背景介绍

数据下载

这可能是模式识别文献中最着名的数据库。费舍尔的论文是该领域的经典之作，至今仍被频繁引用。（例如，参见Duda＆Hart。）数据集包含3个类别，每个类别50个实例，其中每个类别指的是一种鸢尾花。一类可以与另一类线性判别; 后者不能彼此线性判别。

对于三种不同的鸢尾花进行分类任务, ***数据集没有缺省的信息***，目标简单明了，通俗易懂。从项目地址下载号我们需要的数据集。

首先解压我们得到的数据，数据中有两个文件`iris.names`和`Iris.data`，前者是对当前数据集的一个简单解释，后者是我们将要训练的数据。

属性信息

下面是每一行数据中，都好分隔开的字段所代表的不同含义。

```
1.萼片长度cm
2.萼片宽度cm
3.花瓣长度cm
4.花瓣宽度cm
5.类：
- Iris Setosa
- Iris Versicolour
- Iris Virginica
```

这里，将分别使用两种方法 LDA 线性判别分析和 SVM 支持向量机。

分析过程

首先先来看一下数据的大体情况吧！

```
Jason@X1:~/flower/Data$ cat Iris.data  #观察原始数据集
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
4.6,3.1,1.5,0.2,0
5.0,3.6,1.4,0.2,0
5.4,3.9,1.7,0.4,0
4.6,3.4,1.4,0.3,0
5.0,3.4,1.5,0.2,0
4.4,2.9,1.4,0.2,0
4.9,3.1,1.5,0.1,0
5.4,3.7,1.5,0.2,0
4.8,3.4,1.6,0.2,0
4.8,3.0,1.4,0.1,0
4.3,3.0,1.1,0.1,0
5.8,4.0,1.2,0.2,0
5.7,4.4,1.5,0.4,0
5.4,3.9,1.3,0.4,0
5.1,3.5,1.4,0.3,0
5.7,3.8,1.7,0.3,0
5.1,3.8,1.5,0.3,0
5.4,3.4,1.7,0.2,0
```



```
5.1,3.7,1.5,0.4,0
4.6,3.6,1.0,0.2,0
5.1,3.3,1.7,0.5,0
4.8,3.4,1.9,0.2,0
5.0,3.0,1.6,0.2,0
5.0,3.4,1.6,0.4,0
5.2,3.5,1.5,0.2,0
5.2,3.4,1.4,0.2,0
4.7,3.2,1.6,0.2,0
4.8,3.1,1.6,0.2,0
5.4,3.4,1.5,0.4,0
5.2,4.1,1.5,0.1,0
5.5,4.2,1.4,0.2,0
4.9,3.1,1.5,0.1,0
5.0,3.2,1.2,0.2,0
5.5,3.5,1.3,0.2,0
4.9,3.1,1.5,0.1,0
4.4,3.0,1.3,0.2,0
5.1,3.4,1.5,0.2,0
5.0,3.5,1.3,0.3,0
4.5,2.3,1.3,0.3,0
4.4,3.2,1.3,0.2,0
5.0,3.5,1.6,0.6,0
5.1,3.8,1.9,0.4,0
4.8,3.0,1.4,0.3,0
5.1,3.8,1.6,0.2,0
4.6,3.2,1.4,0.2,0
5.3,3.7,1.5,0.2,0
5.0,3.3,1.4,0.2,0
7.0,3.2,4.7,1.4,1
6.4,3.2,4.5,1.5,1
6.9,3.1,4.9,1.5,1
5.5,2.3,4.0,1.3,1
6.5,2.8,4.6,1.5,1
5.7,2.8,4.5,1.3,1
6.3,3.3,4.7,1.6,1
4.9,2.4,3.3,1.0,1
6.6,2.9,4.6,1.3,1
5.2,2.7,3.9,1.4,1
5.0,2.0,3.5,1.0,1
