Clustering (SSS)

Still Seaching for Species

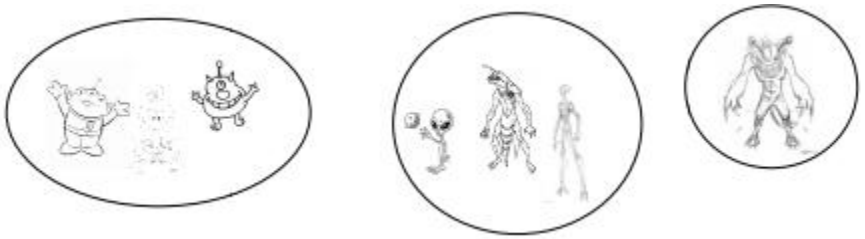
José Manuel Hidalgo Zueras

Índice

1. Introducción.
2. En que consiste el Clustering?
3. Situación del proyecto en el marco de la minería de datos.
4. Estructuras de datos utilizadas.
5. Funciones adicionales requeridas.
6. Implementación del algoritmo.
7. Test y ejemplos.
8. Mejoras.
9. Conclusiones.

1. Introducción

- Dado un dataset tenemos que agrupar a los individuos en clústers (grupos) según la proximidad de sus atributos.



- Uso del Clustering con el algoritmo K-Means para solucionar el problema planteado.

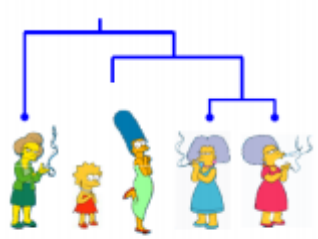
2. En que consiste el Clustering?

Separar un dataset no clasificado en grupos aparentemente ocultos basados en la proximidad o semejanza de los individuos.

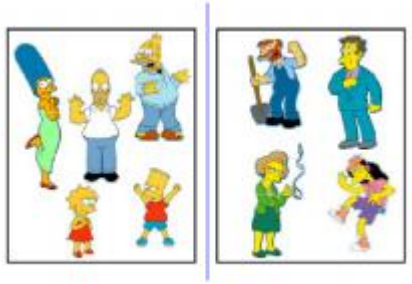
- Una buena solución se basa en dos principios:
  - Homogeneidad
  - Separación

2. En que consiste el Clustering?

- Tipos de Clustering:
  - Hieratical Clustering (organizado en arboles)
    - Agglomerative
    - Dismisive



  - Partitional Clustering (organizado en grupos)



3. Situación del proyecto en el marco de minería de datos.

- Trabajaremos con el Partitional Clustering haciendo uso del algoritmo K-MEANS.

- Agrupación
  - Agrupamos cada individuo al clúster que le corresponda.

- No supervisada
  - No necesitamos saber la clase.
  - Agrupamos individuos al azar.

4. Estructuras de datos utilizadas

```
typedef struct {
        float centroid [numatributos]; //punto central del clúster.
        int index [individuos]; // índice de cada individuo del DS.
        char clase [20]; //clase mayoritaria del clúster.
} cluster;

cluster clust [k]; //tantos clúster como k centroids.
```

5. Funciones auxiliares requeridas.

// Inicializa aleatoriamente los K centroids de cada clúster.

- InicializaCentroids(clust);

// Asigna el individuo A al clúster correspondiente.
// Función llamada K veces (una por cada centroid)

- DistanciaEntreIndividuos(IndividuoA, $centroid_i$);

// Recalcula el centroid de los K clústers según la mediana de
// los atributos del clúster.

- RecalculaCentroids(clust);

6. Implementación.

Algoritmo K-means – Procedimiento

- Objetivo: Agrupar individuos del DS por similitud.

1. Inicializar el centroid con valores aleatorios en el dominio del atributo. Aplicar info privilegiada, si hay.
2. Comprobar la distancia que hay entre cada individuo del DS respecto al centroid de cada cluster.
3. Recalcular el centroid de los K cluster si no se han estabilizado. Criterio recálculo.
4. Repetir los puntos II y III hasta conseguir estabilizar el centroid de TODOS los cluster.

