Mineria de Datos

Clustering I

Dr. Edgar Acuna
Departmento de Matematicas

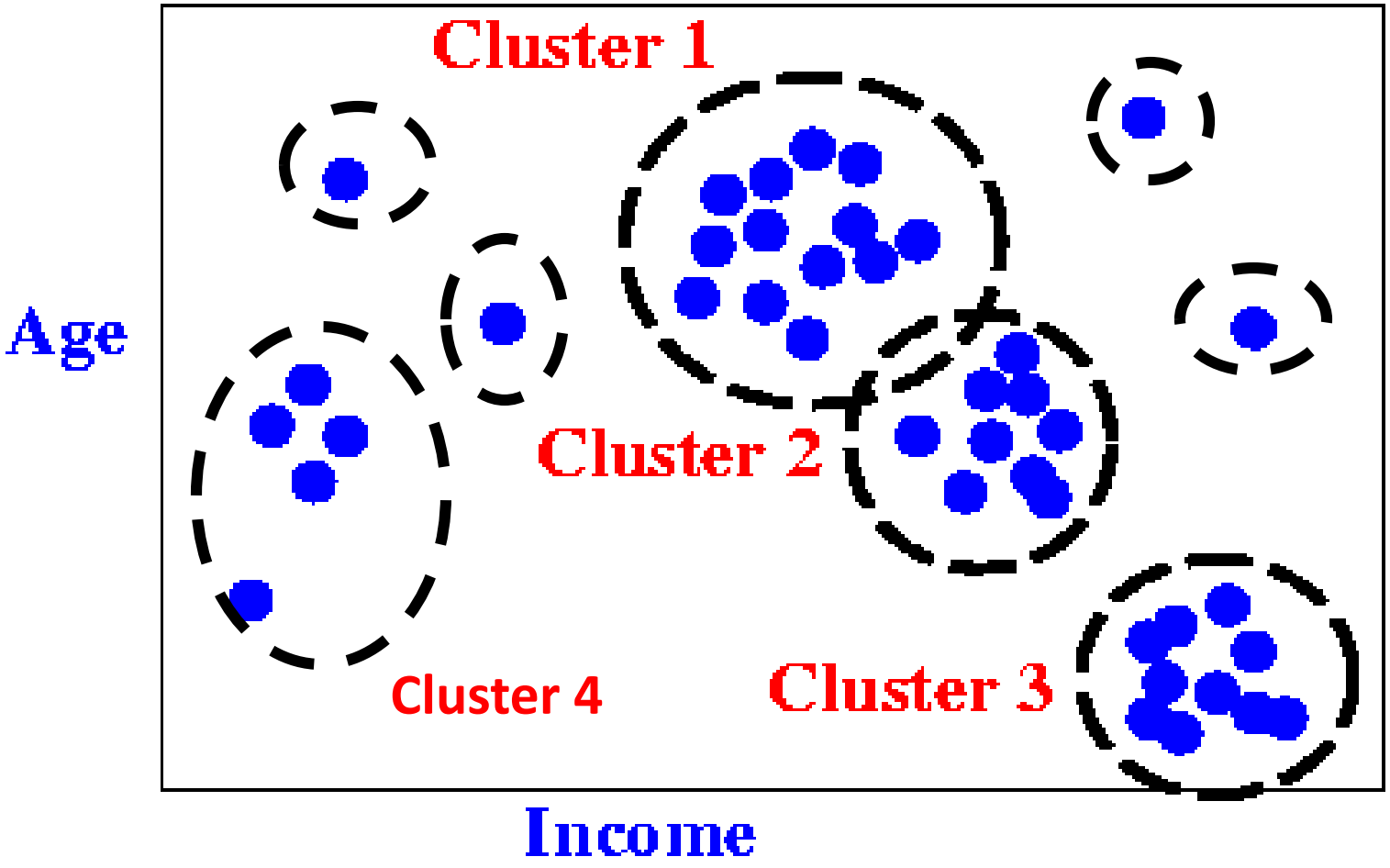
Universidad de Puerto Rico- Mayaguez

academic.uprm.edu/eacuna

Introduccion

- La idea de analisis de conglomerados (“clustering”) es agrupar muestras (filas) o features (columnas) o ambos a la vez, de acuerdo a la separación entre ellas determinada por una medida de distancia dada, llamada **medida de dissimilaridad**. Se supone que las clases a las que pertenecen las muestras no son conocidas.

