

華東理工大學

模式识别大作业

题 目	基于生物力学特征对骨科患者分类
学 院	信息科学与工程
专 业	控制科学与工程
组 员	陈世宽
指导教师	赵海涛

完成日期： 2019 年 12 月 8 日

模式识别作业报告——基于生物力学特征对骨科患者分类

组员：陈世宽

由于各种放假的原因，本学期的模式识别课过得很快，还没什么感觉就已经考完试了。虽然上课的课时很少，但赵海涛老师教的东西还是挺多的。虽然内容很杂，但每个算法都是很精华的。由于上课缺乏编程的实战，本作业旨在通过对 kaggle 上的数据集进行一次机器学习算法的实战，以此巩固对模式识别的认知，也算是一种基于理论的实践吧。

1 数据集简介



本任务为分类任务，旨在通过已知的病人的生物力学特征来对病人患病进行分类。本数据集有两个 csv 格式的文档，虽然不同但也是相关的。column_3C_weka.csv 文档具有三个标签，分别是 Normal (100 人), Disk Hernia (60 人) 和 Spondylolisthesis (150 人)。而 column_2C_weka.csv 文档是将后两种得病的归到一起，从而只具有两种标签，Normal (100 人) 和 Abnormal (210 人)。而这两个文档的生物力学特征都是一样的，具有如下 6 种特征，分别为 pelvic incidence, pelvic tilt, lumbar lordosis angle, sacral slope, pelvic radius 和 grade of spondylolisthesis。（由于这些名词过于专业，在此就不翻译了）。本次作业采用后一种数据，用 KNN 算法来进行简单的二分类任务。

2 实验步骤

2.1 数据的读入和查看

要进行 KNN 算法的编程，首先要将数据读入。本次实验利用 python 语言来进行。选择 python 的原因是 python 有很多现成的包可以进行机器学习算法的处理。Pandas 是数据处理常用的包。关键代码如下：

```
import pandas as pd
data = pd.read_csv('../input/column_2C_weka.csv')
data.head()#默认为5
```

上述数据的最后一行可以查看读入数据集的前五行。结果如图 1 所示。

	pelvic_incidence	pelvic_tilt_numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class
0	63.027818	22.552586	39.609117	40.475232	98.672917	-0.254400	Abnormal
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	Abnormal
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	Abnormal
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	Abnormal
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	Abnormal

图 1 读入数据的前五行

2.2 数据的可视化

对一个数据集进行分类或者回归任务之前，往往要对数据进行可视化操作，越是复杂的数据集越是应该如此。对数据可视化可以使数据的特征更加明显，也能看出数据的分布状态，这是非常重要的一步。对于本次实验，宜采用散布矩阵（scatter matrix）来可视化。散布矩阵是一个方阵的形式，对角线上是各个元素的分布（本实验即 6 个生物力学特征），横轴为值，纵轴为频次。非对角线上的图是各元素之间的关联度。关键代码如下：

```
import matplotlib.pyplot as plt
color_list = ['red' if i=='Abnormal' else 'green' for i in data.loc[:, 'class']]
pd.plotting.scatter_matrix(data.loc[:, data.columns != 'class'],
                           c=color_list,
                           figsize= [15,15],
                           diagonal='hist',
                           alpha=0.5,
                           s = 200,
                           marker = '*',
                           edgecolor= "black")
plt.show()
```

散布矩阵的结果如图 2 所示。

2.3 用 KNN 进行分类

2.3.1 KNN 简介

k 近邻法（k-nearest neighbor, kNN）是一种基本分类与回归方法，其基本做法是：给定测试实例，基于某种距离度量找出训练集中与其最靠近的 k 个实例点，然后基于这 k 个最近邻的信息来进行预测。距离的计算常采用欧氏距离计算。

通常，在分类任务中可使用“投票法”，即选择这 k 个实例中出现最多的标记类别作为预测结果；在回归任务中可使用“平均法”，即将这 k 个实例的实值输出标记的平均值作为预测结果；还可基于距离远近进行加权平均或加权投票，距离越近的实例权重越大。

