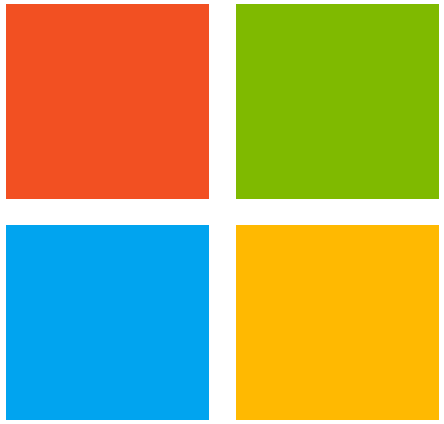


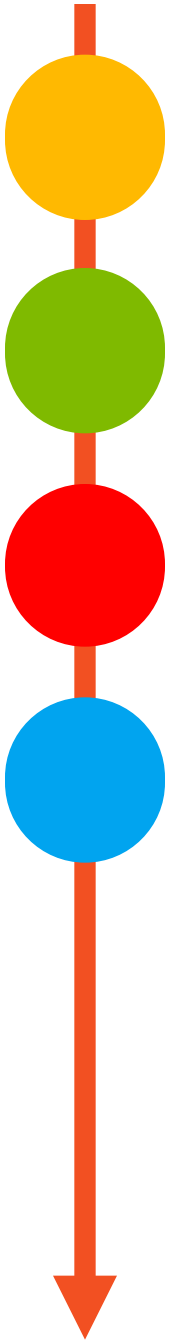
2018 TJMSC Tech. Courses

Machine Learning Algorithm

Yiran Zhuo
Tongji Microsoft Student Club

Oct 28, 2018
Room 516, Ji Shi Building
SSE, Tongji Univ

同济微软俱乐部
Tongji Microsoft Student Club



Introduction of kNN

Introduction of kMeans

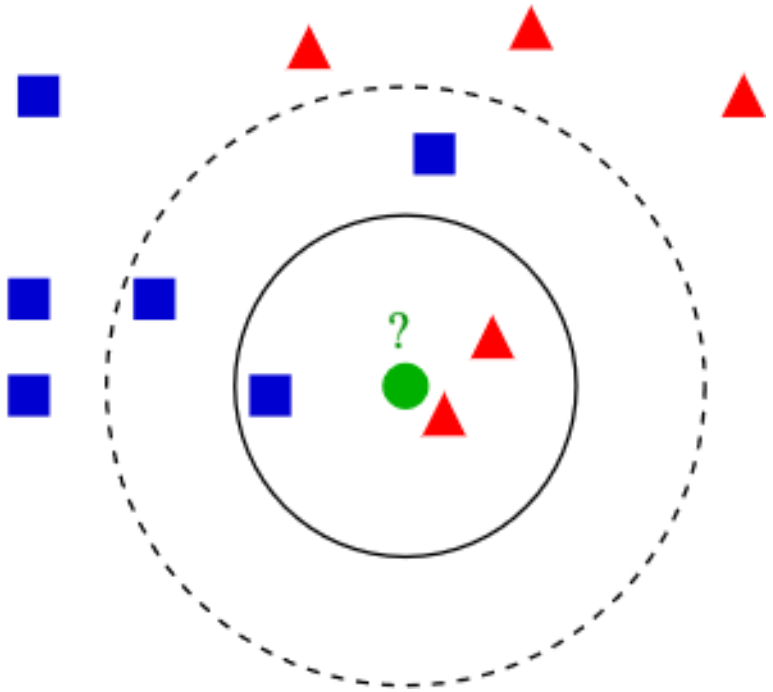
Code of kMeans

Other things

Most materials are referenced from CS451, David Kauchak

A simple supervised learning algorithm

Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).



Pseudocode of kNN

1、Calculating the distance between the point in the known category data set and the current point;
2、Sorted in increasing order of distance;
3、Select k points with the smallest distance from the current point;
4、Determine the frequency of occurrence of the category of the first k points;
5、Returns the category with the highest frequency occurring in the first k points as the predicted classification of the current point

Normalize

$$newValue = (oldValue- min)/(max- min)$$

The general process

1、Collect data: Any method can be used.
2、Prepare data: The values required for distance calculations, preferably structured data formats.
3、Analyze data: Any method can be used.
4、Training algorithm
5、Test algorithm: Calculate the error rate.
6、Using the algorithm: First, the sample data and the structured output result need to be input, and then the k-nearest neighbor algorithm is executed to determine which classification the input data belongs to, and finally the application performs subsequent processing on the calculated classification.