- También es conocido con el nombre clasificacion no supervisado o Segmentacion.


Cluster 1
Age
Cluster 2
Cluster 4
Cluster 3
Income

- Para cada muestra (fila) existe un vector de mediciones $\mathbf{X}=(X_1,\ldots X_G)$.

- El objetivo es identificar grupos de muestras similares basado en n mediciones observadas $\mathbf{X}_1=\mathbf{x}_1,\ldots., \mathbf{X}_n=\mathbf{x}_n$.

- Por ejemplo si las X's representan niveles de expression obtenidos en microarreglos de tumores cancerosos uno podria identificar las caracteristicas de las personas que tienen distintos tipos de tumores.

- Cuando el numero de columnas es bastantes grande se pueden formar tambien grupos de columnas con similar comportamiento y en consecuencia se puede reducir la dimensionalidad que es muy conveniente si se quiere usar luego un modelo para hacer predicciones. Pues es mucho mas conveniente predecir con 10 features que con 100.

- También se puede aplicar conglomerados simultaneamente a filas y columnas (Bi-clustering) (ver paper de Alon, et al, 1999, Getz et al , 2000 y Lazaeronni y Owen, 2000)

Aspectos importantes en el analisis de conglomerados

i) Que features usar? ,

ii) Que medida de dissimilaridad usar?.

iii) Qué método o algoritmo de conglomerado usar?.

iv) Cuantos conglomerados se deben formar?

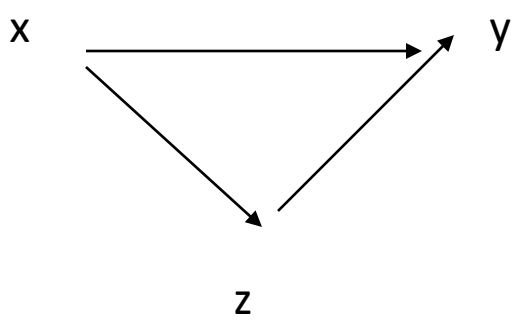
v) Cómo asignar las filas (o columnas) a los conglomerados?

vi) Cómo validar los conglomerados que se han formado?

Propiedades de medidas de dissimilaridad

- No-negatividad: $d(x,y) \geq 0$
- La distancia de una instancia asi mismo es 0, $d(x,x) = 0$
- Simetria: $d(x,y) = d(y,x)$
- Desigualdad Triangular:

  $d(x,y) \leq d(x,z) + d(z,y)$



Medidas de Dissimilaridad(variables continuas)

a) **Distancia Minkowski o norma Lp.**

$$D_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{M}(x_i - y_i)^p\right)^{1/p}$$

Caso particulares:

- Distancia Euclideana: p=2, $D_2 = \sqrt{\left(\sum_{i=1}^{M}(x_i - y_i)^2\right)}$
-
- Distancia Manhattan o City-Block: $D_1 = \sum_{i=1}^{M}| x_i - y_i |$
- Distancia Chebychev, p=∞, $D_\infty = \max_{1\le i\le M} | x_i - y_i |$
- La distancia ponderada Minkoswki, será: $D_p(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{M} w_i(x_i - y_i)^p\right)^{1/p}$

Medidas de Dissimilaridad (Cont)

b) Distancias basadas en formas cuadráticas.

$Q=(Q_{ij})$ es una matriz cuadrada MxM definida positiva de pesos entonces la distancia cuadrática entre x y y está dada por:

$$DQ(x,y) = [(x-y)'Q(x-y)]^{1/2} = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{M}(x_i - y_i)Q_{ij}(x_j - y_j)}$$

Si $Q=V^{-1}$, V matríz de covarianza entre x y y se obtiene la distancia Mahalanobis.

c) Distancia Camberra.

$$D_{Can}(x,y) = \sum_{i=1}^{M}\frac{|x_i - y_i|}{|x_i + y_i|}$$

Si x_i y y_i son ambos ceros entonces el i-ésimo término de la suma se considera como cero.

Medidas de dissimilaridad (variables nominales)

Distancia Hamming. Sean **x** y **y** dos vectores de la misma dimension y con valores en Ω={0,1,2,……..k-1}, entonces la distancia Hamming (D_H) entre ellos se define como el número de entradas diferentes que tienen los dos vectores. Por ejemplo si x=(1,1,0,0) y y=(0,1,0,1) entonces D_H=2.

