Algoritmo K-Means

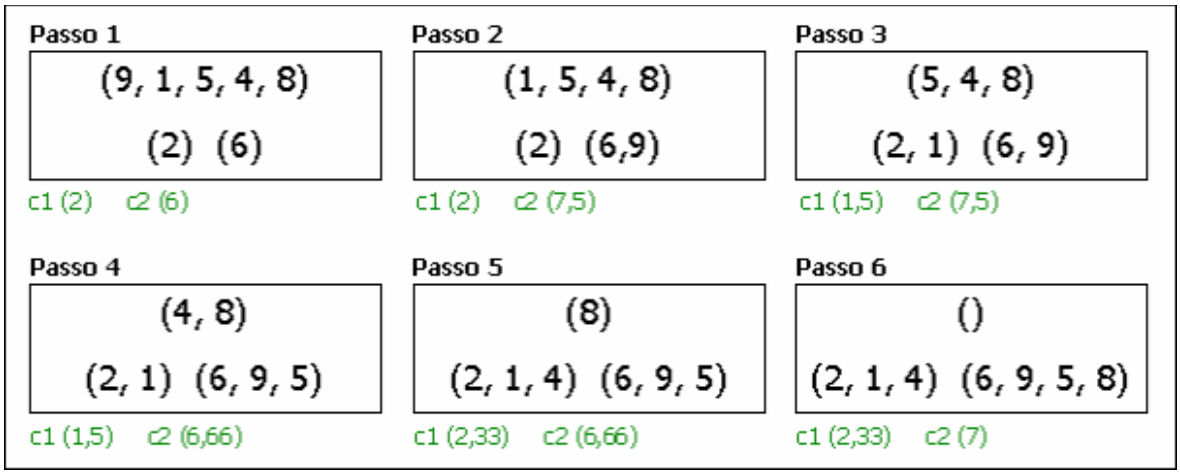
O objetivo do algoritmo K-Means é fornecer uma classificação de informações de acordo com os próprios dados. Esta classificação é baseada em análise e comparações entre os valores numéricos dos dados. Desta maneira, o algoritmo vai fornecer uma classificação automática sem a necessidade de nenhuma supervisão humana, ou seja, sem nenhuma pré-classificação existente. Por causa desta característica, o K-Means é considerado como um algoritmo de mineração de dados não supervisionado.

O algoritmo inicia com a escolha dos k elementos que formaram as sementes iniciais. Escolhidas as sementes iniciais, é calculada a distância de cada elemento em relação às sementes, agrupando o elemento ao grupo que possuir a menor distância (mais similar) e recalculando o centróide do mesmo. O processo é repetido até que todos os elementos façam parte de um dos clusters.

Após agrupar todos os elementos, procura-se encontrar uma partição melhor do que a gerada arbitrariamente. Para isto, calcula-se o grau de homogeneidade interna dos grupos através da Soma de Quadrados Residual (SQRes), que é a medida usada para avaliar o quão boa é uma partição.

Após o cálculo, move-se o primeiro objeto para os demais grupos e verifica-se se existe ganho na Soma de Quadrados Residual, ou seja, se ocorre uma diminuição no valor da SQRes. Existindo, o objeto é movido para o grupo que produzir o maior ganho, a SQRes dos grupos é recalculada e passa-se ao objeto seguinte. Depois de um certo número de iterações ou não havendo mais mudanças, o processo é interrompido.

Segue uma imagem que mostra como o processo funciona.