5.9,3.0,4.2,1.5,1
6.0,2.2,4.0,1.0,1
6.1,2.9,4.7,1.4,1
5.6,2.9,3.6,1.3,1
6.7,3.1,4.4,1.4,1
5.6,3.0,4.5,1.5,1
5.8,2.7,4.1,1.0,1
6.2,2.2,4.5,1.5,1
5.6,2.5,3.9,1.1,1
5.9,3.2,4.8,1.8,1
6.1,2.8,4.0,1.3,1
6.3,2.5,4.9,1.5,1
6.1,2.8,4.7,1.2,1
6.4,2.9,4.3,1.3,1
6.6,3.0,4.4,1.4,1
6.8,2.8,4.8,1.4,1
6.7,3.0,5.0,1.7,1
6.0,2.9,4.5,1.5,1
5.7,2.6,3.5,1.0,1
5.5,2.4,3.8,1.1,1
5.5,2.4,3.7,1.0,1
5.8,2.7,3.9,1.2,1
6.0,2.7,5.1,1.6,1
```

```
5.4,3.0,4.5,1.5,1
6.0,3.4,4.5,1.6,1
6.7,3.1,4.7,1.5,1
6.3,2.3,4.4,1.3,1
5.6,3.0,4.1,1.3,1
5.5,2.5,4.0,1.3,1
5.5,2.6,4.4,1.2,1
6.1,3.0,4.6,1.4,1
5.8,2.6,4.0,1.2,1
5.0,2.3,3.3,1.0,1
5.6,2.7,4.2,1.3,1
5.7,3.0,4.2,1.2,1
5.7,2.9,4.2,1.3,1
6.2,2.9,4.3,1.3,1
5.1,2.5,3.0,1.1,1
5.7,2.8,4.1,1.3,1
6.3,3.3,6.0,2.5,2
5.8,2.7,5.1,1.9,2
7.1,3.0,5.9,2.1,2
6.3,2.9,5.6,1.8,2
6.5,3.0,5.8,2.2,2
7.6,3.0,6.6,2.1,2
4.9,2.5,4.5,1.7,2
7.3,2.9,6.3,1.8,2
6.7,2.5,5.8,1.8,2
7.2,3.6,6.1,2.5,2
6.5,3.2,5.1,2.0,2
6.4,2.7,5.3,1.9,2
6.8,3.0,5.5,2.1,2
5.7,2.5,5.0,2.0,2
5.8,2.8,5.1,2.4,2
6.4,3.2,5.3,2.3,2
6.5,3.0,5.5,1.8,2
7.7,3.8,6.7,2.2,2
7.7,2.6,6.9,2.3,2
6.0,2.2,5.0,1.5,2
6.9,3.2,5.7,2.3,2
5.6,2.8,4.9,2.0,2
7.7,2.8,6.7,2.0,2
6.3,2.7,4.9,1.8,2
6.7,3.3,5.7,2.1,2
7.2,3.2,6.0,1.8,2
6.2,2.8,4.8,1.8,2
6.1,3.0,4.9,1.8,2
6.4,2.8,5.6,2.1,2
7.2,3.0,5.8,1.6,2
7.4,2.8,6.1,1.9,2
7.9,3.8,6.4,2.0,2
6.4,2.8,5.6,2.2,2
6.3,2.8,5.1,1.5,2
6.1,2.6,5.6,1.4,2
7.7,3.0,6.1,2.3,2
6.3,3.4,5.6,2.4,2
6.4,3.1,5.5,1.8,2
6.0,3.0,4.8,1.8,2
6.9,3.1,5.4,2.1,2
6.7,3.1,5.6,2.4,2
6.9,3.1,5.1,2.3,2
5.8,2.7,5.1,1.9,2
6.8,3.2,5.9,2.3,2
6.7,3.3,5.7,2.5,2
6.7,3.0,5.2,2.3,2
6.3,2.5,5.0,1.9,2
```

```
6.5,3.0,5.2,2.0,2
6.2,3.4,5.4,2.3,2
5.9,3.0,5.1,1.8,2
```