- Tambien se pueden usar para variables no binarias.Por ejemplo, si x=(0,1,3,2,1,0,1) y y=(1,1,2,2,3,0,2) entonces DH(**x,y**)=4 .
- En el caso de variables binarias , la distancia Hamming, la distancia L2 y la distancia L1 coinciden.

Medidas de similaridad (Variables continuas))

Similaridad=1-dissimilaridad

a) Medida de correlación:

$$r(x,y) = \frac{\sum_{i=1}^{M}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{M}(x_i - \bar{x})^2 \sum_{i=1}^{M}(y_i - \bar{y})^2}} = \frac{(x - \bar{x})'(y - \bar{y})}{\| x - \bar{x} \| \| y - \bar{y} \|}$$

Nota: 1-s(x,y) puede ser considerado como una medida de dissimilaridad. La distancia coseno es simlar a la correlacion pero considerando que ls medias muestrales de x y y son ceros.

b) Medida de Tanimoto. Se define por

$$S_T(x,y) = \frac{x' y}{\| x \|^2 + \| y \|^2 - x' y}$$

Tambien puede ser usada para variables nominales

Medidas de similaridad (variables nominales)

i)La medida de Tanimoto. Sean X y Y dos vectors de longitud nx y ny respectivamente. Sea $n_{X \cap Y}$ que representa la cardinalidad de la interseccion, entonces la medida de Tanimoto se define por

$$\frac{n_{X \cap Y}}{n_X + n_Y - n_{X \cap Y}}$$

Es decir, la medida de Tanimono es la razón del número de elementos que los vectores tienen en común entre el número de elementos distintos. En el caso de variables binarias, la medida de Tanimoto se reduce a

$$\frac{a+d}{a+2(c+b)+d}$$

Donde a representa el número de posiciones donde los vectores X y Y coinciden en tomar el valor 0, d representa el número de coincidencias donde X y Y valen ambos 1. Mientras que c y b representan el número de no coincidencias.

ii) **El coeficientes de coincidencias simple**. Definido por

$$\frac{a+d}{a+b+c+d}$$

El problema de esta medida es que incluye a a en el numerador, la cual indica ausencia de ambos factores.

iii) **La medida de Jaccard-Tanimoto**. Definida por

$$\frac{d}{b+c+d}$$

iv) **La medida de Russel -Rao**.

$$\frac{d}{a+b+c+d}$$

v) **La medida de Dice-Czekanowski**. Similar a la de Jaccard pero asigna peso doble a las coincidencias. Es decir

$$\frac{2d}{b+c+2d}$$

Librerias y funciones en R para hallar distancias

La libreria **stats** de R tiene una funcion **dist** que calcula la matríz distancia de una matríz de datos usando las distancias Euclideana, Manhattan, Chebychev, Camberra, Mimkowski y binary..

```
librería(stats)
x =matrix(rnorm(20), nrow=5)
    dist(x)
    dist(x, method=”manhattan”,diag = TRUE)
    dist(x, method=”maximum”,upper = TRUE)
# Ejemplo de distancia Camberra entre dos vectores
    x <- c(0, 0, 1, 1, 1, 1)
    y <- c(1, 0, 1, 1, 0, 1)
    dist(rbind(x,y), method="canberra")
```

La libreria **cluster** de R tiene una función **daisy** que calcula la matriz distancia de una matriz de datos usando solamente las distancias euclideana y manhattan y considerando ademas distintos tipos de variables mediante el uso del coeficiente de Gower .

La función **cor** de R calcula la matríz de correlaciones entre las columnas de una matríz y

