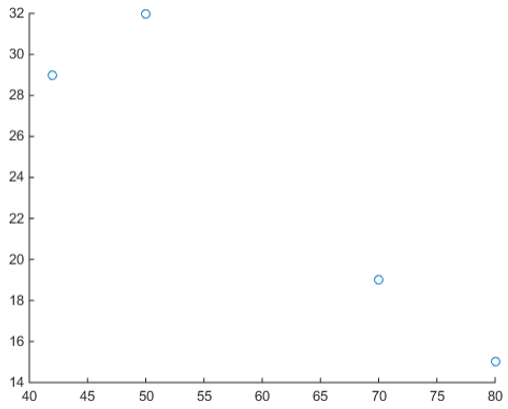

Figura 1: Graficación de los datos en la Tabla 1.

puesto que $\vec{x}_1 = \begin{bmatrix}50 & 32\end{bmatrix}^T$, $\vec{x}_2 = \begin{bmatrix}42 & 29\end{bmatrix}^T$, $\vec{x}_3 = \begin{bmatrix}80 & 15\end{bmatrix}^T$ y $\vec{x}_4 = \begin{bmatrix}70 & 19\end{bmatrix}^T$. Como se comentó anteriormente, los algoritmos de clasificación no supervisada prescinden del arreglo de etiquetas T, el cual en este caso corresponde si la muestra tomada en un día de verano verano ($k = 1$) o de invierno ($k = 2$). Aunque el arreglo T no es necesario para el algoritmo de entrenamiento, él mismo es necesario para la etapa de validación, que se refiere a conocer la precisión del algoritmo. Para las muestras tomadas, según los datos muestreados, el arreglo T viene dado por:

$$T = \begin{bmatrix}1 & 1 & 2 & 2\end{bmatrix}.$$

La graficación de los datos en X permite visualizar más fácilmente el problema de la clasificación, con el eje x representando la húmedad, y el eje y representando la temperatura, y se muestra en la Figura 1.

Un algoritmo de clasificación no supervisada, debe encontrar los K montones o *clusters* de forma que el error de clasificación sea cero. Para este caso, la Figura 2, ilustra los *clústers* que debería encontrar el algoritmo para lograr un error de cero. El algoritmo de clasificación, estima automáticamente el arreglo T, para el conjunto de datos X, por lo que las muestras cuya etiqueta no es conocida puede mezclarse en los datos cuyas etiquetas son conocidas, de modo que se pueda estimar la etiqueta de tal muestra nueva.

Parte II

El algoritmo K-medias para el amontonamiento

A manera de introducción práctica del problema de amontonamiento no supervisado, nos aprestamos a analizar e implementar el popular algoritmo K-medias. A partir de la notación introducida en la sección anterior, se resumen los aspectos importantes del algoritmo K-medias.

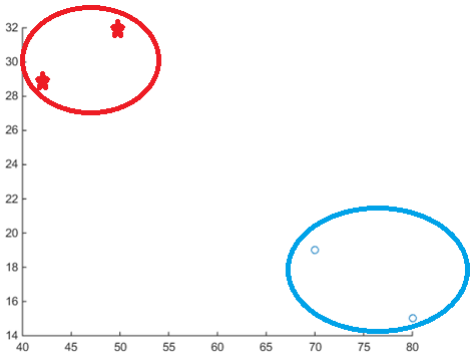

Figura 2: Clasificación con error cero, usando un esquema no supervisado.

- **Objetivo general:** Partir el conjunto de datos de entrada

$$X = \begin{bmatrix} | & \dots & | \\ \vec{x}_1 & \dots & \vec{x}_N \\ | & \dots & | \end{bmatrix}$$

en K nuevos conjuntos X_k, cuya **varianza intra-clase sea la menor posible**. Para lograrlo, el algoritmo de K-medias propone en P iteraciones ajustar las K medias de cada cluster k, minimizando la anteriormente mencionada varianza intra-clase. El número de clases K y de iteraciones a ejecutar P son conocidas. Las siguientes son definiciones utilizadas en la construcción del algoritmo.

  - **Cluster:** Conjunto de datos X_k, los cuales tienen una distancia *pequeña* entre ellos (distancia intra-cluster), respecto a los datos de otros clusters X_J (distancia inter-cluster). A cada cluster X_k le corresponde una *muestra media* $\tilde{\mu}_k$ de las muestras en tal cluster, definida como:

$$\vec{\mu}_k = \frac{\sum_{n=1}^{N} W_{n,k}\vec{x}_n}{\sum_{n=1}^{N} W_{n,k}},$$

    donde:

    - donde la matriz de pesos binaria W está dada por:

$$W = \begin{bmatrix} W_{1,1} & \dots & W_{1,K} \\ \vdots & \ddots & \vdots \\ W_{N,1} & \dots & W_{N,K} \end{bmatrix}$$

      de dimensión $N \times K$ (una fila por muestra, una columna por clase), determina la pertenencia de la muestra n a la clase k, de modo que:

$$W_{n,k} = \begin{cases} 1 & k = \text{argmín}_j d\left(\vec{x}_n - \vec{\mu}_j\right) \\ 0 & \text{de lo contrario} \end{cases}.$$

      En otras palabras, la matriz de pesos $W_{n,k}$ se asigna como $W_{n,k} = 1$ para una muestra $\vec{x}_n$ cuyo cluster con media $\vec{\mu}_k$ es el más cercano y es $W_{n,k} = 1$ en caso contrario. La función de distancia d puede ser la función de distancia Euclidiana:

$$d\left(\vec{x}_n - \vec{\mu}_j\right) = \left\|\vec{x}_n - \vec{\mu}_j\right\|_2$$

      la función de distancia $d\left(\vec{x}_n - \mu_k\right)$ mide la disimilitud entre las muestras $\vec{x}_n \in \mathbb{R}^D$ y $\vec{\mu}_k \in \mathbb{R}^D$ de dimensión D, de modo que si $d\left(\vec{x}_n - \vec{\mu}_k\right) \to \infty$, las muestras $\mathbf{x}_n$ y μ_k son más disimiles. Ejemplos de distancias: distancia de Bhattacharyya, Kolgomorov, etc.