6. Implementación.

Algoritmo K-means - Pseudocódigo

```
INICIALIZA CENTROID para los K cluster
Mientras centroids_no_estabilizados
    Para cada individuo del DS
        Para cada cluster de los K cluster
            Si DISTANCIA entre individuo y centroid es la Mínima
                AGREGAR_INDIVIDUO_A_CLUSTER
            FinSi
        FinPara
    FinPara
    RECALCULA_K_CENTROIDS_SI_NO_ESTABILIZADOS
FinMientras
```

6. Implementación.

Algoritmo K-means Best Solution - Pseudocódigo

```
Para cada K
    INICIALIZA CENTROID para los K cluster
    Mientras centroids_no_estabilizados
        Para cada individuo del DS
            Para cada cluster de los K cluster
                Si DISTANCIA entre individuo y centroid es la Mínima
                    AGREGAR_INDIVIDUO_A_CLUSTER
                FinSi
            FinPara
        FinPara
        RECALCULA_K_CENTROIDS_SI_NO_ESTABILIZADOS
    FinMientras
    GUARDA_SOLUCION_PARA_K_ACTUAL
FinPara
```

6. Implementación

De la Teoría a la Práctica – Estructura de Datos

Cluster

- Atributos
- Métodos

```
class Cluster {

private:
    int     iTotalDataset;
    float   *fCentroid;
    int     iTotalAttributes;   // Indica la dimension del array del centroid
    int     *iIndividuo;        // # indice del individuo en el dataset. Es dinámico pq no sabemos el nº.
    int     iTotalIndividuos;   // longitud del array *individuo.
    float   *fDistancia;        // Distancia del individuo. Es dinámica y se crea conjuntamente con el ind.
    float   *fRowToAvg;         // Mantiene la suma de los atts para cada individuo del cluster.
    float   *fMax;              // Max valor del atributo
    float   *fMin;              // Min valor del atributo
    int     iRecalculos;        // Indica el numero de veces que se ha recalculado el centroid de este cluster.


public:
    Cluster();
    Cluster(int iTotalAttributes,
            int iTotalIndividuos,
            clsAttribute *atts);
    Cluster &operator= (const Cluster & m);     // Sobrecarga de operadores
    ~Cluster(void);                             // Destructor

    // Setters
    void    addIndividuoAndDistance(int iIndividuo, float fDistance);
    void    initCentroid(void);                 // Inicializa el centroid de este cluster a random
    void    setCentroid(float *new_centroid);   // Recibe una row de floats. Un valor para cada atributo.
    void    addToAvg(float *fRow);              // Agrega individuo para futura media.
    void    init_fRowToAvg(void);
    int     setMaxMinRange(clsAttribute *atts);
    void    clearCluster(void);
    void    incrRecalculalos(void);

    // Getters
    float   *getCentroid(void);     // Devuelve el centroid de este cluster
    int     getTotalAttributes(void);
    int     getTotalIndividuos(void);
    // Carga por referencia el indice del individuo y su distancia calculada.
    void    loadIndividuoAndDistance(int iPos,
                                     int *iIndIndex,
                                     float *fIndDistance);
    int     getIndividuo(int iPos);
    float   getDistance(int iPos);
    float   *getMaxRange(void);
    float   *getMinRange(void);
    float   *getAvg(void);          // Devuelve la media calculada del array fRowToAvg.
    int     getRecalculos(void);
};
```

No se detalla clsAttribute

6. Implementación

De la Teoría a la Práctica – Estructura de Datos – Atributos

```
private:
    int     iTotalDataset;
    float   *fCentroid;
    int     iTotalAttributes;   // Indica la dimension del array del centroid
    int     *iIndividuo;        // # indice del individuo en el dataset. Es dinámico pq no sabemos el nº.
    int     iTotalIndividuos;   // longitud del array *individuo.
    float   *fDistancia;        // Distancia del individuo. Es dinámica y se crea conjuntamente con el ind.
    float   *fRowToAvg;         // Mantiene la suma de los atts para cada individuo del cluster.
    float   *fMax;              // Max valor del atributo
    float   *fMin;              // Min valor del atributo
    int     iRecalculos;        // Indica el numero de veces que se ha recalculado el centroid de este cluster.
```

