西安电子科技大学

安全前沿讨论班（I） 课程实验报告

实验名称 K近邻算法的实现

学　院 网络与信息安全学院　　班 级 1618019

姓　名 曹寅峰　　学 号 16020610025

实验日期 2019 年 5 月 21 日

指导教师评语： 指导教师： 年 月 日
实验报告内容基本要求及参考格式
一、实验目的 应用已经掌握的Python程序设计语言的相关知识，大致了解如何用Python实现K近邻分类算法，并创建一个自己的KNN（K-Nearest Neighbor）分类器。对所设计的问题进行编程实现，从而达到对所学知识的练习和巩固。

二、实验环境

Python3.5 jupyter

三、实验基本原理及步骤（或方案设计及理论计算）

1. 计算两点间距离

```
# 写一个函数计算两个点之间的距离
def distance(p1,p2):
    """p1 and p2 are two ndarray;
    Find the distance between points p1 and p2, then return the distance;
    """
    return np.sqrt(np.sum(np.square(p1-p2)))  #对元素运算
```

根据两点间距离公式，直接返回即可。注意这样写的运算单位是单个坐标，很方便。

2. 找出最高频率项

```
def majority_vote(votes):
    """votes is a list;
    Return the most common element in votes"""
    votedic = {}
    mostvotes = None
    for vote in votes:
        if vote not in votedic:
            count = votes.count(vote)#创建计数字典
            votedic[vote] = count
            if count > votedic.get(mostvotes, 0):#如果大于mostvotes 则更新最大值
                mostvotes = vote
    #print(votedic)
    return mostvotes
```

这里我们创建一个频率字典，键值为类别-票数，同时利用 count 找出票数的最大值对应的键，最后返回这个键

3. 另一种高效方法

```
import scipy.stats as ss
def majority_vote_short(votes):
    """
    Return the most common element in votes
    """
    return ss.mode(votes)[0][0]
```

利用 ss 库可以高效找出频率最大的项，其中第一个[0]为频率最大的项的列表形式，第二个[0]为列表中的这一项的元素形式。

4. 找到临近邻居

```
def find_nearest_neighbors(p,points,k=5):
    """
    p: a point;
    points: an array of points;
    k: an integer, default value is 5;
    Find the k nearest neighbors of point p and return their indices."""
    distances = np.zeros(points.shape[0])
# your code is here
    for i in range(len(distances)):
        distances[i]=distance(points[i],p)
    ind = np.argsort(distances)
    return ind[:k]
```

这里我们先计算 p 和 points 中的每个元素的距离，并进行升序排序，返回前 K 个最邻近邻居的**索引号**

5. Knn 函数

```
def knn_predict(p,points,outcomes,k=5):
    """
    - p is the new point we'd like to classify
    - points are our existing data or our training data
    - outcomes are the classes to which these k points belong to."""
    # find k nearest neighbors
    ind = find_nearest_neighbors(p,points,k)
    return(majority_vote_short(outcomes[ind]))
    # predict the class of p based on majority vote
```

利用前面产生的序列，对应 outcomes 的分类结果，再从中选出频率最高分类结果，完成预测

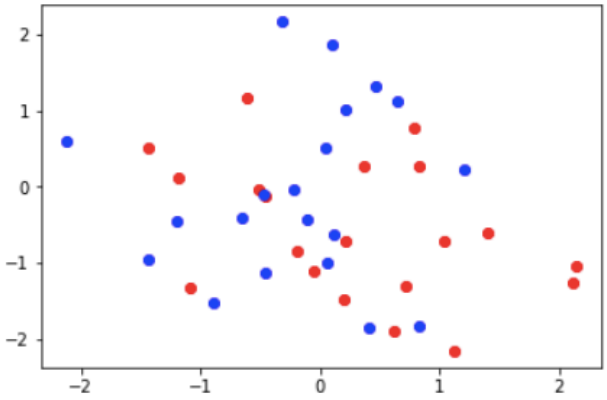
6. 产生模拟数据

```
ss.norm(0,1).rvs((5,2))
```

```
array([[-0.08044652, -1.03691774],
       [ 0.10049695,  1.8971495 ],
       [-1.58641434, -0.80091185],
       [-1.85132799, -0.59819436],
       [-0.96697467,  1.73570409]])
```

这段代码的意思是产生 0+-1 区间的 5 个 2 维元素。

```
In [30]: # we have n red points and n blue points corresponding to the two diffe
plt.figure()
plt.plot(points[:n,0],points[:n,1],"ro")
plt.plot(points[n:,0],points[n:,1],"bo")
plt.savefig("bivariatedata.pdf")
plt.show()
```



