Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0117 Programación Bajo Plataformas Abiertas

MSc. Andrés Mora Zúñiga - I Ciclo 2020

Examen Final

K-Means

Instrucciones Generales:
El examen se debe realiar de manera individual.
Entregue un archivo comprimido que incluya un directorio llamado `src` con los archivos `.h`, y `.c` que lleven a la solución.
El examen debe entregarse a más tardar el 5 de Julio a las 23:59.
El examen es a libro abierto. Puede utilizar internet para guiarse si lo considera necesario.

1. Nota teórica

El reconocimiento de patrones es la ciencia que se encarga de la identificación automatizada de características (patrones) en datos a partir de algoritmos computacionales y con ello tomar acción, como por ejemplo clasificar las observaciones en diferentes clases.

Tiene sus raíces en ingeniería, estadística, procesamiento de señales e inteligencia artifical. Es un proceso completo que abarca:

1. La captura de información utilizando sensores (cámaras, micrófonos, microscopios, de proximidad, etc).
2. El procesamiento de los datos para remover ruido, impurezas, y hasta la imputación de datos cuando la observación es incompleta.
3. La extracción de características de los datos para poder identificar cuáles son los principales diferenciadores en los datos y así poder hallar los patrones que fácilmente los distinguen unos de otros.
4. La clasificación o agrupación en los diferentes grupos de interés.
5. La toma de decisiones luego de conocer la clase a la que pertenece una observación determinada.

En este examen nos vamos a concentrar particularmente en el **punto 4**.

La teoría clásica del reconocimiento de patrones agrupa los clasificadores en dos grupos según el método de aprendizaje:

1. **Aprendizaje supervisado:** el clasificador, durante el entrenamiento, cuenta con información a priori del tipo de dato que va a clasificar (etiqueta de la clase), por lo que puede validarse directamente si la clasificación fue correcta o no.
2. **Aprendizaje no-supervisado:** el clasificador, durante el entrenamiento, NO cuenta con la etiqueta de a qué clase pertenece el individuo en cuestión. Por lo que debe de realizar la clasificación (que en este tipo de algoritmos no supervisados comúnmente es llamada agrupación) contando únicamente con la información del conjunto de datos y no puede validar su resultado directamente al revisar contra una etiqueta.

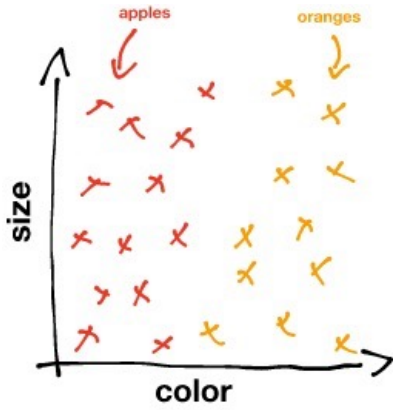
En este examen nos vamos a concentrar específicamente en el **algoritmo K-Means de aprendizaje no supervisado**.

1.1. K-Means

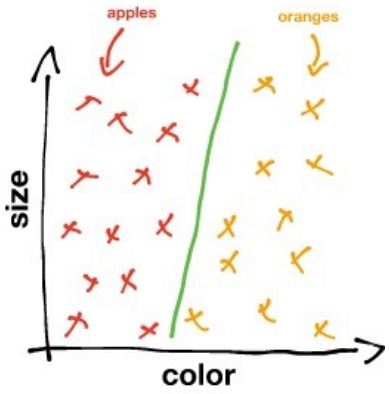
K-Means es uno de los algoritmos más simples de agrupamiento no supervisado el cual permite identificar grupos basándose en el promedio de sus características.

Tome por ejemplo el caso de clasificar frutas, especifífcamente naranjas y manzanas. Tenemos que el principal diferenciador es el color, pero también el tamaño podría ayudar a diferenciar.

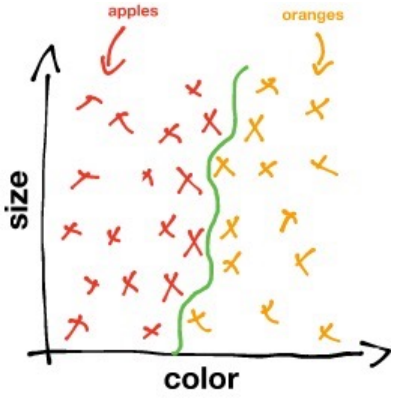
Podríamos representar el problema como lo muestra la siguiente figura:



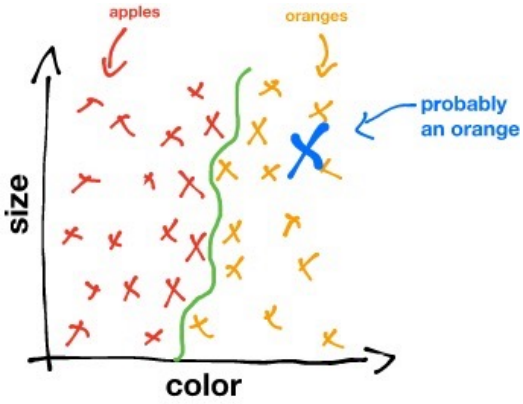
(a) Distribución de manzanas y naranjas



(b) Frontera simple que separa las clases si todas las manzans son rojas y todas las naranjas naranja



(c) Frontera un poco más compleja si las las frutas toman tonos más amarillentos y se parecen más entre ellas.



(d) Si se tiene esa observación, posiblemente lo que se está observando, sea una naranja

Figura 1: Ejemplo de manzanas y naranjas. Imágenes tomadas de `https://zcfy.cc/original/machine-learning-crash-course-part-1-middot-ml-b`

Cómo se puede ver en la Figura 1 las información como el tamaño y el color podría representarse de manera numérica en un plano cartesiano. De esta forma una observación nueva que se tenga, se podría clasificar como manzana o naranja dependiendo de en que lado de la frontera se encuentre, o bien, de una manera aún más sencilla, dependiendo de que tan cerca este de las manzanas o de las naranjas esté, utilizando una simple distancia euclidiana.

El método K-Means toma la idea de cercanía con las observaciones iniciales y lo generaliza un poco más. El K-Means toma los datos de cada clase (cluster), los promedia (calcula el centroide) y luego para decidir a cuál clase (cluster) pertenece una observación, lo decide con la distancia euclidiana más corta a uno de los centroides.

Pensemos ahora en un caso meramente numérico en lugar de las frutas, y ahora son tres clusters en lugar de dos. El problema podría representarme gráficamente de la siguiente manera, incluyendo los centroides en la representación:

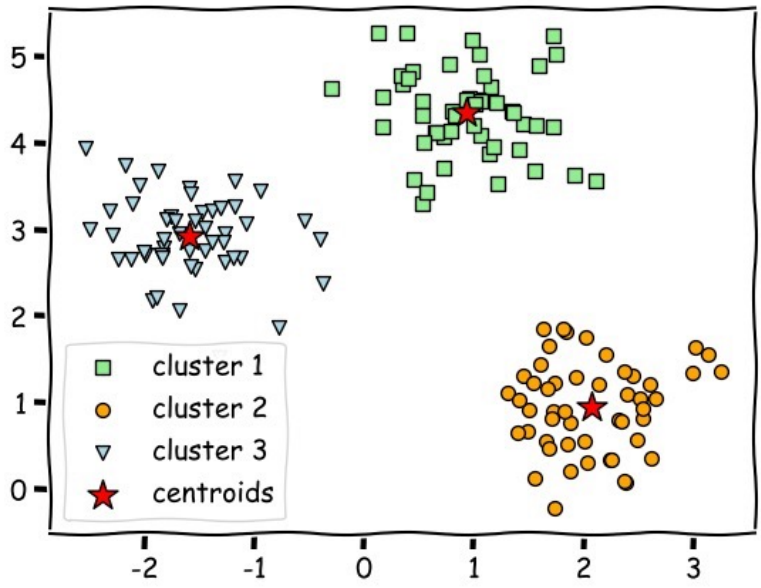


Figura 2: Representación gráfica de tres clusters, con sus centroides. Tomada de `https://heartbeat.fritz.ai/understanding-the-mathematics-behind-k-means-clustering-40e1d55e2f4c`

Para obtener el centroide de un cluster es tan fácil como promediar los valores en X, y en Y de forma independiente para cada uno de los elementos pertenecientes al cluster, y con el resultado se construye una nueva coordenda, es decir:

$$C = (\bar{x}, \bar{y}) \tag{1}$$

Podemos saber a cuál de los tres clusters pertenece una observación calculando la distancia euclidiana entre la observación y cada uno de los tres centroides. De esta forma, la observación pertenece al cluster con el cual la distancia euclidiana a su centroide sea la más corta.