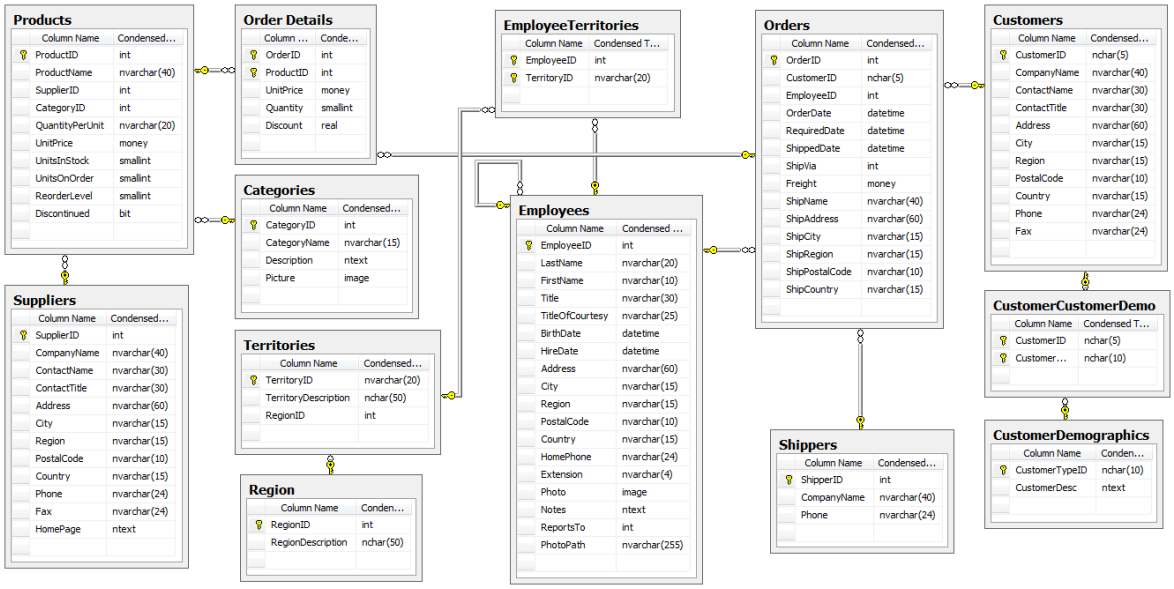


Segue o algoritmo:

```
Especifique k
Selecione os k objetos que serão os centróides dos agrupamentos
para todos os objetos restantes faça
  Calcule a distância entre o elemento e os centróides
  Adicione o elemento ao agrupamento que possuir a menor distância
  Recalcule o centróide do agrupamento
fim para
para todos os k agrupamentos faça
  Calcule a Soma de Quadrados Residual
fim para
repita
  para todos os n elementos faça
   Mova o elemento para os outros agrupamentos
   Recalcule a Soma de Quadrados Residual
      se soma dos Quadrados Residual diminuiu então
     O objeto passa a fazer parte do grupo que produzir maior ganho
     Recalcule a Soma de Quadrados Residual dos grupos alterados
      fim se
  fim para
até Número de interações = i ou Não ocorra mudança de objetos
```

Exemplo de uso do Algoritmo K-Means

Neste exemplo vamos considerar que uma determinada empresa vende produtos para clientes por meio de pedidos compostos por itens de pedidos. Para facilitar o entendimento do cenário e do modelo de dados vamos utilizar a base de dados de exemplo Northwind do SQL Server. O diagrama de entidades com as tabelas que nos interessa é apresentado logo abaixo.



A tabela que contém os itens de pedidos se chama Order Details e possui uma chave primária composta nas colunas OrderID e ProductID. Existem duas chaves estrangeiras na tabela Order Details, sendo que a primeira chave estrangeira relaciona a coluna ProductID da tabela de produtos chamada Products com a coluna ProductID da tabela Order Details. A segunda chave estrangeira relaciona a coluna OrderId da tabela de pedidos chamada Orders com a coluna OrderId da tabela Order Details. O modelo ainda apresenta a tabela de clientes Customers relacionada à tabela de pedidos Orders por meio das colunas CustomerID presente em ambas as tabelas.

A ideia será segmentar os clientes para poder oferecer benefícios. A segmentação deve dividir todos os clientes da base de dados em três categorias: “A”, “B” e “C”. O critério de classificação dos clientes deve levar em consideração

apenas duas variáveis: o total de pedidos de cada cliente e a quantidade total gasta pelo cliente nos pedidos. Os clientes que possuírem mais pedidos e o maior valor de compra serão classificados como “A”.