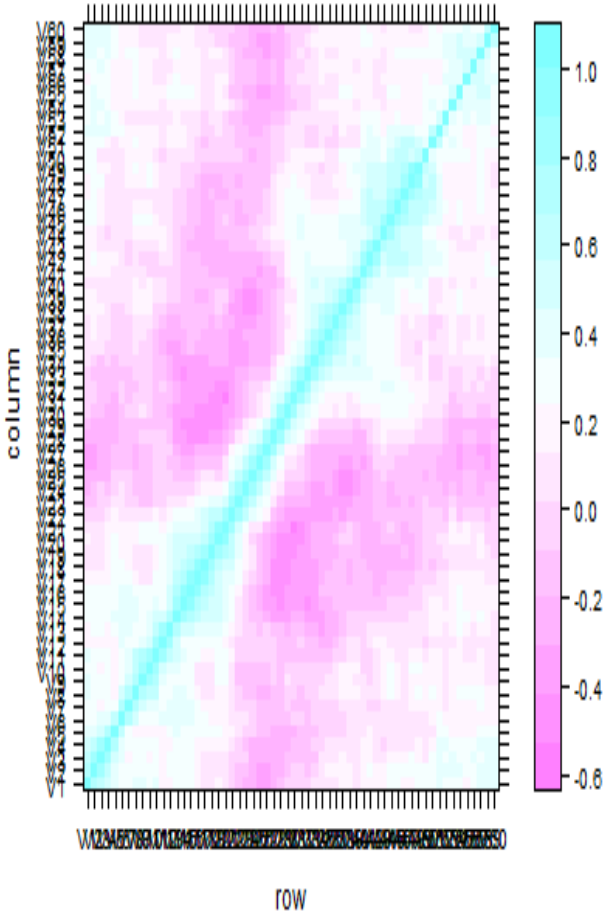
la función **levelplot** de la librería **lattice** permite hacer un "heatmap" de las correlaciones.

Plot de la matríz de corrrelaciones (sonar)

- Sonar, tiene 60 columnas y 208 filas.

levelplot(cor(sonar[,-1[])

- La parte **cyan** de la grafica indica que la correlación es bien alta y positiva, la parte **magenta** indica que es bien alta pero negativa y la parte **clara** indica que la correlación es **cerca de cero**.



Tipos de Algoritmos para Conglomerados ("Clustering")

I. Métodos de particionamiento (Kmeans, PAM)
II. Metodos Jerarquicos
III. Metodos basados en Modelos;(Gaussin Mixtures)
IV. Metodos basados en densidad (DBSCAN): Agrupa en un mismo cluster a instancias que tienen muchos vecinos cercanos proximos, de paso identifica outliers.
V. Metodos basados en Grids (STING, CLIQUE)

I. Métodos de particionamiento

El conjunto de datos es particionado en un número pre-especificado de conglomerados K, y luego iterativamente se va reasignando las observaciones a los conglomerados hasta que algún criterio de parada (función a optimizar) se satisface (suma de cuadrados con respecto a la media dentro de los conglomerados sea la más pequeña).

Ejemplos: **K-means, PAM,** CLARA, **SOM**, Conglomerados basados en modelos de mezclas Gaussianas, Conglomerados difusos, etc.

1-Algoritmo k-means (MacQueen, 1967).

El objetivo es minimizar la disimilaridad de los elementos dentro de cada cluster y maximizar la disimilaridad de los elementos que caen en diferentes clusters.

INPUT: Un conjunto de datos S y k número de clusters a formar;

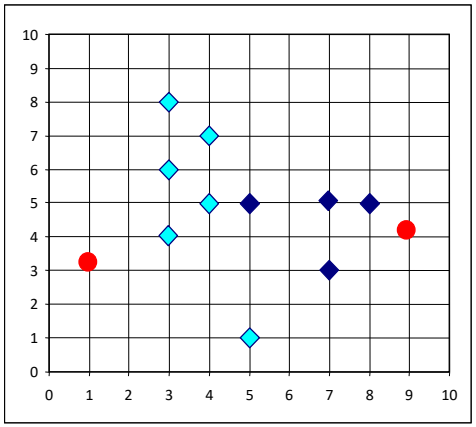
OUTPUT: L una lista de los clusters en que caen las observaciones de S.

1. Seleccionar los centroides iniciales de los K clusters: c_1, c_2, ..., c_K.
2. Asignar cada observación x_i de S al cluster C(i) cuyo centroide c(i) está mas cerca de x_i. Es decir, $C(i)=\text{argmin}_{1\leq k\leq K}\|\mathbf{x}_i-\mathbf{c}_k\|$
3. Para cada uno de los clusters se recalcula su centroide basado en los elementos que están contenidos en el cluster y minimizando la suma de cuadrados dentro del cluster. Es decir,

$$WSS = \sum_{k=1}^{K} \sum_{C(i)=k} \| \mathbf{x}_i - \mathbf{c}_k \|^2$$

Ir al paso 2 hasta que se consiga convergencia.

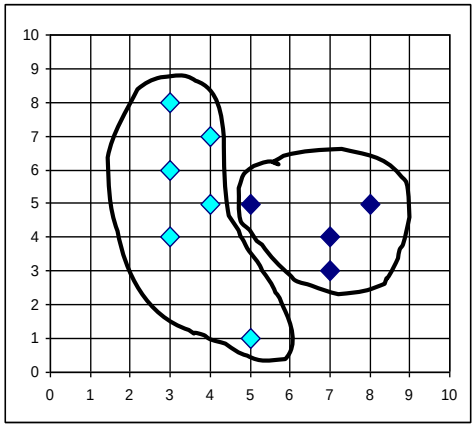
K-Means



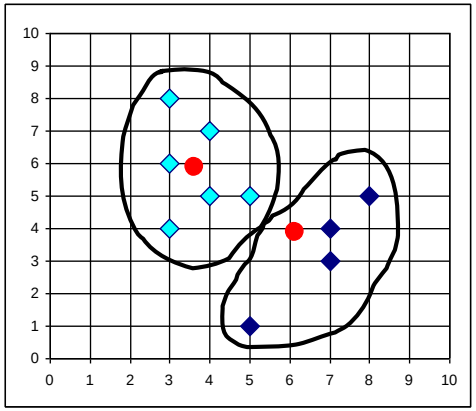
K=2

Escoger arbitrariamente K clusters usando al azar dos puntos como centroides de los mismos.

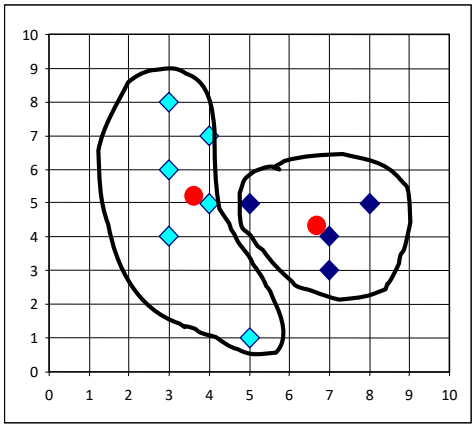
Asignar cada instancia al cluster mas cercano.



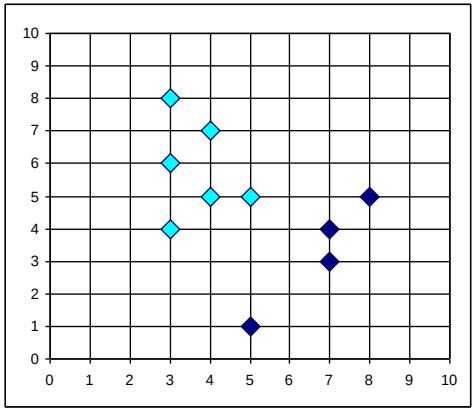
reasignar



Actualizar los centroides



reasignar



Actualizar los centroides

Alternativas para los k centroides iniciales

- Usando las primeras k observaciones
- Eligiendo aleatoriamente k observaciones.
- Tomando cualquier partición al azar en k clusters y calculando sus centroides.

Características del Algoritmo k-means

- No se satisface el criterio de optimizacion globalmente, solo produce un óptimo local.
- El algoritmo de k-means es computacionalmente rápido.
- Puede trabajar bien con datos faltantes (missing values).
- Es sensible a “outliers”.
- Complejidad: O(nkpi), n:numero de instancias, k=numero de clusters,p:numero de predictoras, i:numero de iteraciones. Pero para conjuntos bien grandes es basicamente O(n).

R y la función **kmeans**

R tiene la función **kmeans** que ejecuta el algoritmo **kmeans**.

Ejemplo: Consideremos conjunto de datos **bupa (**6 variables predictoras y 345 observaciones).

```
> kmeans(bupa[,1:6],2)
> kmeans(bupa[,1:6],3)
> #K-means eligiendo como centros iniciales las observaciones10 y 100 de
    Bupa
> medias<-bupa[c(10,100),1:6]
> kmeans(bupa[,1:6],medias)
```

En Python sklearn. cluster. KMeans hace k-means.

Rapidminer tambien hace kmeans primero se elige el operador modeling y luego el operador segmentation

Particionamiento alrededor de medoides (PAM)

Introducido por Kauffman y Rousseauw, 1987.

MEDOIDES, son instancias representativas de los clusters que se quieren formar.