6. Implementación

De la Teoría a la Práctica – Estructura de Datos – Métodos

```
public:
    Cluster();
    Cluster(int iTotalAttributes,
            int iTotalIndividuos,
            clsAttribute *atts);
    Cluster &operator= (const Cluster & m);      // Sobrecarga de operadores
    ~Cluster(void);                              // Destructor

    // Setters
    void    addIndividuoAndDistance(int iIndividuo, float fDistance);
    void    initCentroid(void);                  // Inicializa el centroid de este cluster a random
    void    setCentroid(float *new_centroid);    // Recibe una row de floats. Un valor para cada atributo.
    void    addToAvg(float *fRow);               // Agrega individuo para futura media.
    void    init_fRowToAvg(void);
    int     setMaxMinRange(clsAttribute *atts);
    void    clearCluster(void);
    void    incrRecalculalos(void);

    // Getters
    float   *getCentroid(void);                  // Devuelve el centroid de este cluster
    int     getTotalAttributes(void);
    int     getTotalIndividuos(void);
    // Carga por referencia el indice del individuo y su distancia calculada.
    void    loadIndividuoAndDistance(int iPos,
                                     int *iIndIndex,
                                     float *fIndDistance);
    int     getIndividuo(int iPos);
    float   getDistance(int iPos);
    float   *getMaxRange(void);
    float   *getMinRange(void);
    float   *getAvg(void);                       // Devuelve la media calculada del array fRowToAvg.
    int     getRecalculos(void);
```

6. Implementación

Relación Funciones Pseudocódigo – Implementación

INICIALIZA CENTROID

```
// Inicializa el centroid de este cluster con valores aleatorios para cada uno de los atributos, siempre en el rango de dominio del atributo.
void initCentroid(void);
```

DISTANCIA

```
// Recibe las rows sobre las que aplicar la funcion de distancia. Se le pasa un individuo del DS y el centroid de uno de los cluster.
float MarsMinerMainWin::DistanceBetweenRows(float *rowA, float *rowB, int start, int end, clsAttribute *atts);
```

6. Implementación

Relación Funciones Pseudocódigo – Implementación

AGREGAR_INDIVIDUO_A_CLUSTER

```
// Agrega el índice que ocupa el individuo en el DS en la posición indicada por iTotalIndividuos y su distancia. Permite agilizar la visualización de los resultados sobre las gráficas.
void addIndividuoAndDistance(int iIndividuo, float fDistance);
```