很容易理解，这里我们抽出一个字段来看

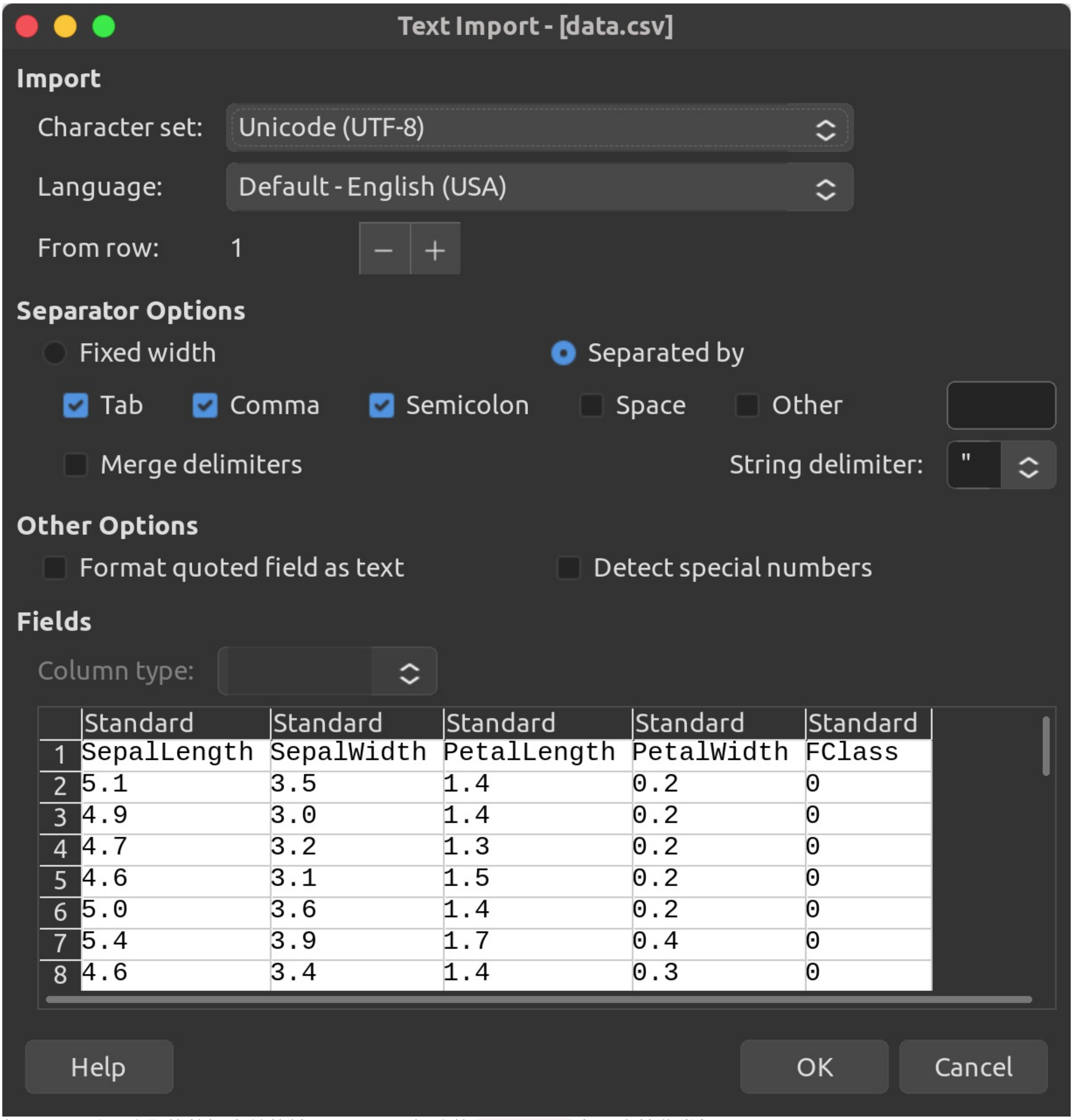
```
     5.9    ,    3.0   ,    5.1    ,    1.8    , 2
 |萼片长度cm|萼片宽度cm|花瓣长度cm|花瓣宽度cm| 类 |
```

数据预处理

好了既然我们已经知道数据是完整的，这里首先对原始数据预处理一下，改成比较适合的格式，这里可以使用python来完成，但是这里我是使用cut command将数据流重定向至一个信的csv文件，因为python还得编译的搞，不太舒服。

这里需要注意的两点是1.win下txt文件会在换行的时候加一个 `^M` 符号，虽然丢我们数据的处理可能不会产生影响，但是还是tr command去掉好了；2.在加入数据到csv文件时，不要忘记加入表头，否则一会数据分析的时候就很难看。

```
Jason@X1:~/flower/Dat$ echo SepalLength,SepalWidth,PetalLength,\
> PetalWidth,FClass > data.csv
Jason@X1:~/flower/Data$ cat Iris.data | cut -d',' -f 1,2,3,4,5 >> data.csv
# 没有返回错误信息，原目录下应该完成了转换。
Jason@X1:~/flower/Data$ xdg-open data.csv
```



好，可以看到我们的数据完美的被分隔了到了新建的 `data.csv` 中，这就非常舒服。

读入数据

利用 `pandas` 数据分析模块和 `numpy` 科学计算模块来分析数据。首先读入我们的数据：

```
import pandas as pd
import numpy as np
from pandas import Series,DataFrame

data_train=pd.read_csv('/home/jason/Documents/ML/flower/data.csv',engine =
'python',encoding='UTF-8')
data_train #dataframe格式
```

这里就可以看到data.csv中的数据了。但是只有数据表我们很难从中找出规律。所以接下来通过pandas中的方法来大体查看一下数据集的全貌。

```
>>> data_train.info()
```

运行一下唉

```
Jason@X1:~/flower/Dat$ py3 linear.py
[150 rows x 5 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
SepalLength    150 non-null float64
SepalWidth     150 non-null float64
PetalLength    150 non-null float64
PetalWidth     150 non-null float64
FClass         150 non-null int64
dtypes: float64(4), int64(1)
memory usage: 5.9 KB
None
```

这里可以看到这150条记录都是非空的，并且前四个字段为float，最后一个为int型，看来Fisher没有故意搞我们，确实数据没有缺省。

然后我们再使用describe来看一下

```
>>> data_train.describe()
       SepalLength  SepalWidth  PetalLength  PetalWidth      FClass
count   150.000000  150.000000   150.000000  150.000000  150.000000
mean      5.843333    3.054000     3.758667    1.198667    1.000000
std       0.828066    0.433594     1.764420    0.763161    0.819232
min       4.300000    2.000000     1.000000    0.100000    0.000000
25%       5.100000    2.800000     1.600000    0.300000    0.000000
50%       5.800000    3.000000     4.350000    1.300000    1.000000
75%       6.400000    3.300000     5.100000    1.800000    2.000000
max       7.900000    4.400000     6.900000    2.500000    2.000000
```

describe() 还是nice啊，我把describe的内容整理成一个表格，这样是不是更清晰一些

	Min	Max	Mean	SD	Class	Correlation
SepalLength	4.3	7.9	5.84	0.83	0.7826	
SepalWidth	2.0	4.4	3.05	0.43	-0.4194	
PetalLength	1.0	6.9	3.76	1.76	0.9490	(high!)
PetalWidth	0.1	2.5	1.20	0.76	0.9565	(high!)

可以观察到，表格中最后两个item中Class的致是最高的，可以假设一下我们的花朵分类应该会跟这两个item关系密切（PS：因为要求用三次二分类做，所以这里我们可以现不用猜测数据简单关系）。