Para un pre-especificado número de clusters K, el procedimiento PAM está basado en la búsqueda de los K MEDOIDES,

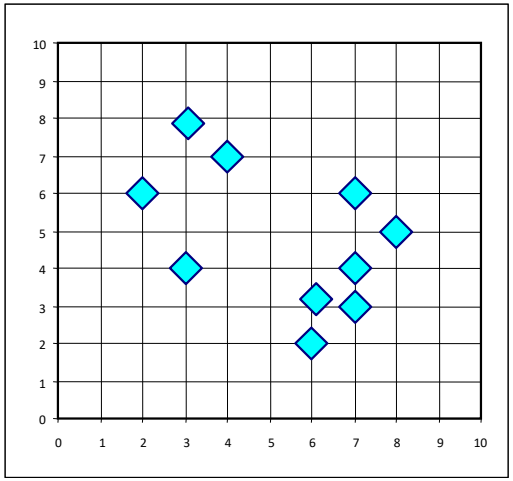
$\mathbf{M} = (\mathbf{m}_1, \ldots, \mathbf{m}_K)$ de todas las observaciones a clasificar

Para encontrar M hay que minimizar la suma de las distancias de las observaciones a su mas cercano Medoide.

$$M^* = \arg\min_M \sum_i \min_k d(x_i, m_k)$$

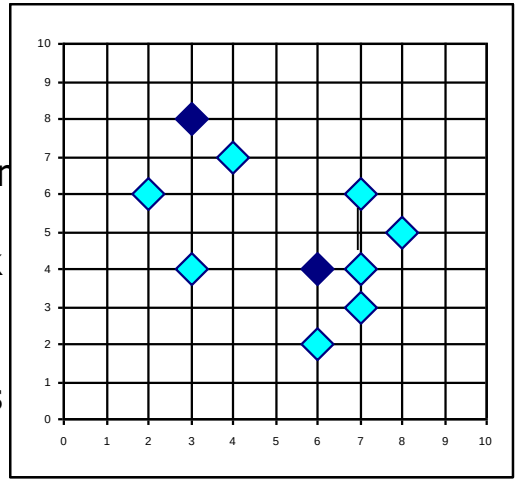
d es una medida de dissimilaridad

k-Medoides



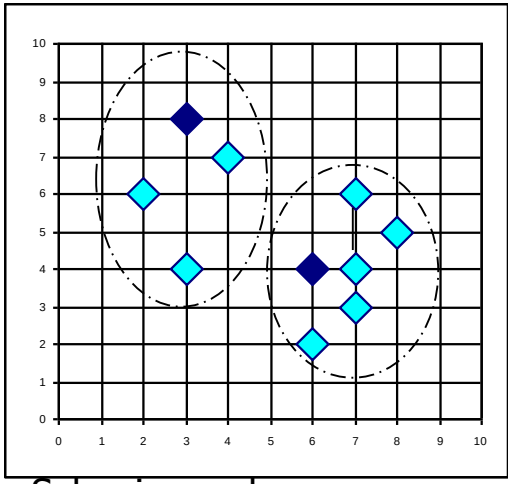
K=2

Arbitrariamente escoger k instances como medoides

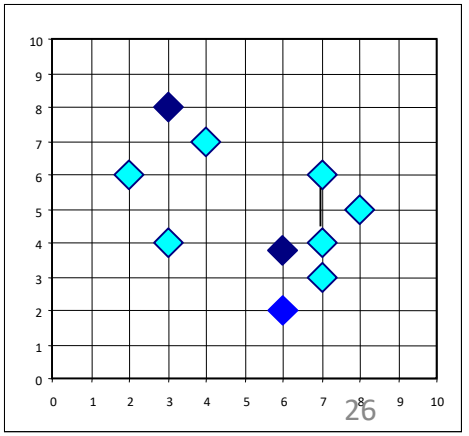


Asignar las instancias restantes a su mas cercano medoide.