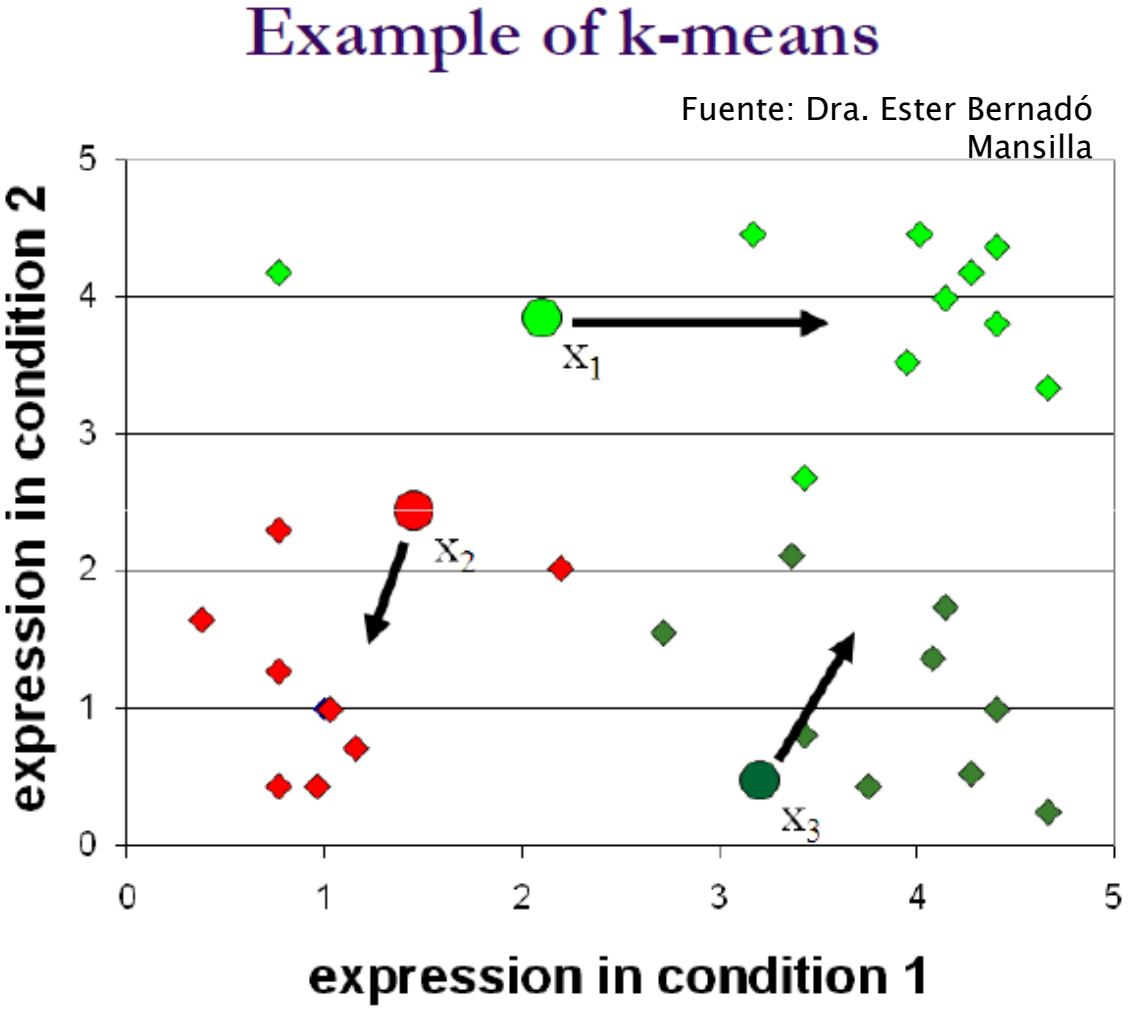
RECALCULA_K_CENTROIDS_SI_NO_ESTABILIZADOS

```
// Establece como nuevo centroid, el valor medio de los atributos que tienen los individuos asignados al cluster del centroid.  Lo hace si la distancia de la media con respecto al centroid, es superior al límite especificado.
bool MarsMinerMainWin::recalculaCentroids(int K, Cluster **clusters, clsAttribute *attts);
```

6. Implementación

Visualización de los Resultados

Representación válida para individuos de 2 atributos.



PROBLEMA

- Tenemos individuos con N atributos.

6. Implementación

Visualización e Interpretación de los Resultados

SOLUCIÓN

- Representar los individuos de cada clúster por distancia a su centroid.

Results for K = 7

Cluster 0 results
Purple 0
Red 27
Clustering Rate 100.00
Re calcultate # 4

Cluster 1 results
Purple 50
Red 50
Clustering Rate 50.00
Re calcultate # 4

Cluster 2 results
Purple 0
Red 23
Clustering Rate 100.00
Re calcultate # 4

Cluster 3 results
Purple 0
Red 0
Clustering Rate 100.00
Re calcultate # 4

Cluster 4 results
Purple 0
Red 0
Clustering Rate 100.00
Re calcultate # 4

Cluster 5 results
Purple 50
Red 0
Clustering Rate 100.00
Re calcultate # 4

Cluster 6 results
Purple 0
Red 0
Clustering Rate 100.00
Re calcultate # 4

Fuente: José Manuel Hidalgo Zueras

7. Test y ejemplos.

Visualización e Interpretación de los Resultados

Ejemplos mediante Aplicación MarsMiner

8. Mejoras.

- Pre Proceso de la información.
- Ponderación de los atributos si existe conocimiento previo del DS.
- Validación de los clúster resultantes.

9. Conclusiones

- ✓ Permite organizar conjuntos de datos por similitud
- ✓ Se busca un clustering rate próximo al 100%
- ✓ La inicialización del centroid es clave
- ✓ Función de distancia es clave
- ✓ La representación en 2D habitual no presenta todos los atributos. Necesaria una gráfica 2D para cada par.

Preguntas?