



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

Clustering

25 de octubre de 2018

Algoritmos y Estructura de Datos III

Integrante	LU	Correo electrónico
Buceta, Diego	001/17	diegobuceta35@gmail.com
Springhart, Gonzalo	308/17	glspringhart@gmail.com

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

1. Introducción al problema

En este informe vamos a ver distintos problemas cuyas posibles soluciones involucran árboles generadores mínimos y algoritmos que los generan. En particular los problemas son el **Clustering** y el **Arbitraje**.

2. Clustering

En el mundo del machine learning, problema de **Clustering** consiste en poder **agrupar datos** en distintos grupos llamados clusters, donde se espera que los datos de un cluster tengan algún tipo de **relación o característica** que los **distinga** de los datos en otros clusters. Este problema es de gran interés ya que tiene muchos usos como por ejemplo:

- Las tomografías por emisión de positrones utilizan análisis de clusters para poder diferenciar tejidos en imágenes tridimensionales.
- La investigación de mercados utilizan el clustering para poder particionar la población de consumidores en grupos para poder entender mejor la relación entre ellos.
- El clustering puede ser utilizado por motores de búsqueda de páginas web para poder agrupar resultados similares.
- La segmentación de imágenes utiliza clustering para poder detectar bordes u objetos en imágenes.

El análisis de Clusters es un problema de **aprendizaje no supervisado**, que puede ser abarcado de **varias maneras**, dependiendo de como uno **interprete que es lo que constituye un cluster** y como se puede encontrar de forma eficiente. Los diversos algoritmos que resuelven el problema se pueden categorizar según su interpretación de lo que es cluster. Se puede reformular el problema de Clustering como un problema de **optimización multiobjetivo**, donde el algoritmo y los parámetros (por ejemplo la función para calcular distancias, o la cantidad esperada de clusters) relacionados a este dependen del conjunto de datos a analizar. El análisis de clusters es un **proceso iterativo** que involucra prueba y error, generalmente es necesario modificar parámetros hasta conseguir los resultados satisfactorios.

2.1. Dificultades del clustering, Aprendizaje Supervisado vs No Supervisado

Como se mencionó anteriormente el análisis de clusters es un problema de **aprendizaje no supervisado**, para poder entender como esto afecta al problema vamos a explicar de forma resumida los dos tipos principales de aprendizajes que existen en el mundo del machine learning.

Aprendizaje Supervisado

El Aprendizaje Supervisado es una técnica de machine learning que, a partir de conjuntos de entrenamiento de entradas y salidas esperadas, utiliza un algoritmo que intenta inferir una función que pueda predecir las salidas adecuadas para cualquier entrada nueva. Se llama supervisado ya que a medida que el algoritmo hace predicciones sobre los conjuntos de entrenamiento se van corrigiendo los datos de acuerdo a las salidas esperadas. Los problemas de aprendizaje supervisado se pueden agrupar en problemas de **clasificación**, cuando las salidas son clases o categorías y problemas de **regresión**, cuando las salidas son valores reales.

Aprendizaje No Supervisado

A diferencia del aprendizaje supervisado, en el **aprendizaje no supervisado** no se sabe nada de la información de entrada **a priori**. A partir de un conjunto de datos de entrada se corren algoritmos que se encargan de descubrir estructuras o características en los datos. Como no hay salidas esperadas, la única forma de saber si los datos resultantes de correr los algoritmos son

correctos es realizar evaluaciones sobre los resultados. Pese a esta desventaja, el aprendizaje no supervisado es útil cuando por ejemplo, no se sabe nada sobre los datos a estudiar, por lo que se puede usar un algoritmo de este tipo para obtener información sobre los mismos.

Entonces podemos ver que el problema de análisis de clusters cae en la segunda categoría de aprendizaje, esto lo hace un problema difícil, ya que no existe una solución correcta con la que podríamos revisar los resultados obtenidos de los algoritmos de clustering, además por lo mencionado anteriormente, la definición de cluster en si es ambigua y dependiendo de lo que se decida definir como un cluster, los resultados obtenidos sobre un mismo conjunto de datos de entrada pueden ser muy diferentes.

