EE1390

Prototype methods

EE18BTECH11025 and EE18BTECH11016

Prototype Methods

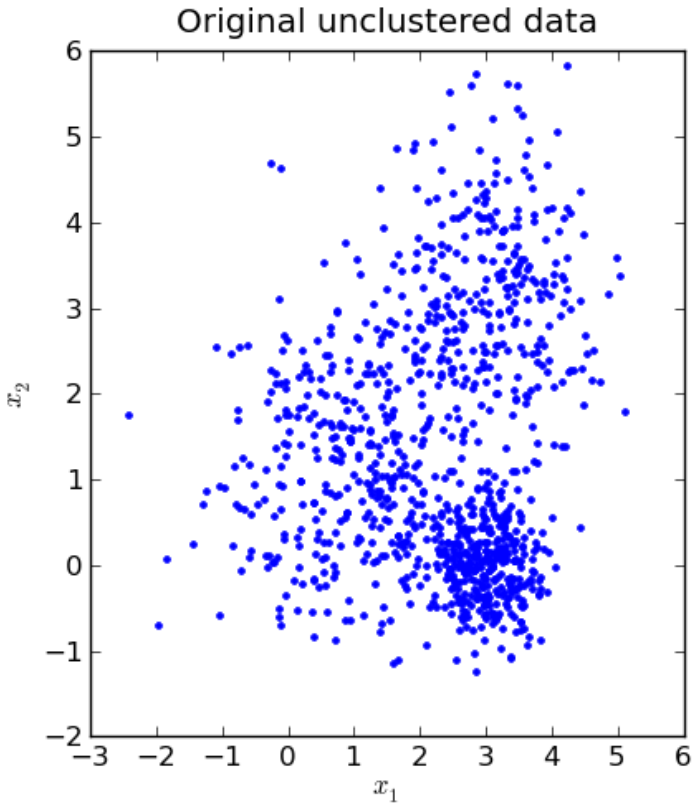
A prototype is an element of the data space that represents a group of elements.
A cluster prototype serves to characterize the cluster, their elements.

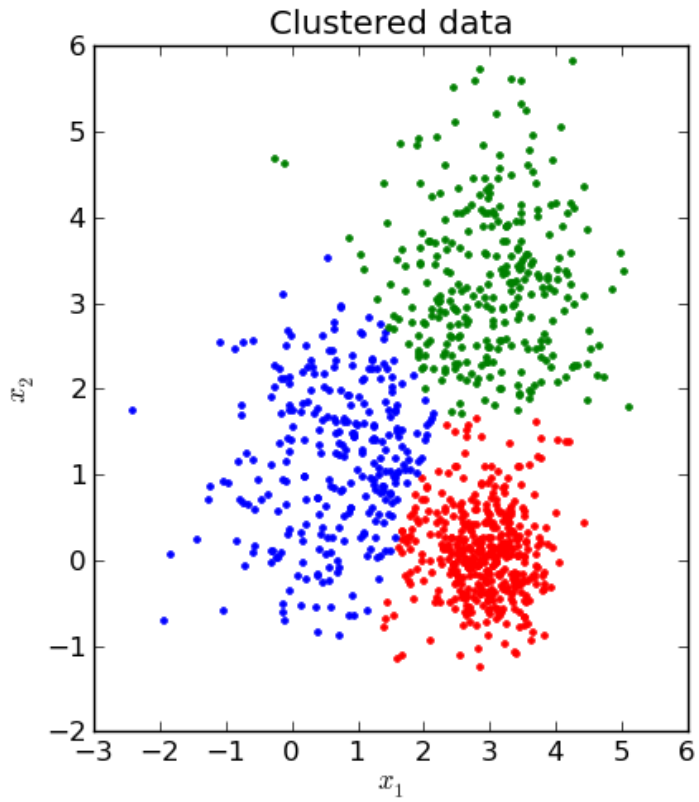
K-means Clustering

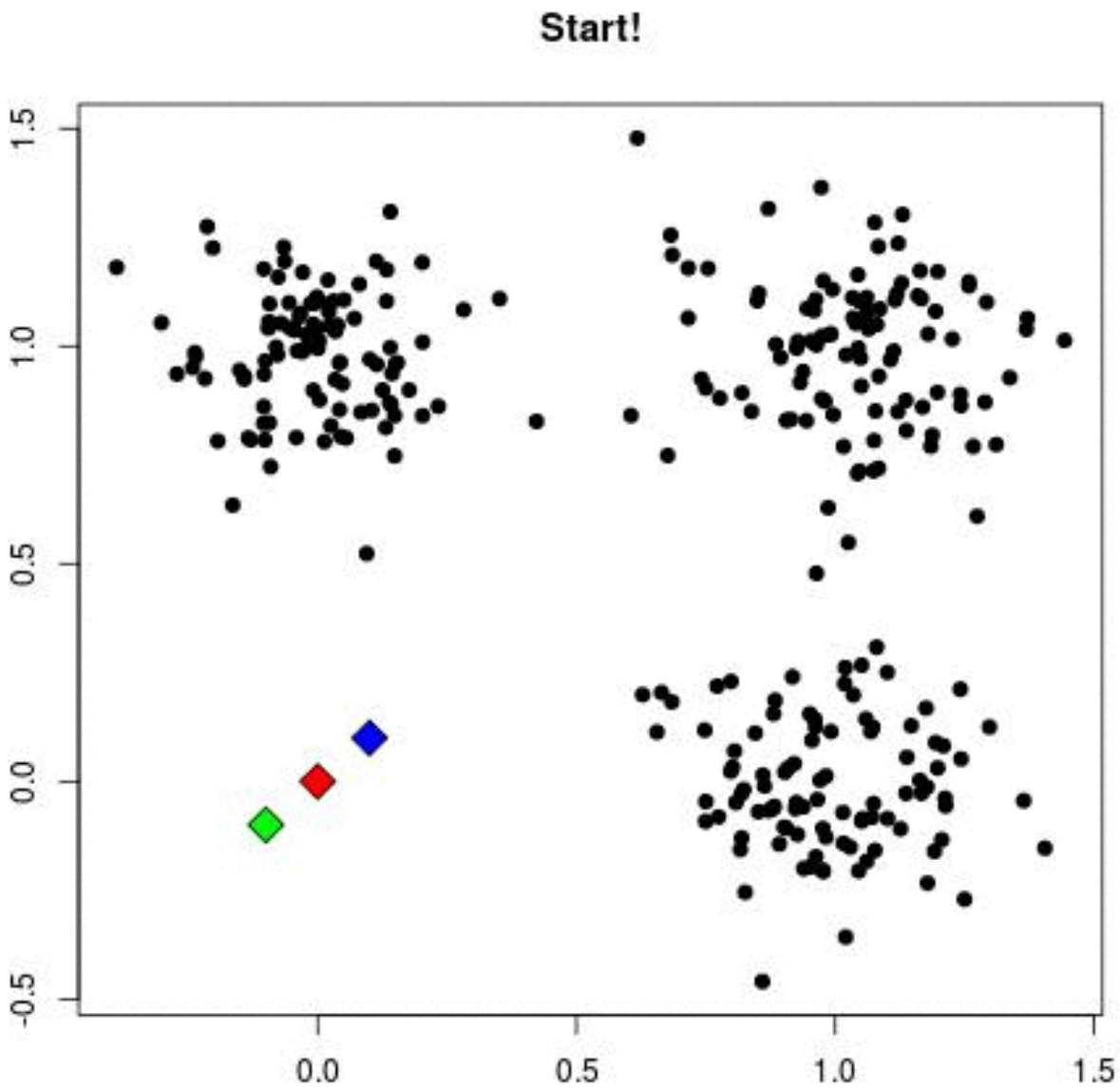
K-means clustering is a method for finding clusters and cluster centers in a set of unlabeled data.
One chooses the desired number of cluster centers, say R, and the K-means procedure iteratively moves the centers to minimize the total within cluster variance.

K-means Clustering

Given an initial set of centers, the K-means algorithm alternates the two steps:
1. for each center we identify the subset of training points (its cluster)that is closer to it than any other center
2. the means of each feature for the data points in each cluster are computed, and this mean vector becomes the new center for that cluster


Original unclustered data
x_1
x_2


Clustered data
x_1
x_2


Start!
1.5
1.0
0.5
0.0
-0.5
0.0
0.5
1.0
1.5

Linear Vector Quantization

Prototypes are placed strategically with respect to the decision boundaries in an ad-hoc way. LVQ is an online algorithm, observations are processed one at a time.
The idea is that the training points attract prototypes of the correct class,and repel other prototypes.
When the iterations settle down, prototypes should be close to the training points in their class.
The learning rate α is decreased to zero with each iteration

Linear Vector Quantization

1. Choose R initial prototypes for each class: $m_1(k), m_2(k), \ldots, m_R(k)$, k= 1,2, . . . , K, for example, by sampling R training points at random from each class.

2. Sample a training point x_i randomly (with replacement), and let (j, k)index the closest prototype $m_j(k)$ to x_i

   (a) If $g_i = k$(i.e., they are in the same class), move the prototype towards the training point:

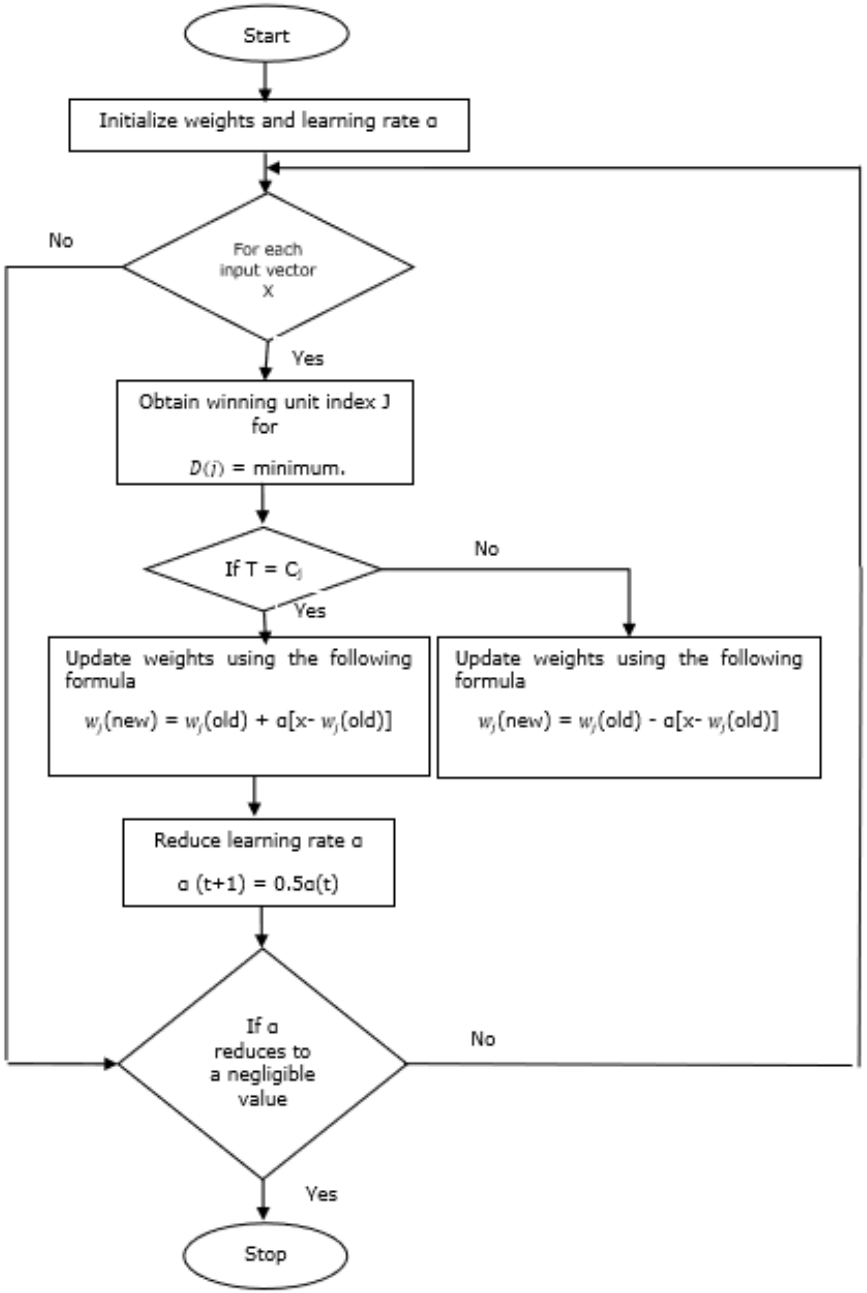
   $$m_j(k) \leftarrow m_j(k) + \alpha(x_i - m_j(k)),$$

   where α is the learning rate.

   (b) If $g_i \neq k$ (i.e., they are in different classes), move the prototype away from the training point:

   $$m_j(k) \leftarrow m_j(k) - \alpha(x_i - m_j(k))$$

3. Repeat step 2, decreasing the learning rate with each iteration towards zero.


Start
Initialize weights and learning rate α
For each input vector X
No
Yes
Obtain winning unit index J for
D(j) = minimum.
If T = Cj
No
Yes
Update weights using the following formula
wj(new) = wj(old) + α[x- wj(old)]
Update weights using the following formula
wj(new) = wj(old) - α[x- wj(old)]
Reduce learning rate α
α (t+1) = 0.5α(t)
If α reduces to a negligible value
No
Yes
Stop

LVQ2

The condition of LVQ2 is formed by window. This window will be based on the following parameters:
x - the current input vector
yc - the reference vector closest to x
yr - the other reference vector, which is next closest to x
dc - the distance from x to yc
dr - the distance from x to yr

The input vector x falls in the window, if:

$$d_c/d_r > 1 - \theta \text{ and } d_r/d_c > 1 + \theta$$

Here, θ is the number of training samples.

Updating can be done with the following formula

$y_c(t+1)=y_c(t)+(t)[x(t) - y_c(t)]$ (belongs to different class)

$y_r(t+1)=y_r(t)+(t)[x(t) - y_r(t)]$ (belongs to same class)

Here α is the learning rate.

LVQ3

In LVQ3, we will take the two closest vectors namely y_c1 and y_c2 and the condition for window is as follows

Min[d_c1/d_c2, d_c2/d_c1]>(1 - θ)(1 + θ)

Here $\theta \approx 0.2$

Updating can be done with the following formula

$y_c1(t+1)=y_c1(t) + \beta(t)[x(t) - y_c1(t)]$ (belongs to different class)

$y_c2(t+1)=y_c2(t) + \beta(t)[x(t) - y_c2(t)]$ (belongs to same class)

Here β is the multiple of the learning rate α and $\beta = m\alpha(t)$ for every $0.1 < m < 0.5$