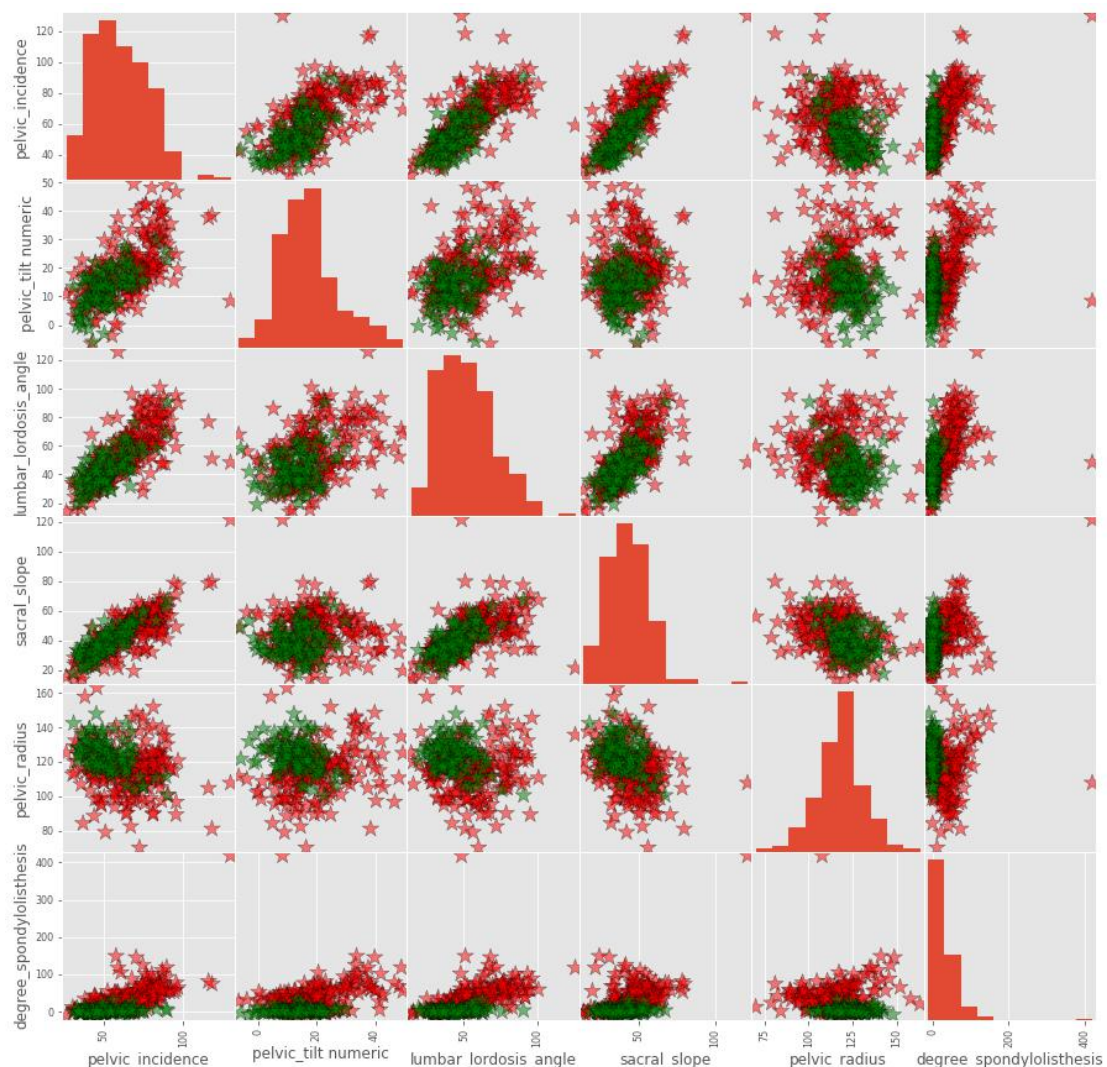


图 2 数据集散布矩阵结果

k 近邻法不具有显式的学习过程，事实上，它是懒惰学习（lazy learning）的著名代表，此类学习技术在训练阶段仅仅是把样本保存起来，训练时间开销为零，

待收到测试样本后再进行处理。

k 值的选择会很大程度上的影响算法的结果。通常，较小的 k 值预测的结果对近邻点比较敏感，当选择的近邻点恰好是噪声，则预测误差就增大。 k 值的减小意味着整体模型变得更复杂，容易产生过拟合。如果是较大的 k 值时，与输入样本较远的（不相似的）的训练样本也会对结果产生影响。 k 值的增大意味着整体模型变得简单。


2.3.2 KNN 的使用

下面介绍本次实验 `knn` 的使用。`Sklearn` 此部分主要应用的包，它是涵盖了主要的机器学习算法以及常用的数据处理的函数和类。首先先对数据集划分，按 7:3 的比例随机分成训练集和测试集。再利用 `sklearn` 中的 `KNeighborsClassifier` 函数进行训练集的拟合，最后对测试集进行分类，以便查看模型的泛化性能。