When to Use It

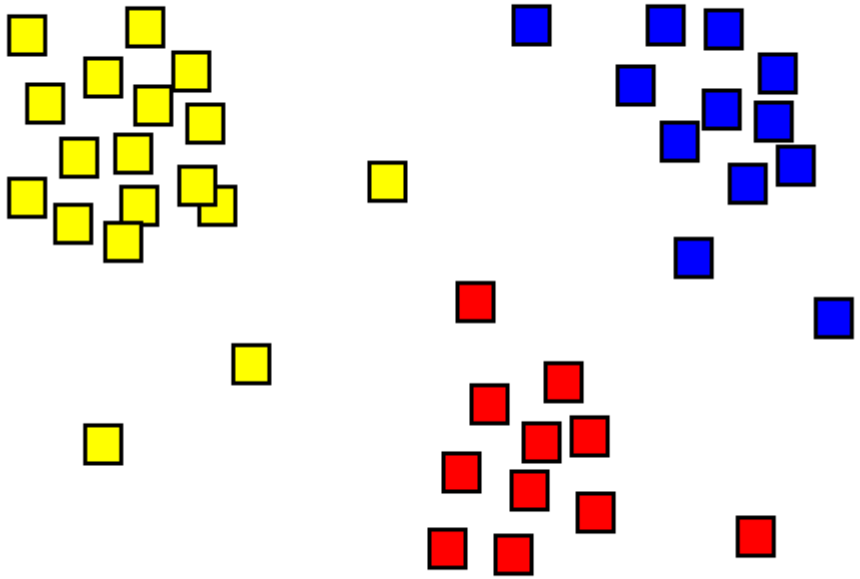
Advantages: high precision, insensitive to outliers, no data input assumptions.

Disadvantages: high computational complexity and high space complexity

Applicable data type: numeric and nominal

What is clustering

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters)



Pseudocode of kMeans

Select K points as the initial centroid /*usually random selection*/
repeat
{
Assign each point to the nearest centroid to form K clusters
Recalculate the centroid of each cluster
}
until
(The center of mass does not change or reaches the maximum number of iterations)

Example of kMeans

D={2,8,3,9}，　which means D1=2 , D2=8 , D3=3 , D4=9
K = 2

Example of kMeans

choose a centroid /*random*/

In this example, we choose 2、3

Example of kMeans

Circle1

Example of kMeans

Circle1

Example of kMeans

Circle2

Example of kMeans

同济微软俱乐部
Tongji Microsoft Student Club

Circle2

Example of kMeans

同济微软俱乐部
Tongji Microsoft Student Club

Circle3

Centroids are not changed

Result:

{2,3} {8,9}

Principle

Why is the update of centroid taking mean values of all particles? That is, why should the centroid unchanged be the algorithm termination condition?

Principle

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2$$

$$\begin{aligned}
\frac{\partial}{\partial c_k}\mathrm{SSE} &= \frac{\partial}{\partial c_k} \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2 \\
&= \sum_{i=1}^{K} \sum_{x \in C_i} \frac{\partial}{\partial c_k} (c_i - x)^2 \\
&= \sum_{x \in C_k} 2\,(c_k - x_k) = 0
\end{aligned}$$

$$\sum_{x \in C_k} 2\,(c_k - x_k) = 0 \Rightarrow m_k c_k = \sum_{x \in C_k} x_k \Rightarrow c_k = \frac{1}{m_k} \sum_{x \in C_k} x_k$$

The general process

1、Collect data: Use any method.
2、Prepare data: You need numerical data to calculate the distance. You can also map the nominal data to binary data and use it for distance calculation.
3、Analyze data: Use any method.
4、Training algorithm:
5、Test algorithm: Apply clustering algorithm and observation results. Quantized error metrics such as sum of squared errors (described later) can be used to evaluate the results of the algorithm.
6、Use algorithm: Can be used for any application you want. Typically, the cluster centroid can make decisions based on the data of the entire cluster.

When to Use it

Advantages: easy to implement.
Disadvantages: It may converge to a local minimum and converge slowly on large data sets.
Applicable data type: numeric data.

Talk is cheap


```python
from numpy import *

def distEclud(vecA, vecB):
    return sqrt(sum(power(vecA - vecB, 2))) #计算A、B间的欧式距离

def randCent(dataSet, k): #随机生成随机的质心，约束条件为大于数据中出现的最小值，小于最大值
    n = shape(dataSet)[1]
    centroids = mat(zeros((k,n)))
    for j in range(n):
        minJ = min(dataSet[:,j])  #获取第j维数据的最小值
        rangeJ = float(max(dataSet[:,j]) - minJ) #获取数据范围
        centroids[:,j] = mat(minJ + rangeJ * random.rand(k,1)) #生成随机数
    return centroids