学习怎样预测网格;学习enumerate的使用； 学习怎样使用

随后我们产生了 50 个二维随机点，并随机分为 2 类，并将图像存储到 pdf 中。

7. 学习怎样预测网格; 学习 enumerate 的使用； 学习怎样使用 numpy 中的 meshgrid.

```
# plot a prediction grid
# This means that once we've observed our data,
# we can examine some part of the predictor space and compute the class prediction for each
# point in the grid using the knn classifier.
# so instead of finding out how our classifier might classify a given point,
# we can ask how it classifies all points that belong to a rectangular region of the predictor sp


# meshgrid takes in two or more coordinate vectors, say one vector containing the x values,
# the other containing the y values of interest;
# it returns matrices, the first containing the x values for each grid point and the
# secoond containing the y values for each grid point.
def make_predictin_grid(predictors,outcomes,limits,h,k):
    """Classify each point on the prediction grid."""
    (x_min, x_max, y_min, y_max) = limits
    xs = np.arange(x_min, x_max, h)
    ys = np.arange(y_min, y_max, h)
    xx, yy = np.meshgrid(xs, ys)

    prediction_grid = np.zeros(xx.shape,dtype=int)
    for i,x in enumerate(xs):
        for j,y in enumerate(ys):
            p = np.array([x,y])
            prediction_grid[j,i] = knn_predict(p,predictors,outcomes,k)
    return (xx,yy,prediction_grid)
```

这里是对空间中每个点（网格）进行 knn 预测。其中 enumerate() 函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据和数据下标,画出预测网格。Meshgrid 用于创建指定个数的**坐标矩阵。最后利用这些信息，预测结果保存到 prediction_grid 中。**

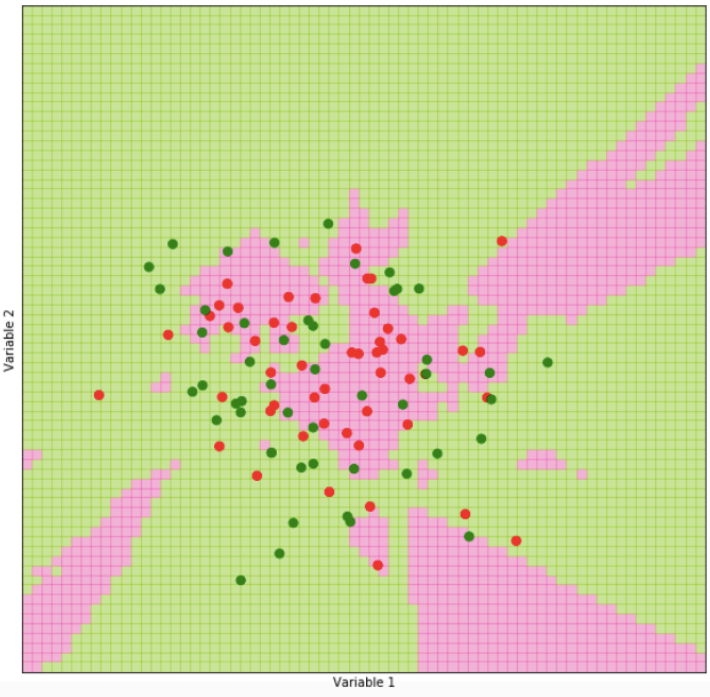
8. 学习怎样画出预测网格

```
# a function used to plot the prediction grid
def plot_prediction_grid (xx, yy, prediction_grid, filename):
    """ Plot KNN predictions for every point on the grid."""
    from matplotlib.colors import ListedColormap
    background_colormap = ListedColormap (["hotpink","lightskyblue", "yellowgreen"])
    observation_colormap = ListedColormap (["red","blue","green"])

    plt.figure(figsize =(10,10))
    plt.pcolormesh(xx, yy, prediction_grid, cmap = background_colormap, alpha = 0.5)
    plt.scatter(predictors[:,0], predictors [:,1], c = outcomes, cmap = observation_colormap, s = 50)

    plt.xlabel('Variable 1'); plt.ylabel('Variable 2')
    plt.xticks(()); plt.yticks(())
    plt.xlim (np.min(xx), np.max(xx))
    plt.ylim (np.min(yy), np.max(yy))
    plt.savefig(filename)
```

其中主要代码是

plt.pcolormesh(xx, yy, prediction_grid, cmap = background_colormap, alpha = 0.5)

