Clasificación no supervisada

M. Sc. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Computación, bachillerato en Ingeniería en Computación,
PAttern Recongition and MAchine Learning Group (PARMA-Group)

30 de julio de 2019

A continuación se presentan distintos enfoques de clasificación no supervisada.

1. Introducción al aprendizaje no supervisado

El aprendizaje no supervisado consiste en particionar un conjunto de N datos de entrada $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$ (donde cada muestra es de dimensión D, $\mathbf{x} = \{x_1, x_2, \ldots, x_D\}$) en G grupos, máximamente homogeneos (montones o clústers). Tales G grupos idealmente deben corresponder a las K clases reales. La Figura 1 muestra el resultado final típico de un algoritmo de amontonamiento, e ilustra el proceso de asignación de etiquetas automático, prescindiendo del conjunto de etiquetas correctas $\mathbf{t}$, necesario en el problema de clasificación supervisada.

En la literatura es posible encontrar diferentes esquemas para agrupar los distintos algoritmos de amontonamiento existentes. Un enfoque para clasificar

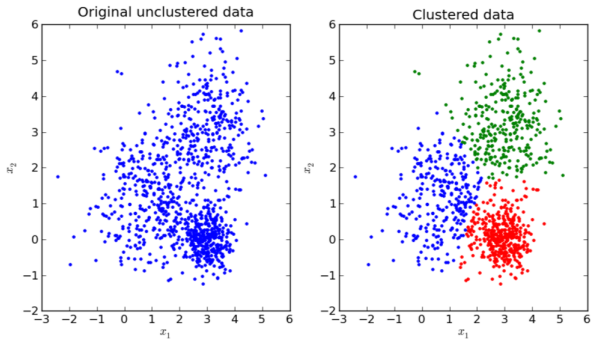

Figura 1: Resultado de un algoritmo de amontonamiento típico.

los diferentes algoritmos de amontonamiento se sugiere en [**?**], y se resume como sigue:

1. **Secuenciales: (***Basic Sequential Algorithm Scheme***)** : Los vectores son presentados una o pocas veces al algoritmo, con un resultado dependiente del orden de presentación. La cantidad de clases K es desconocida.

2. **Jerárquicos:**

   a) Algoritmos aglomerativos: Se inicia con el número máximo de clusters y se van "aglomerando", produciendo un número descendiente de clusters.

   b) Algoritmos divisivos: Lo opuesto.

3. **Basados en Grafos:** (clustering espectral y corte de grafos): Interpretación de datos como nodos de un grafo conectados por sus distancias.

4. **Basados en distribución:** (Maximización de la esperanza): Pertenencia probabilistica o suave, haciendo interpretación de distribución de los datos.

5. **Basados en la minimización de error:** (K-medias): Minimiza una función de costo, el número de clases K es conocido.

6. **Basados en redes neuronales:** redes neuronales de tipo ART, algoritmos de aprendizaje competitivo, etc.

2. El algoritmo de amontonamiento K-medias

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.

- **Objetivo general:** Partir el conjunto de datos de entrada con N muestras:

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

  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:

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

donde la función de distancia $d\left(\vec{x}_n - \vec{\mu}_k\right)$ 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 $\vec{x}_n$ y μ_k son más disimiles. Ejemplos de distancias: distancia de Bhattacharyya, Kolgomorov, etc.

- **Función de costo:** Sumatoria de las distancias de todas las muestras $\vec{x}_n$ que pertenecen a la clase k, respecto a la muestra media $\vec{\mu}_k$, para todas las K clases:

$$J(X) = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K} W_{n,k} d\left(\vec{x}_n - \vec{\mu}_k\right)}{\sum_{n=1}^{N}\sum_{k=1}^{K} W_{n,k}},$$

donde K corresponde a la cantidad de clusters, N a la cantidad de muestras. La distancia total de todas las muestras de un cluster K a su media $\vec{\mu}_k$

- **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:

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

   b) **Fase de minimización del error o maximización de la verosimilitud:** Como se analizó en laboratorios anteriores: la maximización de la verosimilitud equivale a la minimización de la función de error, en este caso $J(X, \mu)$, tiene su mínimo error cuando:

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

   Por lo que entonces en esta etapa se re-calculan las K medias $\vec{\mu}_1, \ldots, \vec{\mu}_k$.
3. La matriz de pesos resultante W es utilizada para realizar el etiquetamiento por cada clase en el conjunto de muestras X.