2.2. Heurística para resolver el problema

Para resolver el problema utilizamos un algoritmo basado en el método de detección de clusters explicado en el paper "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters" de Charles T. Zahn, y su forma de calcular los clusters es la siguiente: Consideramos nuestros datos como una serie de puntos en un plano, tomando un grafo completo G usando los puntos como vértices y las distancias entre ellos como pesos en las aristas. Luego calculamos el Árbol Generador Mínimo de G y lo llamamos T . Para poder encontrar los clusters hay que eliminar de T a los ejes **inconsistentes**, según el paper un eje e es inconsistente si su peso supera en cierta cantidad de desviaciones estandar a las medias de los pesos de las aristas que estén a distancia k de los vértices que están en sus puntas. Al remover un eje inconsistente, las componentes conexas formadas son identificadas con un índice, entonces luego de recorrer todas las aristas, todas las componentes conexas van a tener un índice distinto y podemos interpretarlas como los clusters del grafo. Para poder calcular el AGM del grafo G se implementaron dos algoritmos, el algoritmo de Kruskal y el algoritmo de Prim, ambos levemente modificados para utilizar estructuras de Lista de Incidencia y Lista de Adyacencia respectivamente. Además, se implementaron dos versiones de Kruskal, una con path-compression y otra sin, a fin de poder compararlos en la experimentación.

2.3. Principios de la Forma (Gestalt Principles)

El método de detección de clusters esta basado en los *principios de la forma de organización perceptible* que están explicados en el paper anteriormente mencionado, en resumen se intento formular un método matemático que detecte clusters utilizando los principios de tal forma que los clusters formados correspondan con los que una persona podría percibir al ver el grafo. Los diversos principios explican como la percepción humana organiza datos sensoriales visuales, reconoce patrones y simplifica imágenes complejas. El principio fundamental para la detección de clusters es el **principio de la proximidad** que dice: "Los objetos o formas que se encuentren cerca unos de otros aparentan **formar grupos**", incluso si los objetos o formas son muy diferentes van a parecer formar parte de un grupo si se encuentran **cerca**. En este principio, el sentido de que dos objetos estén cerca no necesariamente es el de que su distancia sea corta, varios objetos pueden ser agrupados siguiendo este principio pero aplicado sobre otras características de los mismos, como sus tamaños, sus formas, sus colores, etc.

3. Justificación teórica

3.1. Arboles Generadores Mínimos

En el método explicado anteriormente los clusters se calculan sobre un AGM del grafo completo de los datos,

4. Algoritmos presentados

4.1. Kruskal

KRUSKALSINPATHCOMP(*ListaIncidencia* : grafoCompleto, *cantNodos* : entero)

```
1 padre ← vector de enteros de tamaño de cantNodos y cargado con el valor de su posición en cada posición
2 AGM ← lista de incidencia vacia, de tamaño cantNodos-1
3 OrdenarPorPeso(grafoCompleto)
4 for e:Arista ∈ grafoCompleto
5     do
6         if getPadre(indice(e.primerNodo),padre) == getPadre(indice(e.segundoNodo),padre)
7             then agregar(e,agm)
8 Devolver AGM
```

GETPADRE(*entero* : indice, *padre* : vectordeenteros)

```
1 if padre[indice] == indice
2     then
3         Devolver indice
4     else
5         getPadre(indice(padre[indice]),padre)
6
```

GETPADRECONPATHCOMP(*entero* : indice, *padre* : vectordeenteros, *altura* : vectordeenteros, *nivelesSubidos* : entero)

```
1 if padre[indice] == indice
2     then
3         altura[indice] ← nivelesSubidos
4         Devolver indice
5     else
6         padre[indice] ← getPadreConPathComp(indice(padre[indice]),padre,altura,nivelesSubidos+1)
7         Devolver padre[indice]
8
```

