

K-Means

[Data,Proto]=GenerateData(Centers,Dispersions,NumbersOfData);

Script:

GenerateData.m

Scopo:

Generazione dati e prototipi di cluster inizializzati in modo casuale per il testing dell'algoritmo KMeans.

Input:

Centers, NxM, centri dei cluster dei dati;
Dispersions, Nx1, dispersione di ogni cluster di dati;
NumbersOfData, Nx1, numero di dati in ogni cluster.

Output:

Data, sum(NumbersOfData)xM, lista di dati generati dallo script;
Proto, NxM, inizializzazione casuale dei prototipi dei cluster.

Note:

N = numero di cluster;
M = dimensione spazio delle caratteristiche (M = 2 per la visualizzazione).

Esempio:

Centers=[1 2; 3 4; 5 6];
Dispersions=[0.2 0.5 1];
NumbersOfData=[50 30 20];
[Data,Prot]=GenerateData(Centers,Dispersions,NumbersOfData);



[KProto,Class]=KMeans(Data,Proto,MaxNIter);

Script:

KMeans.m

Scopo:

Algoritmo KMeans.

Input:

Data, DxM, dati;
Proto, Nx1, prototipi delle classi inizializzati;
MaxNIter, 1, massimo numero di iterazioni di KMeans.

Output:

KProto, NxM, prototipi calcolati con KMeans ;
Class, Dx1, classificazione di ogni dato.

Note:

N = numero di cluster;
D = numero di dati;
M = dimensione spazio delle caratteristiche (M = 2 per la visualizzazione).

Esempio:

...

Test...

Verificare il comportamento dell'algoritmo KMeans (bontà della classificazione, presenza di unità morte, ...) al variare di:

- Inizializzazione dei prototipi;
- Sovrapposizione tra cluster (dispersione dei dati);
- Cluster con numero di dati differenti;
- Numero di cluster;
- ...

Es. 1) Buona inizializzazione, bassa dispersione dei dati, numero di dati bilanciato tra cluster

```
[Data,KProto]=GenerateData([-1 -2; 3 4; 2 -1],[0.1 0.1 0.1],[50 50 50]);  
KMeans(Data,Prot,10);
```

Es. 2) Cattiva inizializzazione, bassa dispersione dei dati, numero di dati bilanciato tra cluster

```
[Data,KProto]=GenerateData([-1 -2; 3 4; 2 -1],[0.1 0.1 0.1],[50 50 50]);  
Prot=[0 1; -1 -1; -1 0];  
KMeans(Data,Prot,20);
```

Es. 3) Buona inizializzazione, bassa dispersione dei dati, numero di dati sbilanciato tra cluster

```
[Data,KProto]=GenerateData([-1 -2; 3 4; 2 -1],[0.1 0.1 0.1],[20 20 100]);  
KMeans(Data,Prot,10);
```

Es. 4) Cattiva inizializzazione, bassa dispersione dei dati, numero di dati sbilanciato tra cluster

```
[Data,KProto]=GenerateData([-1 -2; 3 4; 2 -1],[0.1 0.1 0.1],[20 20 100]);  
Prot=[0 1; 1 -1; -1 0];  
KMeans(Data,Prot,10);
```

Es. 5) Buona inizializzazione, alta dispersione dei dati, numero di dati bilanciato tra cluster

```
[Data,KProto]=GenerateData([-1 -2; 3 4; 2 -1],[1 1 1],[50 50 50]);  
KMeans(Data,Prot,10);
```

...

Note:

per verificare se 5 è un'unità morta:
find(Class==5)

per plottare i prototipi veri sull'immagine
hold on;
plot (Centers(:,1),Centers(:,2),'g.','MarkerSize',20);
hold off;

SOM

```
[SOM]=SOMTest;
```

Script:

SomTest.m

Scopo:

SOM 1D per “ordinamento topologico” dei caratteri Braille.

Input:

...

Output:

SOM, SOM per il riconoscimento e l'ordinamento dei caratteri Braille.

Note:

...

Esempio:

...

Osservazione

Dati → **Elaborazione dati (calcolo features)** → SOM → Classificazione

Test...

- Osservare l'ordinamento introdotto dei caratteri Braille proposto dalla SOM...
- Modificare le features utilizzate per il riconoscimento dei caratteri (in ComputeImageFeatures.m)...
- Testare la SOM per il riconoscimento dei caratteri, introducendo rumore sui dati...

Es.

```
[SOM]=SOMTest;
```

```
I=double(imread('v.tif'));
for Noise=[0 1 5 10 25 50 100 150 200]
    RecognitionTest(SOM,I+Noise*randn(size(I)));
    pause;
end
```

Note:

This image shows a single sheet of white paper with horizontal blue ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.