Alternativamente el algoritmo de K-medias se puede detener utilizando algún criterio de convergencia, como por ejemplo el cambio en las medias μ.

La Figura 2 muestra el funcionamiento del algoritmo K-medias.

La función de costo

$$J(X) = \frac{\sum_{n=1}^{N} \sum_{k=1}^{K} W_{n,k} d\left(\vec{x}_n - \vec{\mu}_k\right)}{\sum_{n=1}^{N} \sum_{k=1}^{K} W_{n,k}},$$

mide la «calidad» de los clústers, al analizar su dinámica o comportamiento para distintos números de K clusters. En un caso donde la cantidad K es baja, los clusters se subsegmentan, porlo que la sumatoria total de distancias de todos los elementos al centroide del cluster es alta. Conforme aumenta K, $J(X)$ disminuye a un alto ritmo, hasta que sucede lo que se conoce como «sobresegmentación» de los clusters, es decir, cuando K es mayor a la cantidad de clústers «reales», el valor de $J(X)$ es bajo y se estabiliza. Lo anterior se ilustra en la Figura 3, y se le refiere también como el «método del codo».

3. El algoritmo de corte de grafos

El algoritmo de corte de Grafos es otro enfoque para realizar el *clustering* de los datos. A continuación se presenta el detalle del algoritmo de corte de grafos para el amontonamiento o clustering en $K = 2$ clases. Tal algoritmo se basa en el concepto de grafo, el cual se puede definir formalmente como sigue a continuación.

(Loading...)

Figura 2: Funcionamiento del algoritmo K-medias. Tomado de [**?**].

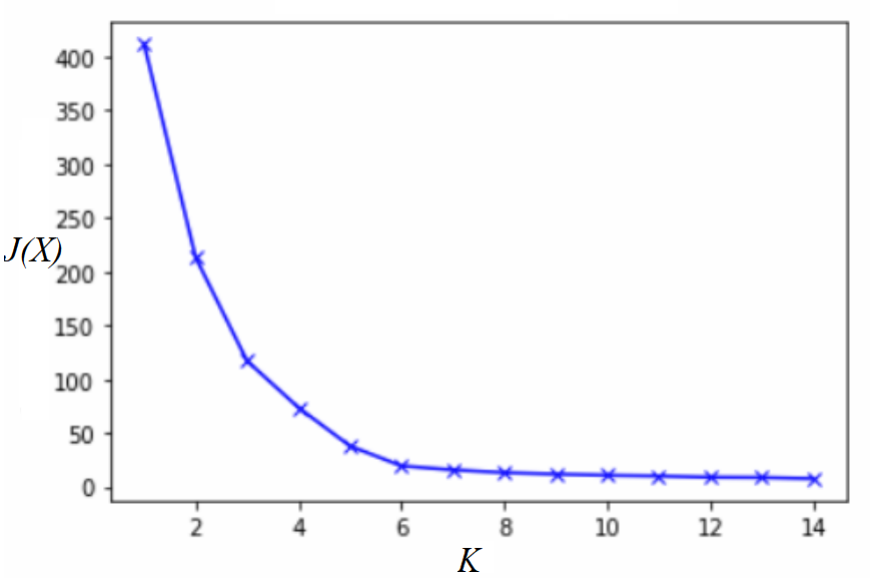

Figura 3: Método del codo, con el cual se puede escoger, según la gráfica, $K = 5$.