UNIRPADRES(*indiceNodo1* : entero, *indiceNodo2* : entero, *padre* : vectordeenteros)

```
1 padreNodo1 ← getPadre(indiceNodo1, padre)
2 padre[indiceNodo2] ← padreNodo1
```

UNIRPADRESCONPATHCOMP(*indiceNodo1* : entero, *indiceNodo2* : entero, *padre* : vectordeenteros, *altura* : vectordeenteros, *nivelesSubidos* : entero)

```
1 padreNodo1 ← getPadreConPathComp(indiceNodo1, padre,altura,0)
2 padreNodo2 ← getPadreConPathComp(indiceNodo2, padre,altura,0)
3 padreMenosAltura ← min(altura[padreNodo1],altura[padreNodo2])
4 padreMasAltura ← max(altura[padreNodo1],altura[padreNodo2])
5 padre[padreMenosAltura] ← padreMasAltura
```

ARMARGRAFOCOMPLETO(*nodos* : vectordeNodos)

```
1 listaAristas ← inicializar lista de incidencia
2 matrizAristas ← inicializar matriz de adyacencia
3 for i ← 0 to tam(nodos)
4     do
5         for j ← i + 1 to tam(nodos)
6             do
7                 armar arista con datos de v[i] y v[j]
8                 agregar arista a listaAristas
9 armar matriz de adyacencia con la lista de incidencia
10 Devolver Matriz de adyacencia y Lista de incidencia
```

RETIRAREJESINCONSISTENTES(*listaAristas* : *listaincidencia*, σ_T , *profVecindario*, f_T , *forma*, *cantidadDeClusters*, *vectordeenteros*)

```

1  for e : listaAristas
2      do
3          calcular media y desviacion respecto del vecindario de profVecindario
            de profundidad de cada extremo de e. (usando una modificación de BFS)
4          if e es inconsistente
5              then
6                  sacar e de las listas
7                  recorrer en la lista de ady todos los nodos alcanzables
                    de uno de los extremos y modificar su representante
                    en padre con cantidadDeClusters (usando una modificación de BFS)
8                  aumentar en 1 el valor de cantidadDeClusters

```

4.2. Prim

4.3. Complejidad

5. Experimentación

5.1. Variaciones

Los experimentos estarán centrados en analizar las diferentes tipos clusterizaciones que pueden realizarse variando las definiciones de eje inconsistente. Intentaremos analizar las configuraciones necesarias para que se pueda alcanzar una clusterización lo más cercana a la de la percepción humana y los resultados interesantes al que pueden llegarse.

Dados los siguientes: f_T multiplicador del promedio, σ_T multiplicador de la desviación, y la profundidad del vecindario de los extremos del eje candidato, $W(XY)$ el peso del eje candidato, y sea X e Y sus nodos extremos, definiremos un eje inconsistente:

- Forma 1: $\frac{W(XY)}{\text{Promedio}(\text{Vecindario}(X))} > f_T$ y $\frac{W(XY)}{\text{Promedio}(\text{Vecindario}(Y))} > f_T$,
Es decir, la proporción entre el peso del eje candidato y el promedio de peso del vecindario de sus extremos es mayor al coeficiente dado.
- Forma 2: $W(XY) > \text{Promedio}(\text{Vecindario}(X)) + \sigma_T * \text{desviacion}(\text{Vecindario}(X))$
y $W(XY) > \text{Promedio}(\text{Vecindario}(Y)) + \sigma_T * \text{desviacion}(\text{Vecindario}(Y))$,
Es decir, que el peso del eje candidato supere al promedio del vecindario de sus extremos por al menos σ_T unidades de la desviación del vecindario del extremo.
- Forma 3: Que se cumpla ambas

6. Arbitraje
7. Justificación teórica
8. Algoritmos presentados
- 8.1. Complejidad
9. Experimentación
10. Comparaciones de algoritmos de los problemas resueltos
11. Bibliografía

- https://en.wikipedia.org/wiki/Cluster_analysis
- https://en.wikipedia.org/wiki/Principles_of_grouping
- <https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a>
- <https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d>
- <https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms>