DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

Clustering

24 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 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 imagenes 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 algortimos 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

Aprendizaje No Supervisado

2.2. Heuristica para resolver el problema

Para resolver el problema utilizamos una **heuristica**, la misma esta basada en el método de detección de clustes 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 vertices 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 estandard 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 indice, entonces luego de recorrer todas

las aristas, todas las componentes conexas van a tener un indice 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 implentaron 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           Devolver indice
3     else
4           getPadre(indice(padre[indice]),padre)
5
```

getPadreConPathComp($entero : indice, padre : vectordeenteros, altura : vectordeenteros, nivelesSubidos : entero$

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

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

RETIRAREJESINCONSISTENTES($listaAristas : listaincidencia, \sigma_T, profVecindario, f_T, forma, cantidadDeCluster$ $vectordeenteros$)

```
1  for e : listaAristas
2  calcular media y desviacion respecto del vecindario de profVecindario de profundidad de cada extremo de e. (u
3  si es inconsistente
4  sacar e de las listas
5  recorrer en la lista de ady todos los nodos alcanzables de uno de los extremos y modificar su representante en
6  aumentar en 1 el valor de cantidadDeClusters
7
```

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: $\frac{W(XY)}{Promedio(Vecindario(X))} > f_T$ y $\frac{W(XY)}{Promedio(Vecindario(Y))} > f_T$,
  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: $W(XY) > Promedio(Vecindario(X)) + \sigma_T * desviacion(Vecindario(X))$ y $W(XY) > Promedio(Vecindario(Y)) + \sigma_T * desviacion(Vecindario(Y))$,
  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