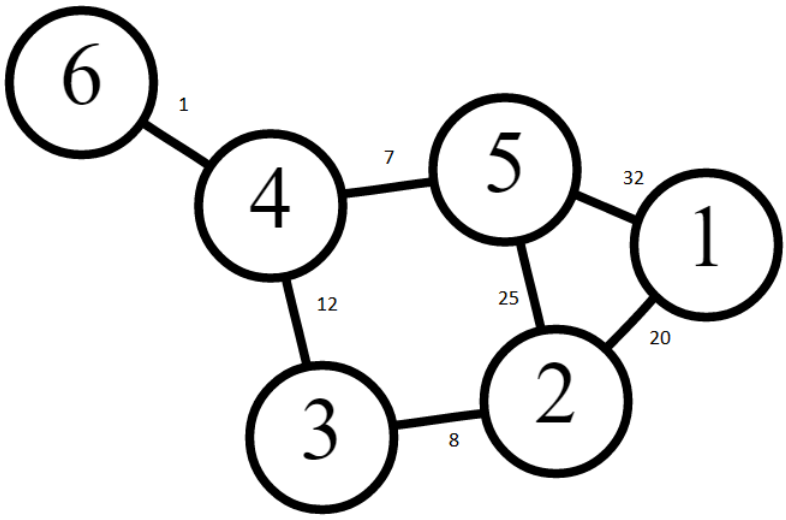

Figura 4: Ejemplo de grafo no dirigido.

3.1. Grafos

Un grafo G se define formalmente como un conjunto de n vértices o nodos $V = \{v_1, v_2, \ldots, v_n\}$, los cuales están conectados por los aristas definidos en el conjunto de aristas $E = \{e_1, e_2, \ldots, e_n\}$, por lo que entonces en forma matemática, el grafo viene dado por:

$$G = \{E, V\}$$

Cada nodo v_i contiene una *etiqueta* k la cual debe identificar unívocamente a tal nodo. Los nodos están conectados entre sí por los aristas, con lo cual un arista e_k se define con la tripleta:

$$e_k = \langle v_i, v_j, w_{i,j} \rangle$$

donde entonces el arista k conecta a los nodos v_i y v_j, con costo o peso $w_k \in \mathbb{R}$, el cual es un escalar que denota el costo de ir desde el nodo origen v_i al nodo de destino v_j. Si para todo arista e_k es posible ir desde el vértice de origen al vértice de destino, y también del vértice de destino al de origen, se dice que el mismo es un grafo no dirigido. De otro modo el grafo es un grafo no dirigido. En la Figura 4 se presenta un grafo no dirigido, el cual implementaremos en la presente Tarea Programada.

Para tal grafo de la Figura 4 se define el conjunto de nodos como:

$$V = \{v_1 = 1, v_2 = 2, v_3 = 3, v_4 = 4, v_5 = 5, v_6 = 6\}$$

y el conjunto de vértices vendría dador por:

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$$

Muestra	Húmedad (%)	Temperatura (C°)	Clima
1	50	32	Verano
2	42	29	Verano
3	80	15	Invierno
4	70	19	Invierno
5	75	13	?

Cuadro 1: Muestras recolectadas en distintos días.

donde:

$$
\begin{aligned}
e_1 &= \langle v_6, v_4, w_{6,4} = 1 \rangle \\
e_2 &= \langle v_4, v_5, w_{4,5} = 7 \rangle \\
e_3 &= \langle v_4, v_3, w_{4,3} = 12 \rangle \\
e_4 &= \langle v_2, v_1, w_{2,1} = 20 \rangle \\
e_5 &= \langle v_3, v_2, w_{3,2} = 8 \rangle \\
e_6 &= \langle v_5, v_1, w_{5,1} = 32 \rangle \\
e_7 &= \langle v_5, v_2, w_{5,2} = 25 \rangle
\end{aligned}
$$

Tome en cuenta que el grafo es un grafo no dirigido, por lo que por ejemplo para el arista e_1 se puede pasar del nodo v_6 al v_4 y viceversa. Los grafos pueden implementarse en un lenguaje de programación en múltiples formas, una muy utilizada es la matriz para representar el peso o costo entre los nodos v_i y v_j, en la fila i y columna j. A tal matriz se le conoce como matriz de adyacencia. Por ejemplo, para el grafo anterior, la matriz de adyacencia $M \in \mathbb{R}^{n \times n}$ viene dada por:

$$
M = \begin{bmatrix}
0 & 2 & \infty & \infty & 32 & \infty \\
2 & 0 & 8 & \infty & 25 & \infty \\
0 & 8 & 0 & 12 & \infty & \infty \\
\infty & \infty & 12 & 0 & 7 & 1 \\
32 & 25 & \infty & 7 & 0 & \infty \\
\infty & \infty & \infty & 1 & \infty & 0
\end{bmatrix}
$$

Observe que para los grafos no dirigidos, se tiene que $M_{i,j} = M_{j,i}$, por lo que la matriz es simétrica.