Inicialmente calculamos o total de pedidos para cada cliente e a quantidade gasta pelo cliente em todos os pedidos por meio da consulta vista logo abaixo. O resultado desta consulta é armazenado numa tabela chamada PERFIL.

```
select o.customerid,                              -- Cliente
      count(distinct o.orderid) as qtd_pedidos,-- total de pedidos
      sum(d.unitprice*d.quantity) as qtd_gasta  -- total de valor gasto
into perfil
from orders as o, [order details] as d
where o.orderid = d.orderid
group by o.customerid
order by o.customerid
```

Para realizar a classificação desejada nos dados da tabela PERFIL utilizaremos o algoritmo K-Means. Dois atributos serão utilizados para classificar os clientes: Quantidade de Pedidos e Total Gasto. No entanto, o algoritmo K-Means pode trabalhar com qualquer quantidade de atributos para classificar os valores.

Esta implementação foi colocada na stored procedure ST_KMEANS. Para tornar mais modular o algoritmo, a implementação do cálculo da distância entre os pontos foi feita na função DIST(), que deve ser criada antes da stored procedure.

A listagem a seguir apresenta a chamada da stored procedure ST_KMEANS para o exemplo da tabela PERFIL. O primeiro parâmetro que deve ser passado para a procedure é o nome da tabela, seguido pelos parâmetros dos atributos. O quarto parâmetro indica qual é a quantidade de classificações que o algoritmo deve utilizar (clusters). A listagem abaixo apresenta os 23 primeiros pontos classificados de acordo com o algoritmo K-Means, colocando-os os em ordem de acordo com a classe a que pertencem.

```
/* EXECUTANDO A STORED PROCEDURE KMEANS NA TABELA PERFIL */
EXEC ST_KMEANS 'PERFIL','QTD_PEDIDOS','QTD_GASTA',3
```

	X	Y	CLUSTER
1	1.0000	100.8000	1
2	2.0000	357.0000	1
3	2.0000	1488.7000	1
4	3.0000	522.5000	1
5	3.0000	649.0000	1
6	3.0000	1571.2000	1
7	3.0000	1719.1000	1
8	3.0000	1947.2400	1
9	3.0000	3172.1600	1
10	3.0000	5297.8000	1
11	4.0000	1402.9500	1
12	4.0000	1615.9000	1
13	4.0000	1992.0500	1
14	4.0000	2423.3500	1
15	4.0000	3361.0000	1
16	4.0000	3490.0200	1
17	5.0000	836.7000	1
18	5.0000	1467.2900	1
19	5.0000	1480.0000	1
20	5.0000	2844.1000	1
21	5.0000	3063.2000	1
22	5.0000	3460.2000	1
23	5.0000	3810.7500	1

No resultado apresentado pela stored procedure a coluna X equivale ao valor da primeira coluna passada como parâmetro, que no nosso exemplo é QTD_PEDIDOS, e a coluna Y equivale ao valor da segunda coluna passada como parâmetro, que no nosso exemplo é QTD_GASTA.

A Stored Procedure ainda possui um quinto parâmetro. Se este parâmetro não for passado, a stored procedure retorna os dados classificados como na listagem anterior. Se o quinto parâmetro for passado como “1”, por exemplo, a stored procedure retorna as coordenadas dos centróides de cada classe. A listagem a seguir apresenta a chamada à stored procedure com o uso do quinto parâmetro e o seu resultado.

```
/* EXECUTANDO A STORED PROCEDURE KMEANS NA TABELA PERFIL */
/* OBTENDO AS COORDENADAS DOS CENTRÓIDES */
EXEC ST_KMEANS 'PERFIL','QTD_PEDIDOS','QTD_GASTA',3,1
```

	ID	CX	CY
1	1	6.4667	5519.0832
2	2	13.5769	26035.3900
3	3	29.6667	115464.4867

No nosso exemplo, o algoritmo classificou os dados em três classes: classe 1, 2 e 3. De acordo com a definição do tipo de cliente a classe 1 equivale ao Cliente C, a classe 2 equivale ao Cliente B e a classe 3 equivale ao Cliente A.

Vejamos na prática o funcionamento.