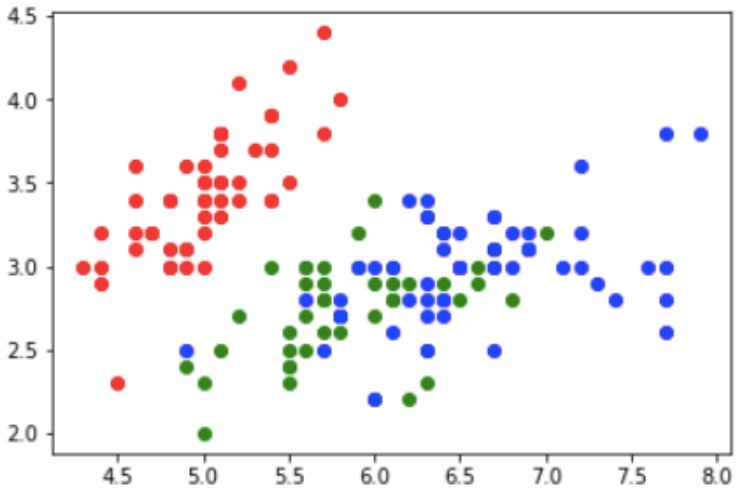
将之前的坐标矩阵，预测结果矩阵，和背景颜色绘出图像。



9. 在真实数据集上应用自己创建的 KNN 分类器

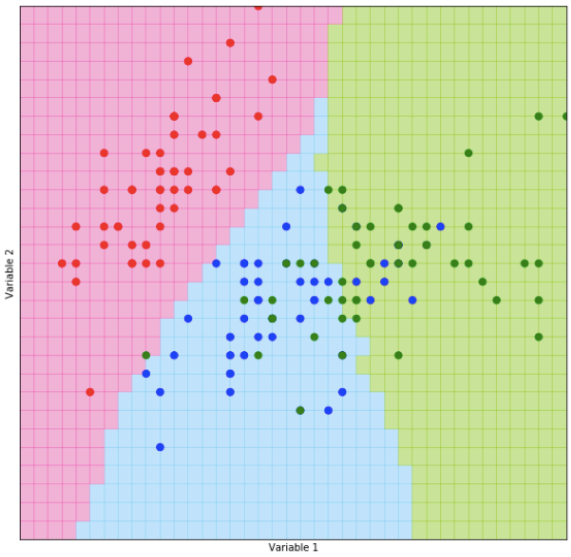
```
from sklearn import datasets
iris = datasets.load_iris()
```

```
predictors = iris.data[:,0:2] # all of the rows but only columns 0 and 1
outcomes = iris.target

plt.plot(predictors[outcomes==0][:,0],predictors[outcomes==0][:,1],"ro")
plt.plot(predictors[outcomes==1][:,0],predictors[outcomes==1][:,1],"go")
plt.plot(predictors[outcomes==2][:,0],predictors[outcomes==2][:,1],"bo")
plt.savefig("iris.pdf")
```



从 iris 中选出二维向量，并且画出他们的坐标，分类。

```
k=50
filename = "iris_grid.pdf"
limits = (4,8,1.5,4.5)
h=0.1
(xx,yy,prediction_grid) = make_predictin_grid(predictors,outcomes,limits,h,k)
plot_prediction_grid(xx,yy,prediction_grid,filename)
```



再利用同样的方法，画出预测网络

```
]: (150,)
```

```
]: sk_predictions == my_predictions
```

```
]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True, False,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True, False,  True,  True,  True,
         True,  True,  True,  True, False,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
        False,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True,  True,  True,  True,  True,  True,  True,  True,  True,
         True, False,  True,  True,  True, False])
```

这里可以看到，我们的预测结果和官方预测结果稍有不同

```
]: print(100*np.mean(sk_predictions == my_predictions))
   # our predictions and the scikit predictions agree 96% of the time

   96.0
```

```
]: print(100*np.mean(sk_predictions == outcomes))

   83.33333333333334
```

```
]: print(100*np.mean(my_predictions == outcomes))

   84.66666666666667
```

我们的预测结果正确概率是 96%

官方均方误差为 83.3%

而我们的均方误差为 84.7%，稍大一些

四、实验结果分析及回答问题（或测试环境及测试结果）

```
]: p1 = np.array([1,1])
   p2 = np.array([4,4])
   print(distance(p1,p2))
   p1 = np.array([0,3])
   p2 = np.array([4,0])
   print(distance(p1,p2))

   4.242640687119285
   5.0
```

两点间距离计算正确

```
# test your function
votes = [1,2,3,1,2,3,1,2,3,3,3,3]
print(majority_vote(votes))
```

```
3
```

最高频率项也正确

```
points = np.array([[1,1],[1,2],[1,3],[2,1],[2,2],[2,3],[3,1],[3,2],[3,3]])
p = np.array([2.5,2])

ind = find_nearest_neighbors(p,points,2)
print(points[ind])
```

```
[[2 2]
 [3 2]]
```

找出了最邻近的点

```
In [25]: outcomes = np.array([0,0,0,0,1,1,1,1,1])
         p = np.array([2.5,2.7])
         knn_predict(p,points,outcomes,k=2)
```

```
Out[25]: 1
```

预测结果为 1

总结：

通过这次实验，了解了 knn 分类器的原理，以及 numpy 的编程技巧，mpplot 画图的技巧。