3.2. Grafos a partir de conjuntos de datos

Un conjunto de datos en $\mathbb{R}^2$, el algoritmo de amontonamiento representa las distancias euclidianas entre dos puntos o muestras x_i e x_j como los costos o pesos de ir de un nodo al otro. Es por ello que a partir de un conjunto de datos X podemos contruir un grafo, en el cual los nodos vienen dados por las muestras, y los aristas tienen asociada la distancia euclidiana entre muestras. A partir del ejemplo de la estimación del clima en un día (verano o invierno), tómese el conjunto de datos definido, y además, una nueva muestra para cuya clase (invierno o verano) es desconocida.

La Figura 5 muestra una graficación de tales datos.

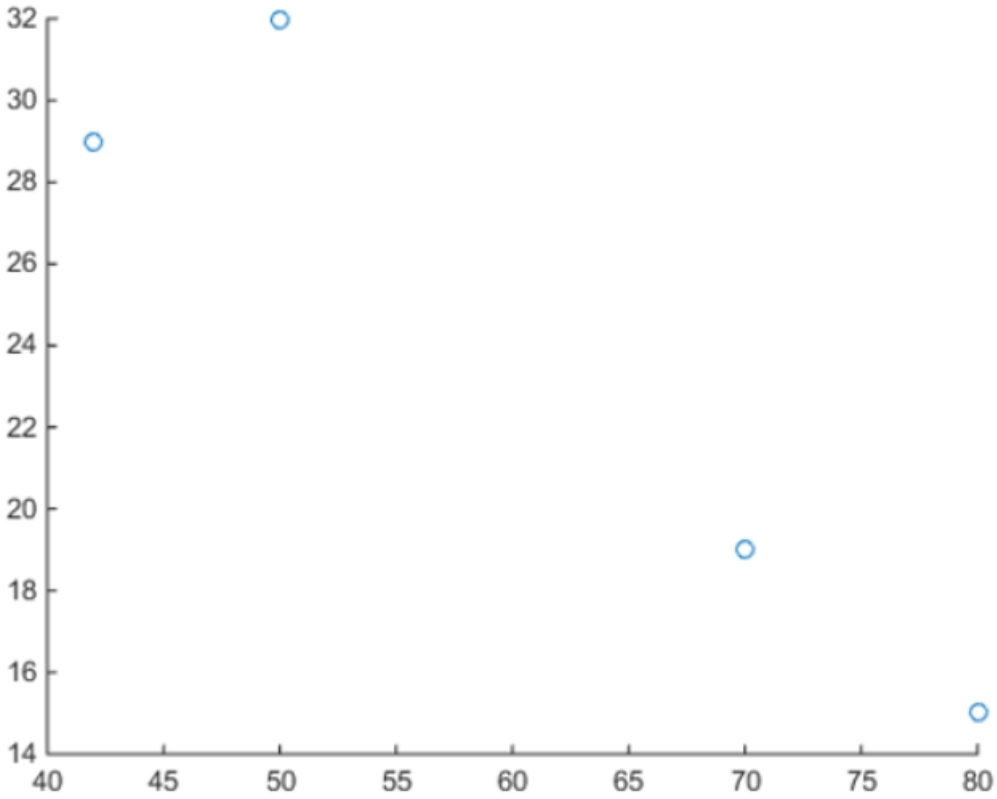

Figura 5: Graficación de los datos.