- **Flujo del algoritmo K-medias:** Basados en las definiciones anteriores, el algoritmo K-medias kMedias(X, K, P), ejecutado para K clusters y en P iteraciones, consiste en lo siguiente:

  1. Inicializar aleatoriamente las K medias $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_K$.
  2. Por cada iteración $p = 1, \ldots, P$, ejecutar:
     a) **Fase de etiquetamiento de las muestras o *expectativa:*** Calcular para todas las muestras en X la matriz de pesos W la pertenencia a cada cluster k, según las medias $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_K$, como se especificó anteriormente:
     $$W_{n,k} = \begin{cases} 1 & k = \text{arg}\min_j d\left(\vec{x}_n - \vec{\mu}_j\right) \\ 0 & \text{de lo contrario} \end{cases}.$$
     b) **Fase de minimización del error o maximización de la verosimilitud:** En esta etapa se re-calculan las K medias $\vec{\mu}_1, \ldots, \vec{\mu}_k$:
     $$\vec{\mu}_k = \frac{\sum_{n=1}^{N} W_{n,k}\vec{x}_n}{\sum_{n=1}^{N} W_{n,k}},$$
  3. La matriz de pesos resultante W es utilizada para realizar el etiquetamiento por cada clase en el conjunto de muestras X.

Parte III
Requerimientos del programa

El programa debe ser desarrollado en Python, usando el **paradigma orientado a objetos (POO)**, y además debe cumplir los siguientes requerimientos funcionales:

1. Implementar una interfaz gráfica (5 % extra) que permita visualizar la cantidad de datos N (dada por el usuario) generados aleatoriamente, para $K = 2$, dos clases, con un símbolo distinto por clase, según el arreglo de etiquetas previamente conocido T.
   a) El usuario además debe especificar los centroides de ambas clases $\vec{\mu}_1$ y $\vec{\mu}_2$ y las desviaciones estándar (de los datos generados).
   b) Como parámetros del algoritmo K medias ($K = 2$ para clasificacion binaria), el algoritmo debe recibir la cantidad de iteraciones P.
   c) Debe implementarse un botón que permita ejecutar el algoritmo K-medias, y visualizar el etiquetamiento de las muestras según el algoritmo, en una nueva ventana.
2. Presentar el resultado de la clasificación automática usando el algoritmo de K-medias, en la interfaz gráfica del programa, coloreando los puntos segun la membresia estimada por su algoritmo.
3. Opcionalmente puede implementar la función que verifique cuál es la tasa de error respecto al arreglo de etiquetas T (20 % extra).
4. Opcionalmente puede implementar la visualización paso por paso de como cambian las medias en cada iteración (20 % extra).

1. Implementación

Para construír este y los demas proyectos, es necesario utilizar el siguiente conjunto de herramientas:

1. Star UML, para la diagramación de los casos de uso, y el diagrama de clases.

2. El paquete *Anaconda* (Python 3.5) el cual incluye la librería *scipy* necesaria para la manipulación de imágenes (se usará el IDE Spyder).

3. Lyx junto con LaTeX para la elaborar la documentación externa.

2. Evaluación

El siguiente corresponde al esquema de evaluación:

1. Implementación (70 %)

   a) Correcto funcionamiento en las pruebas funcionales (50 %)

   b) Uso correcto de las prácticas recomendadas de programación en el curso: documentación interna, variables y métodos signiticativos, etc. (10 %)

   c) Uso apropiado del paradigma OO (10 %).

2. Documentación externa (30 %)

   a) Seguimiento de la estructura propuesta, usando diagramas UML de disenio, y casos de uso (15 %).

   b) Correcta redacción, uso de figuras y citas bibliográficas (15 %)

3. Pruebas

3.1. Prueba 1

Se generaran los datos con los siguientes valores: generar_datos(numberSamplesPerClass = 200, mean1 = [2, 3], mean2 = [27, 27], stds1 = [3, 3], stds2 = [4, 3]). Se espera que la tasa de aciertos sea casi siempre de 100 %. Se ejecutara por 50 iteraciones.

3.2. Prueba 2

Se generaran los datos con los siguientes valores: generar_datos(numberSamplesPerClass = 200, mean1 = [2, 3], mean2 = [10, 10], stds1 = [3, 3], stds2 = [4, 3]). Se ejecutara por 50 iteraciones.

3.3. Prueba 3

Se generaran los datos con los siguientes valores: generador_datos.generar_datos(numberSamplesPerClass = 200, mean1 = [2, 3], mean2 = [6, 6], stds1 = [3, 3], stds2 = [4, 3]). Se ejecutara por 50 iteraciones.