机器学习——KNN

对于k近邻算法，最简单的理解就是：它并不是一种严格的学习算法，不需要经过显式的学习过程，只是通过比对测试数据与训练数据（说是训练数据，其实就是已有的数据）的距离，根据远近来得到最有可能的结果。

流程：

对于KNN算法的流程一般如下：
1、计算测试数据与我们已有的数据的点之间的距离

2、根据得到的距离进行排序

3、选定与测试点距离最小的前k个点

4、确定前k个点所在类别出现的频率

5、返回最高频率的类别作为预测分类

实战（海伦约会例子）：

```
import numpy as np
import operator
"""
inX是测试集
dataSet是训练集
"""
def classif0(inX,dataSet,labels,k):
    #行数
    m=dataSet.shape[0]
    #扩展单行的测试集
    diffMat=np.tile(inX,(m,1))-dataSet
    sqMat=diffMat**2
    sqDisance=sqMat.sum(axis=1)
    distance=sqDisance*0.5
    sort_distance=distance.argsort()
    classCount={}
    for i in range(k):
        votelabel=labels[sort_distance[i]]
        classCount[votelabel]=classCount.get(votelabel,0)+1

sortClassCount=sorted(classCount.items(),key=operator.itemgetter(1),reverse=True
)
    return sortClassCount[0][0]

def file2matrix(filename):
    fr=open(filename)
    lines=fr.readlines()
    number=len(lines)
    returnMat =np.zeros((number,3))
    classLabelVector=[]
    index=0
    for line in lines:
        line=line.strip().split('\t')
```

```
        returnMat[index,:]=line[0:3]
        if line[-1]=='didntLike':
            classLabelVector.append(1)
        elif line[-1]=='smallDoses':
            classLabelVector.append(2)
        elif line[-1]=='largeDoses':
            classLabelVector.append(3)
        index+=1
    return returnMat,classLabelVector

def autoNorm(dataSet):
    minVal=dataSet.min(0)
    maxVal=dataSet.max(0)
    ranges=maxVal-minVal
    normDataSet=np.zeros(np.shape(dataSet))
    m=dataSet.shape[0]
    normDataSet=dataSet-np.tile(minVal,(m,1))
    normDataSet=normDataSet/np.tile(ranges,(m,1))
    return normDataSet,ranges,minVal

def datingClassTest():
    filename="datingTestSet.txt"
    datingDataMat,datingLabels=file2matrix(filename)
    hoRatio=0.10
    normMat,ranges,minVal=autoNorm(datingDataMat)
    m=normMat.shape[0]
    num_test=int(m*hoRatio)
    errorCount=0.0
    for i in range(num_test):

classResult=classif0(normMat[i,:],normMat[num_test:m,:],datingLabels[num_test:m]
,4)
        print("分类结果：%d\t真是类别：%d" % (classResult,datingLabels[i]))
        if classResult!=datingLabels[i]:
            errorCount+=1.0
    print("错误率：%f%%" %(errorCount/float(num_test)*100))

datingClassTest()
```

得到的结果：

```
分类结果：1  真是类别：1
分类结果：3  真是类别：3
分类结果：3  真是类别：3
分类结果：2  真是类别：2
分类结果：2  真是类别：1
分类结果：1  真是类别：1
错误率：4.000000%
```

总的来说就是计算每个对应项对的欧式距离来比对，接下来我们使用sklearn的函数来解决手写数字识别：

```
import numpy as np
import operator
from os import listdir
from sklearn.neighbors import KNeighborsClassifier as KNN
def img2vector(filename):
```

```
    returnVect=np.zeros((1,1024))
    fr=open(filename)
    for i in range(32):
        linestr=fr.readline()
        for j in range(32):
            returnVect[0,32*i+j]=int(linestr[j])
    return returnVect

def handwritingClassTest():
    hwLabels=[]
    trainingFileList=listdir('trainingDigits')
    m=len(trainingFileList)
    trainingMat=np.zeros((m,1024))
    for i in range(m):
        filenameStr=trainingFileList[i]
        classNumber=int(filenameStr.split('_')[0])
        hwLabels.append(classNumber)
        trainingMat[i,:]=img2vector('trainingDigits/%s' %(filenameStr))
    clf=KNN(n_neighbors=3,algorithm='auto')
    clf.fit(trainingMat,hwLabels)
    testFilelist=listdir('testDigits')
    errorCount=0.0
    mTest=len(testFilelist)
    for i in range(mTest):
        filenameStr=testFilelist[i]
        classNumber=int(filenameStr.split('_')[0])
        vectorUnderTest=img2vector('testDigits/%s' %(filenameStr))
        classifierResult=clf.predict(vectorUnderTest)
        print("分类结果是%d\t真实结果是%d" %(classifierResult,classNumber))
        if(classifierResult!=classNumber):
            errorCount+=1.0
    print("总共错了%d个数据\n错误率是%f%%" %(errorCount,errorCount/mTest*100))

handwritingClassTest()
```

```
分类结果是9  真实结果是9
分类结果是9  真实结果是9
分类结果是9  真实结果是9
分类结果是9  真实结果是9
分类结果是9  真实结果是9
分类结果是9  真实结果是9
总共错了12个数据
错误率是1.268499%
```

当然你可以修改函数的参数来得到更好的结果。

总结:

KNN是一个很好理解的模型，训练时间复杂度有O（n），而且对异常值不敏感，但是它的空间复杂性太高，计算量有点大，同时无法给出数据的内在含义。