Dist. euclidianas	1	2	3	4	5
1	0	8.544	34.48	23.85	31.4
2	8.544	0	40.49	29.73	36.67
3	34.48	40.49	0	10.77	5.38
4	23.85	29.73	10.77	0	7.81
5	31.4	36.67	5.38	7.81	0

Cuadro 2: Muestras recolectadas en distintos días.

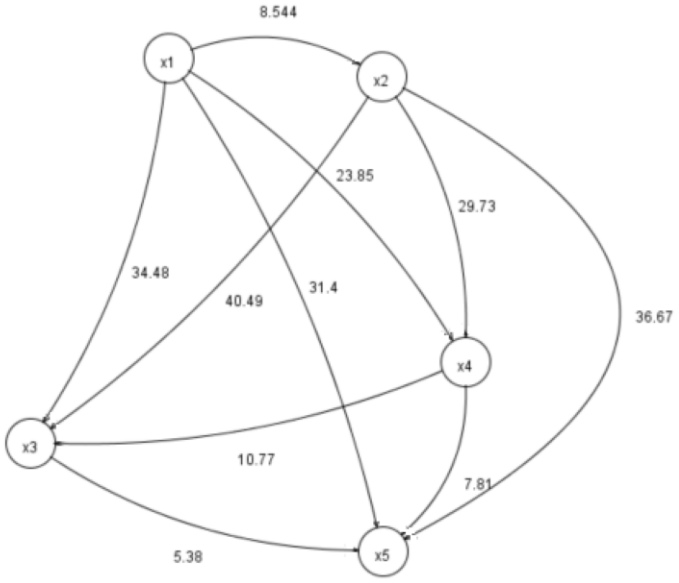

Figura 6: Grafo construído a partir de las distancias entre las muestras.

El algoritmo de corte de grafos construye un grafo definiendo un nodo v_i por cada dato $\vec{x}_i$, y definiendo el peso de los aristas que conectan a todos los otros nodos como la distancia Euclidiana entre cada dato. Es por ello que procedemos a calcular la distancia Euclidiana entre cada par de datos, lo que implicitamente define la matriz de adyacencia del grafo, en la Tabla 2.

La Figura 6 muestra gráficamente el grafo construído a partir de la matriz de adyacencia de la Tabla 2.

El algoritmo de corte de grafos busca el arista donde realizar el corte, y para ello, construye lo que se denomina un árbol de expansión mínima. La construcción de tal árbol se puede realizar con el algoritmo de Prim, el cual se detalla en la siguiente sección.

3.3. Construcción del árbol de expansión mínima: el algoritmo de Prim

Un árbol es básicamente un grafo en el cual se definen los conceptos de nodo padre y nodo hijo. Los nodos hijos no pueden conectarse con sus antecesores. Uno de los árboles más sencillos es el árbol binario, el cual restringe a cada nodo a tener un máximo de 2 hijos. Un árbol con n hijos máximo se denomina un árbol *n-ario*. A continuación se detallan los conceptos básicos de un árbol con un árbol.

El árbol de expansión mínima es aquel cuyo costo total (sumatoria de todos los pesos entre los nodos), es el menor. El algoritmo de Prim define dos conjuntos de nodos:

- El conjunto de nodos A incluidos en el árbol de expansión mínima, inicializado como vacío:
$$A = \{\}$$
- El conjunto de nodos B no incluidos en el árbol de expansión mínima, el cual se inicializa con todos los nodos del grafo:
$$B = \{v_1, v_2, \ldots, v_n\}$$

El **algoritmo de Prim** para construír el árbol de expansión mínima consiste básicamente en lo siguiente:

1. Eliminar un nodo aleatorio del conjunto B y agregarlo al conjunto A.
2. Buscar el arista e_k que conecte a un nodo $v_i \in A$ que pertenezca al conjunto A a un nodo $v_j \in B$, de forma que el peso $w_{i,j}$ sea el menor posible.
    a) Se conserva tal arista e_k entre tal par de nodos
3. Incluir el nodo v_j al conjunto A y excluir tal nodo del conjunto B.
4. Repetir los dos pasos anteriores hasta que el conjunto B esté vacío.