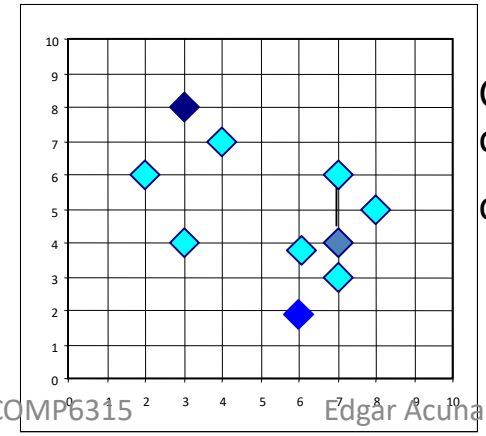
Total Cost = 20



Seleccionar al azar una instancia nonmedoide ,O_{ramdom}



Calcular el costo total de swapping



Total Cost = 26

Swapping O y O_{ramdom}

Si se mejora la calidad

Hacer el loop hasta que no haya cambios

R y la función PAM

La función pam de la librería cluster encuentra los conglomerados usando el particionamiento alrededor de medoides.

Aplicaremos PAM al conjunto de datos bupa.

>pambupa=pam(bupa[,1:6],2,diss=F)

Generalmente los resultados son un poco mejores que el de

k- means.

Complejidad del PAM :$O(k(n-k)^2)$

Rapidminer tambien hace k-medoides, primero se elige el operador modeling y luego el operador segmentation.

Scikit-sklearn no tiene implementado el PAM.

Mapas auto-organizados (Self-orgamizing Maps, SOM)(Kohonen, 1988)

Son algoritmos de particionamiento que son restringidos al hecho que los clusters pueden ser representados en una estructura regular de dimensión baja, tal como un grid (los más usados son los SOM en una y dos dimensiones).

Los clusters que son cercanos entre sí aparecen en celdas adyacentes del grid. O sea, SOM mapea el espacio de entrada de las muestras en un espacio de menor dimension en el cual la medida de similaridad entre las instancias es medida por la relación de cercanía de los vecinos.

Cada uno de los K clusters es representado por un objeto prototipo Mi , i = 1,…K .

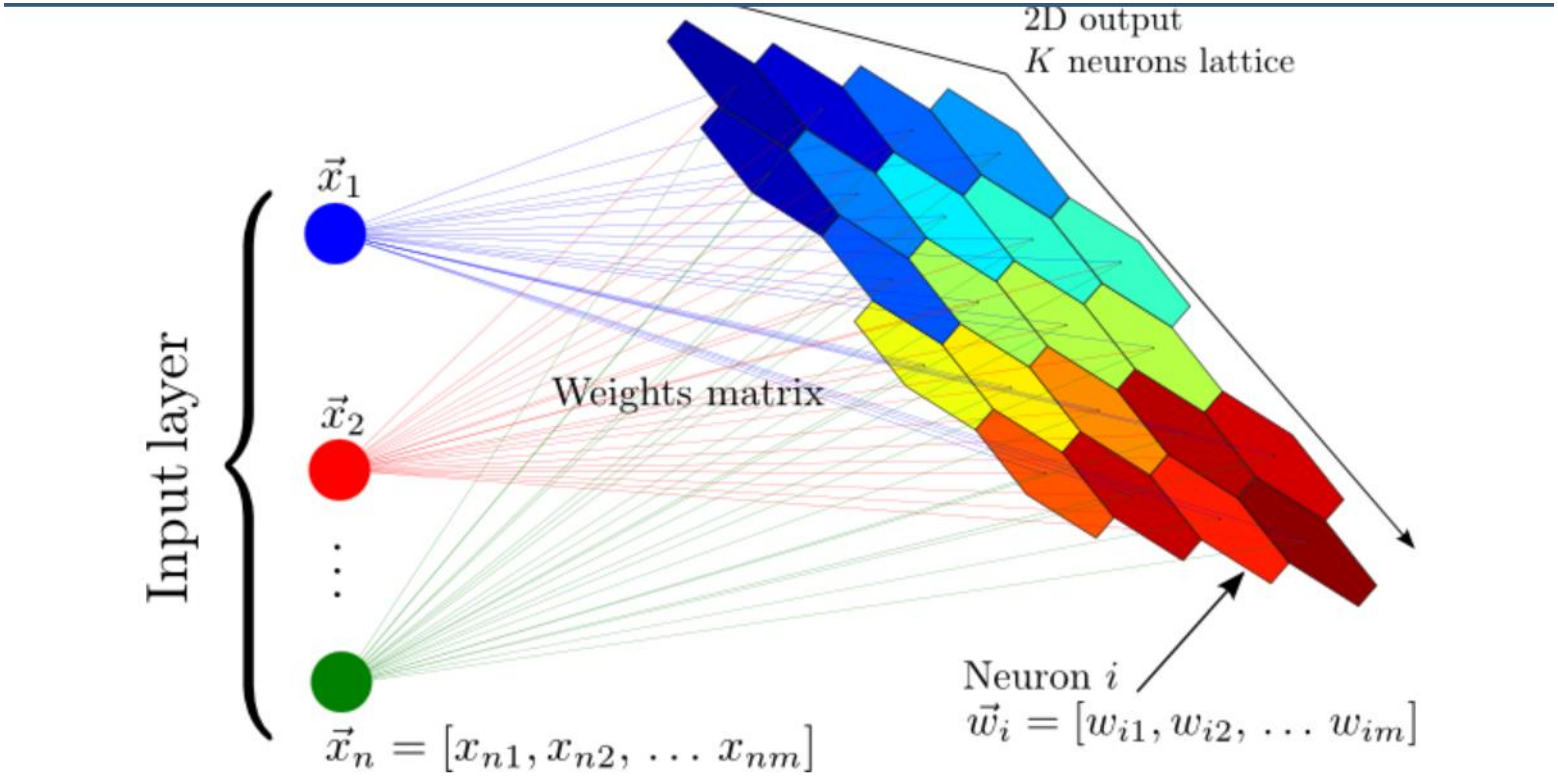
Mapas auto-organizados (Self-orgamizing Maps, SOM)