关键代码如下：

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,random_state = 1)
knn = KNeighborsClassifier(n_neighbors = 3)
x,y = data.loc[:,data.columns != 'class'], data.loc[:, 'class']
knn.fit(x_train,y_train)
prediction = knn.predict(x_test)
print('With KNN (K=3) accuracy is: ',knn.score(x_test,y_test))
```

结果如图 3 所示。



```
With KNN (K=3) accuracy is: 0.8602150537634409
```

图 3 $k=3$ 时的结果

2.3.3 超参数的选择

虽然准确率为 0.86，看上去不错，但不知道是不是真的是最好的。因为 $k=3$ 是我随便取的。

经过上小节对 `knn` 的介绍，`knn` 实际上是一个无参的算法，但也有一个超参数。 k 实际上是本次实验的超参数，它的值对 `knn` 的分类效果起了很大的作用，因此要选择一个最合适的 k 值。将一定范围内的 k 值遍历，取效果最好的作为 k 的最终取值。

关键代码如下：

```
neig = np.arange(1, 25)
train_accuracy = []
test_accuracy = []
for i, k in enumerate(neig):

    knn = KNeighborsClassifier(n_neighbors=k)
    # Fit with knn
    knn.fit(x_train,y_train)
    #train accuracy
    train_accuracy.append(knn.score(x_train, y_train))
    # test accuracy
    test_accuracy.append(knn.score(x_test, y_test))
plt.figure(figsize=[13,8])
plt.plot(neig, test_accuracy, label = 'Testing Accuracy')
plt.plot(neig, train_accuracy, label = 'Training Accuracy')
plt.legend()
plt.title('-value VS Accuracy')
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.xticks(neig)
plt.savefig('graph.png')
plt.show()
print("Best accuracy is {} with K =
{}".format(np.max(test_accuracy),1+test_accuracy.index(np.max(test_accuracy))))
```

结果如图 4 所示。

2.3.4 结果分析

可以看到， $k=1$ 时，训练集的结果很好，但测试集结果很差。这是因为 k 值小的时候，模型的复杂度比较高，造成了过拟合现象。而当 k 值很大时，可以理解为测试点周围的 k 个最近的点的种类变多，也会导致准确率的下降。所以， k 的值不能太大也不能太小。在本次实验中， k 的最佳值为 18，此时的准确率为 0.88。

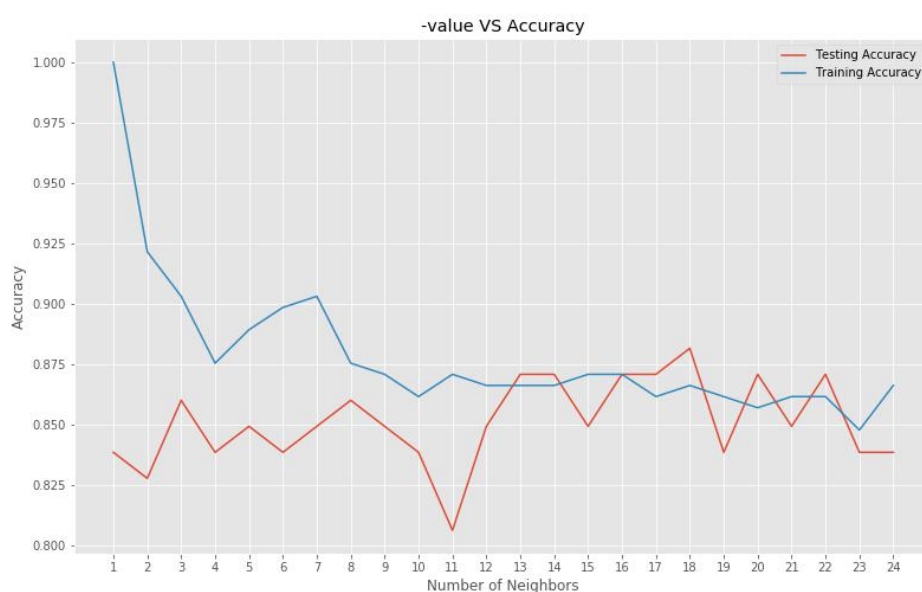


图 4 不同 k 值对应的结果

2.3.5 混淆矩阵及相关指标

混淆矩阵也是评判分类好坏的一个方法。先说明几个概念，TP(True Positive): 真实为 0，预测也为 0，FN(False Negative): 真实为 0，预测为 1，FP(False Positive): 真实为 1，预测为 0，TN(True Negative): 真实为 1，预测也为 1。混淆矩阵就是由这四个值的数量组成的矩阵。如图 5 所示。

	Predicted as Positive	Predicted as Negative
Labeled as Positive	True Positive(TP)	False Negative(FN)
Labeled as Negative	False Positive(FP)	True Negative(TN)

图 5 混淆矩阵图示

根据混淆矩阵各个位置的值，有以下 3 个主要的指标。分别是精确率 (Precision): 分类正确的正样本个数占分类器分成的所有正样本个数的比例，召

回率(Recall): 分类正确的正样本个数占正样本个数的比例和 F1-score (F1 值) : 为精确率和召回率的调和均值。各自的公式如式 1 到 3 所示。

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$R = \frac{TP}{TP + FN} \tag{2}$$

$$\frac{2}{F} = \frac{1}{P} + \frac{1}{R} \tag{3}$$

可以看出, f1 是综合精确率和召回率的指标, 它更加能反映模型整体的好坏。一般来说, 模型的 f1 值越高, 分类的效果越好。

关键代码如下:

```
from sklearn.metrics import classification_report, confusion_matrix
cm = confusion_matrix(y_test,prediction)
print('Confusion matrix: \n',cm)
print('Classification report: \n',classification_report(y_test,prediction))
```

K 取不同值时的结果分别如图 6 和如 7 所示。

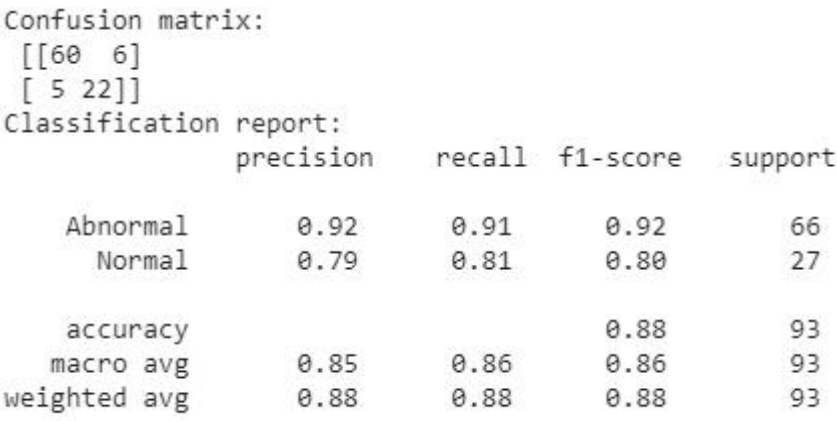


图 6 K=18 时的结果


```

Confusion matrix:
[[59  7]
 [ 6 21]]
Classification report:

```

	precision	recall	f1-score	support
Abnormal	0.91	0.89	0.90	66
Normal	0.75	0.78	0.76	27
accuracy			0.86	93
macro avg	0.83	0.84	0.83	93
weighted avg	0.86	0.86	0.86	93

图 7 K=3 时的结果

可以看到，k=18 时的 f1 值大于 k=3 时的值，在这个角度下看，k 取 18 的分类效果比 k=3 好。

3 感悟与总结

由于本人是第一次接触这种数据的处理，编程能力也有不足，因此在网上借鉴了很多程序的例子。这次模式识别的作业让我接触了以前放在 kaggle 上的真实数据集，感觉令自己的自信心有了加强。还有通过本次作业，使本人第一次在 github 上上传了一个项目吧。

本次作业主要就是根据人的生物力学特征来对其骨骼疾病进行了简单的二分类任务。采用了老师上课讲过的 knn 算法。虽然上课可能没有很认真的听讲，但在课后做的该作业包含的一些点感觉老师上课也都提过，也算是对知识点的补充与回顾吧。通过本次作业，使我对模式识别有了新的认识；通过本次作业，使我有对数据集进行基本操作的能力。

最后，再次感谢老师这半个学期的努力教学，在大学里这么负责尽职的老师不多了。

4 不足与展望

虽说基本上完成了该数据集的分类任务，但还是有以下不足的：

1. 程序很多都是借鉴别人写好的 2 算法只用了很简单的 knn 算法，也没啥改进 3 数据集其实比较简单，如果是很复杂的数据集，分类效果肯定是远远不够的。

展望：

希望在以后能认真读完李航的统计学习方法第二版，理解透彻机器学习的各类算法，并且能自行编程实现，然后对以后的科研任务有一定的帮助。

附：文件说明

column_2C_weka.csv: 数据集

homework.py: 程序