A continuación se ilustra tal algoritmo para el caso de ejemplo tratado hasta ahora, por iteración. Se inicializan primero los conjuntos de la siguiente forma:

$$A = \{\}$$

$$B = \{v_1 = x_1, \ldots, v_i = x_i, \ldots, v_n = x_n\}$$

- Iteración 1: Se toma aleatoriamente el nodo v_1 y se agrega a A, de modo que:
$$A = \{v_1\}$$
$$B = \{v_2, v_3, v_4, v_5\}$$

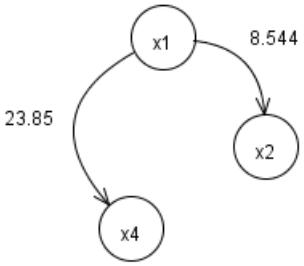

Figura 8: Iteracion 3.

- Iteración 2: Tomar el nodo más cercano del conjunto B al nodo v_1, el cual en este caso es el nodo v_2. La Figura 7 ilustra como se va construyendo el árbol de expansión mínima.

$$A = \{v_1, v_2\}$$

$$B = \{v_3, v_4, v_5\}$$

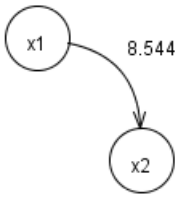

Figura 7: Iteracion 2.

- Iteración 3: Tomar el camino más cercano (arista con menor costo) entre algún nodo del conjunto B y otro del conjunto A, el cual en este caso es el arista dado por $\langle v_1, v_4, w_{1,4} = 23{,}85 \rangle$. La Figura 8A ilustra como se va construyendo el árbol de expansión mínima.

$$A = \{v_1, v_2, v_4\}$$

$$B = \{v_3, v_5\}$$

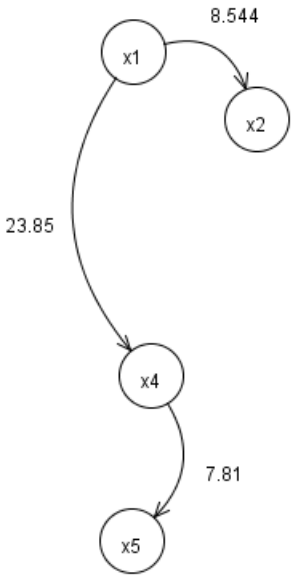

Figura 9: Iteracion 4.

- Iteración 4: Tomar el camino más cercano (arista con menor costo) entre algún nodo del conjunto B y otro del conjunto A, el cual en este caso es el arista dado por $\langle v_4, v_5, w_{4,5} = 7{,}81 \rangle$. La Figura 9 ilustra como se va construyendo el árbol de expansión mínima.

$$A = \{v_1, v_2, v_4, v_5\}$$

$$B = \{v_3, v_5\}$$

- Iteración 5: Tomar el camino más cercano (arista con menor costo) entre algún nodo del conjunto B y otro del conjunto A, el cual en este caso es el arista dado por $\langle v_3, v_5, w_{3,5} = 5{,}38 \rangle$. La Figura 10 ilustra como se va construyendo el árbol de expansión mínima.

$$A = \{v_1, v_2, v_3, v_5, v_4\}$$

$$B = \{\}$$

Para crear los dos montones o clusters, se puede cortar el arbol de expansión mínima en el arista con mayor magnitud, en este caso 23.85.

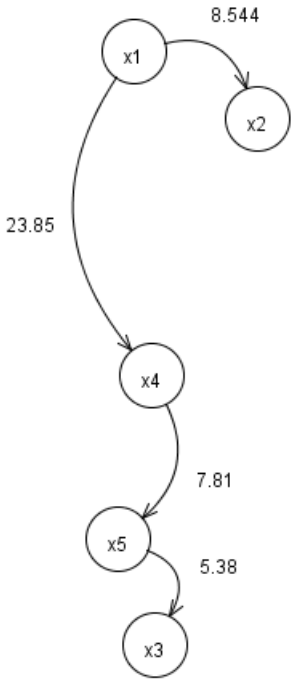

Figura 10: Iteracion 5.