En este método la posición de una observación en el espacio inicial de entradas influye de alguna manera en su asignación a un conglomerado.

Se construye un plot en donde las observaciones similares son ploteadas cercanas entre sí.

El algoritmo SOM es bastante similar a k-means
Un SOM es entrenado como una red neural.
El SOM puede ser considerado también como una técnica de reducción de dimensionalidad y guarda relación con la técnica conocida como escalamiento multidimensional (proceso de encontrar un conjunto de puntos en un espacio multidimensional que son similares entre si.)

Visualizando una SOM



Overview of the SOM neural network

El algoritmo SOM considerando un grid rectangular de dos dimensiones

Paso1-Seleccionar los numeros de filas (q_1) y columnas (q_2) en el grid. Luego habrá $K=q_1q_2$ clusters

Paso2. Inicializar el tamaño del parámetro de actualizacion (la tasa de aprendizaje en terminos de redes neurales) α (α=1) y el radio del grid (r=2)

Paso3. Inicializar los vectores prototipos M_j, j ε (1,…,q1) x (1,….,q2) mediante la eleccion aleatoria de K instancias.

Paso4. Para cada observacion x del conjunto de datos hacer los siguiente:

Identificar el vector índice j' del prototipo Mj mas cercano a xi.

Identificar un conjunto S de prototipos vecinos de $M_{j'}$. Es decir,

$S=\{j: \text{distancia } (j, j')<r\}$

La distancia puede ser Euclideana o cualquiera otra

Actualizar cada elemento de S moviendo el correspondiente prototipo hacia x_i:

$M_j \leftarrow M_j + \alpha(x_i - M_j)$ para todo j ε S

Paso5. Disminuir el tamaño de α y r en una cantidad predeterminada y continuar hasta alcanzar convergencia.

R y la función SOM

La función **SOM** de la librería **class** de B. Ripley construye un SOM, al igual que la libreria **som.** **Aunque los resultados son solamente visuales. De cada centroide salen lineas que representan cada una de las variables**

Hoy en dia es mejor usar la libreria som de R.

Otros software: Somtoolbox en Matlab, Cluster por Eissen y Genecluster (MIT Center por Genome Research). Este ultimo muestra los elementos de cada cluster y Rapidminer

SOM usando la libreria SOM

```
library(som)
vehicle.f=filtering(vehicle[,-19])
vehicle.f.n=normalize(vehicle.f)
som.vehi=som(vehicle.f.n,topol="hexa",xdim=2,ydim=2)
summary(som.vehi)
print(som.vehi)
attributes(som.vehi)
$names
 [1] "data"      "code"      "visual"    "qerror"    "init"      "alpha"     "neigh"
"topol"     "alpha0"    "radius0"   "rlen"      "xdim"      "ydim"
[14] "err.radius" "inv.alp.c"  "code.sum"

$class
[1] "som"
Plot(som.vehi)
dim(som.vehi$visual[som.vehi$visual$x==0 &som.vehi$visual$y==0,])
```

Aplicando SOM a los datos vehicle