```

Talk is cheap


```python
def kMeans(dataSet, k, distMeas=distEclud, createCent=randCent):#四个参数，数据集、簇数、求距离函数、生成质心函数
    m = shape(dataSet)[0] #获取数据点总量m
    clusterAssment = mat(zeros((m,2))) #用于存储分配结果，含两列，第一列存所属簇索引值，第二列存到质心距离
    centroids = createCent(dataSet, k) #随机选择质心
    clusterChanged = True
    while clusterChanged: #终止条件：质心没有发生变化
        clusterChanged = False
        for i in range(m): #对每个数据点，将它分配给距离最近的质心
            minDist = inf; minIndex = -1
            for j in range(k): #遍历k个簇的质心，求其距离
                distJI = distMeas(centroids[j,:],dataSet[i,:])
                if distJI < minDist:
                    minDist = distJI; minIndex = j
            if clusterAssment[i,0] != minIndex:
                clusterChanged = True
            clusterAssment[i,:] = minIndex,minDist**2
        print(centroids)
        for cent in range(k): #对每个数据点计算完毕后，重新计算每簇的质心
            ptsInClust = dataSet[nonzero(clusterAssment[:,0].A==cent)[0]]
            centroids[cent,:] = mean(ptsInClust, axis=0)
    return centroids, clusterAssment
```

Performance evaluation and post-processing

同济微软俱乐部
Tongji Microsoft Student Club

The clustering effect is evaluated by SSE.

Post-processing (optimizing the result while keeping the k value constant): The cluster with the largest SSE is divided into two clusters.
Filter out the points in the cluster, then use kMeans whose k=2 to divide it into two cluster; then find another cluster to merge with one of the two clusters, calculating possible merging situation and find the smallest of the total SSE values.

Optimization – biKmeans

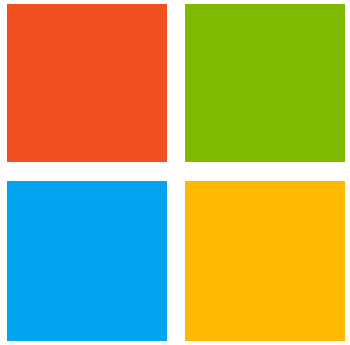
Purpose: overcome converge to a local minimum

First, treat all points as a cluster and then split the cluster into two. Then select the cluster that minimizes the cluster cost function (that is, the sum of squared errors) into two clusters. The algorithm will be done when the number of clusters is equal to the number k given by the user.

Pseudocode of biKmeans


```
Treat all the points as a cluster
When the cluster is smaller than the number k
        For each cluster
                Calculate the total error
                Perform K-means clustering on this cluster
with a k value of 2
                Calculate the total error after dividing the
cluster into two clusters
        Select the cluster that minimizes the error to
divide
```

```python
def biKmeans(dataSet, k, distMeas=distEclud):
    m = shape(dataSet)[0] #获取数据点总数
    clusterAssment = mat(zeros((m,2))) #同Kmeans
    centroid0 = mean(dataSet, axis=0).tolist()[0] #求所有点的均值作为初始质心
    centList =[centroid0] #建立一个保存质心的列表
    for j in range(m): #计算初始SSE
        clusterAssment[j,1] = distMeas(mat(centroid0), dataSet[j,:])**2
    while (len(centList) < k): #终止条件：分出k个簇
        lowestSSE = inf #初始将最小SSE设为无穷大
        for i in range(len(centList)): #对当前所有簇
            ptsInCurrCluster = dataSet[nonzero(clusterAssment[:,0].A==i)[0],:] #取第i个簇中的所有数据点
            centroidMat, splitClustAss = kMeans(ptsInCurrCluster, 2, distMeas) #对第i个簇进行划分
            sseSplit = sum(splitClustAss[:,1]) #计算这种划分产生两组簇的SSE
            sseNotSplit = sum(clusterAssment[nonzero(clusterAssment[:,0].A!=i)[0],1]) #取没有划分的簇的SSE
            print("sseSplit, and notSplit: ",sseSplit,sseNotSplit)
            if (sseSplit + sseNotSplit) < lowestSSE: #考察其是否小于最小SSE，如果是，则更新
                bestCentToSplit = i
                bestNewCents = centroidMat
                bestClustAss = splitClustAss.copy()
                lowestSSE = sseSplit + sseNotSplit
        bestClustAss[nonzero(bestClustAss[:,0].A == 1)[0],0] = len(centList) #调用Kmeans分出的两个簇为0，1，将其更新为相应的值
        bestClustAss[nonzero(bestClustAss[:,0].A == 0)[0],0] = bestCentToSplit
        print('the bestCentToSplit is: ',bestCentToSplit)
        print('the len of bestClustAss is: ', len(bestClustAss))
        centList[bestCentToSplit] = bestNewCents[0,:].tolist()[0] #更新质点
        centList.append(bestNewCents[1,:].tolist()[0])
        clusterAssment[nonzero(clusterAssment[:,0].A == bestCentToSplit)[0],:]= bestClustAss #更新簇划分编号、SSE
    return mat(centList), clusterAssment
```



Microsoft

同济微软俱乐部
Tongji Microsoft Student Club

微软学生俱乐部

扫一扫二维码，加入该群。

关注TJMSC微信公众号