Recordando que la distancia euclidiana entre dos puntos puede calcularse como:

$$d(A, B) = \sqrt{(A.x - B.x)^2 + (A.y - B.y)^2} \tag{2}$$

Donde:

- A es un punto con coordenadas (x, y)
- B es un punto con coordenadas (x, y)

1.2. Algoritmo K-Means

Entrada: Número entero K con la cantidad de clusters que se desean y el conjunto de datos.
Salida: Los K clusters que se formaron.
Pasos:

1. Tomar del conjunto de observaciones K puntos aleatorios y utilizarlos arbitrariamente como los centroides.
2. Asignar a los centroides las etiquetas $1, 2, ..., K$.
3. Para cada observación:
   a) Calcular la distancia euclidiana entre la observación o_i y cada uno de los k centroides.
   b) Tomar la etiqueta del centroide con el cual tenga menor distancia y asignársela a la observación.
4. Recalcular los centroides promediando todas las observaciones que pertenezcan actualmente a dicho cluster.
5. Repetir los pasos 3 y 4 un número determinado de veces, por ejemplo, 100.

Al finalizar dichos pasos, se cuenta con una etiqueta asignada a cada observación. Esa etiqueta es al cluster al que pertenece.

2. Desarrollo

El estudiante deberá de implementar en el lenguaje C una programa capaz de realizar una agrupación no supervisada utilizando el algoritmo K-Means para datos en 2 dimensiones.

2.1. Código fuente

La implementación que debe realizar el estudiante consta de los siguiente archivos de código fuente:

1. `kmeans.h`: archivo de encabezado con la declaración de tipos de datos personalizados y las firmas de todas la funciones que se deben de implementar en el examen.
2. `kmeans.c`: código fuente con la implementación de las funciones descritas en la siguiente sección.

2.2. Requerimientos

En esta sección se enumeran los distintos tipos de datos por definir y las funciones MÍNIMAS a implementar para el programa de K-Means. El estudiante debe de ordenar su código según lo estipulado en la sección anterior y debe mantener el orden para que su código sea legible y fácil de entender. **Nota:** se recomienda hacer más funciones para algunos pasos intermedios del programa.

- Tipo de dato `p2D`: este tipo se va a utilizar para representar la parte númerica de las observaciones dos dimensiones. Almacena 2 números de punto flotante.
- Tipo de dato `obs`: este tipo se utiliza para representar la observación como un todo, es decir su parte númerica y la etiqueta del cluster al que pertenece. Entonces almacena un p2D y un int.
- Tipo de dato `centroide`: este tipo se utiliza para representar el centroide del cluster es decir su parte númerica y la etiqueta del cluster. Entonces almacena un p2D y un int.
- Función `importData`: esta función recibe la ruta del archivo desde donde se van a importar las observaciones, y regresa un puntero que apunta a un bloque de memoria en el heap con todas las observaciones importadas. Además recibe también como parámetro un puntero a un entero en el regresa la cantidad de observaciones importadas del archivo. Inicialmente el valor de la etiqueta de la observación es 0. **Nota:** El archivo a importar es `data.txt` que va como adjunto a este documento.

- Función `getInitialCentroids`: esta función recibe la cantidad K de clusters que se desea fomar, y el puntero de las observaciones. Regresa un puntero que apunta a K centroides en el heap, formados a partir de K observaciones aleatorias tomadas del conjunto de datos. Estos centroides son el punto de partida del algoritmo.
- Función `kMeans`: esta función recibe por parámetro el puntero de los centroides, el puntero de las observaciones, la cantidad de observaciones, la cantidad de clusters y procede a realizar la asignación de las etiquetas a las observaciones según el Algoritmo K-Means con 100 iteraciones. No regresa nada, pues todo lo modifica a través de los punteros que recibe.
- Función `exportClusters`: esta función recibe el puntero de los centroides, el puntero de las observaciones, la cantidad de observaciones, la cantidad de clusters, y la ruta donde se desea exportar el archivo. Con estos parámetros procede a escribir un archivo con el siguiente formato: K Centroides, línea en blanco, observaciones con su respectiva etiqueta. Por ejemplo:

```
x_c1 y_c1 e_c1
x_c2 y_c2 e_c2
...

x_1 y_1 e_1
x_2 y_2 e_2
x_3 y_3 e_3
...
```

  Donde:

  - x_{ci} es la coordenada x del centroide i, y_{ci} es la coordenada y del centroide i y e_ci es la etiqueta del centroide i.
  - x_j es la coordenada x de la observación j, y_j es la coordenada y de la observación j y e_j es la etiqueta de la observación j.

3. Programa Principal

En el programa principal (`main.c`) debe de importar el archivo `kmeans.h` y con sus funciones realizar los siguientes pasos:

- Recibir como argumento de línea de comandos la cantidad K de clusters deseada.
- Importar las observaciones del archivo `datos.txt`.
- Hacer la agrupación de las observaciones con el algoritmo K-Means.
- Exportar el resultado al archivo `clusters.txt`.