Clustering

Author: Zhang Xiaozheng Date: 2019/10/24

In unsupervised learning, we don't know the classes of one data set, but we still want to find some relationship. In this case, we can use clustering algorithm. After clustering, the data set may be divided into lots of cluster. But we don't know what common feature these data have. we need to find by ourselves.

So how to do clustering job? I will introduce three algorithm which are *K-means*, *Learn Vector Quantization(LVQ)* and *AGNES*.

K-means

If we have a data set *D* and it's clusters *C*:

$$D = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}\}$$
$$C = \{\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_k}\}$$

The loss function is:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|_2^2$$

where

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

which means it is the central point.

To minimize this loss function, one sample and naive way is to try all possible clustering combinations and see which one is the optimal. But it takes a long time. So we take the greedy strategy to obtain an approximate solution.

First, we choose *k* vectors as initial central points

$$\mu_1 = \mathbf{x_1}, \mu_2 = \mathbf{x_2}, \ldots, \mu_k = \mathbf{x_k}$$
$$\mu = \{\mu_1, \mu_2, \ldots, \mu_k\}$$

Then for every *x* in *D*, we calculate the distance between *x* and every central point.

$$dis(x_i, \mu_i) = \|x_i - \mu_i\|_2$$

Then we put this *x* into the cluster whose central point is nearest to it. After iterate all *x* in *D*, we calculate new central points of every cluster, and then repeat previews work until central points don't change or change a little.

Learn Vector Quantization (LVQ)

Unlike *K-means*, LVQ requires the data set to have labels, so these labels can help LVQ to do clustering job.

First, we have training set *D*. We random choose *q* vectors and let their label to be *t*.

$$p_1 = x_i, p_2 = x_j, \ldots, p_q = x_k$$
$$t_1 = y_i, t_2 = y_j, \ldots, t_q = y_k$$

Then we random choose one *x* in *D*, and find the nearest *p*.

If *p*'s label *t* is same as *x*'s label *y*, we update *p*:

$$p_{new} = p_{old} + \eta(x - p_{old})$$
$$where \ \ \eta \ \ is\, learning\, rate$$

If *p*'s label *t* is different with *x*'s label *y*, we update *p*:

$$p_{new} = p_{old} - \eta(x - p_{old})$$

Intuitively, what these algorithm do is when the nearest *p*'s label is same as *x*'s label, we let this *p* close to this *x*, otherwise, we let this *p* far away from this *x*.

Then repeat previews step, until reach the maximum number of iterations or those vectors *p* don't change or change a little.

Because LVQ randomly choose the data, so even if we can't obtain all the data or even we can't obtain all the data, LVQ can still work.

AGNES

In AGNES algorithm, we suppose every instance in training set as a cluster. Then we merge these cluster continuously until reach the number of clusters we want.

So how to merge clusters? In AGNES, we choose two nearest clusters to merge. Here is a problem, how to measure the distance between two clusters? Usually, there are three ways:

$$d_{min}(C_i, C_j) = min_{x \in C_i, z \in C_j} dist(x, z)$$
$$d_{max}(C_i, C_j) = max_{x \in C_i, z \in C_j} dist(x, z)$$
$$d_{avg}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i} \sum_{z \in C_j} dist(x, z)$$

When we choose different way to measure the distance between two clusters, AGNES algorithm can be called *single-linkage*, *complete-linkage* and *average-linkage* algorithm.

After merge two clusters, we use new clusters and repeat previews step until the number of clusters reach what we want.