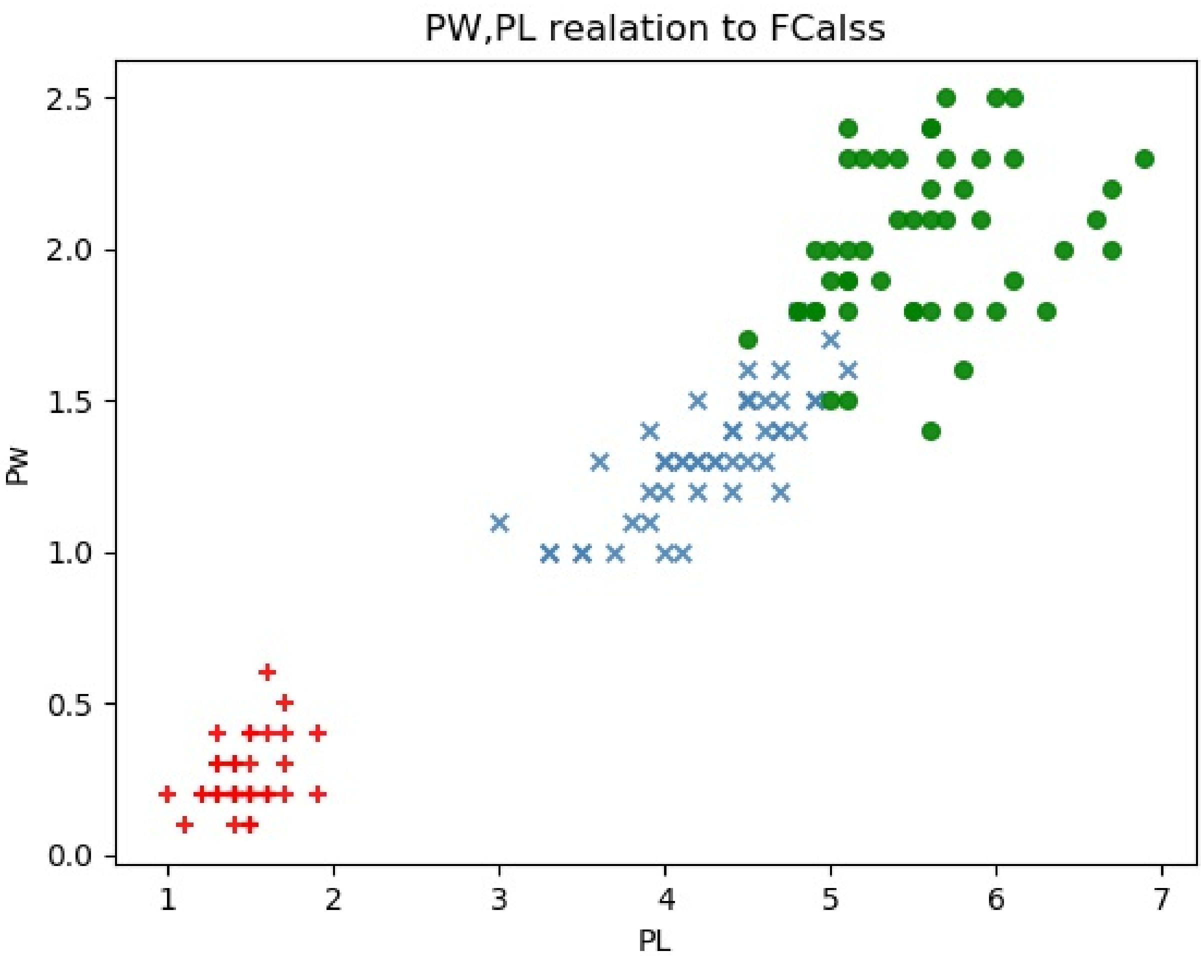
PL和PW与属性结果的关系

第一个猜想是和PL，PW有关，看一下情况

```
import pandas as pd
```

```
import numpy as np
from pandas import Series,DataFrame

data_train=pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')
import matplotlib.pyplot as plt
fig = plt.figure()
fig.set(alpha=0.2) #设定图表颜色颜色

plt.scatter(data_train.PetalLength[data_train.FClass==1], # x轴数据为PL
            data_train.PetalWidth[data_train.FClass==1], # y轴数据为PW
            s = 30, # 设置点的大小
            c = 'steelblue', # 设置点的颜色
            marker = 'x', # 设置点的形状
            alpha = 0.9, # 设置点的透明度
            )

plt.scatter(data_train.PetalLength[data_train.FClass==2],
            data_train.PetalWidth[data_train.FClass==2],
            s = 30,
            c = 'green',
            marker = 'o',
            alpha = 0.9,
            )

plt.scatter(data_train.PetalLength[data_train.FClass==0],
            data_train.PetalWidth[data_train.FClass==0],
            s = 30,
            c = 'red',
            marker = '+',
            alpha = 0.9,
            )
plt.ylabel(u"Pw")
plt.title(u"PW,PL realation to FCalss") #标题
plt.xlabel(u"PL")

plt.show()
```

```
Jason@X1:~/flower/Dat$ py3 linear.py
```



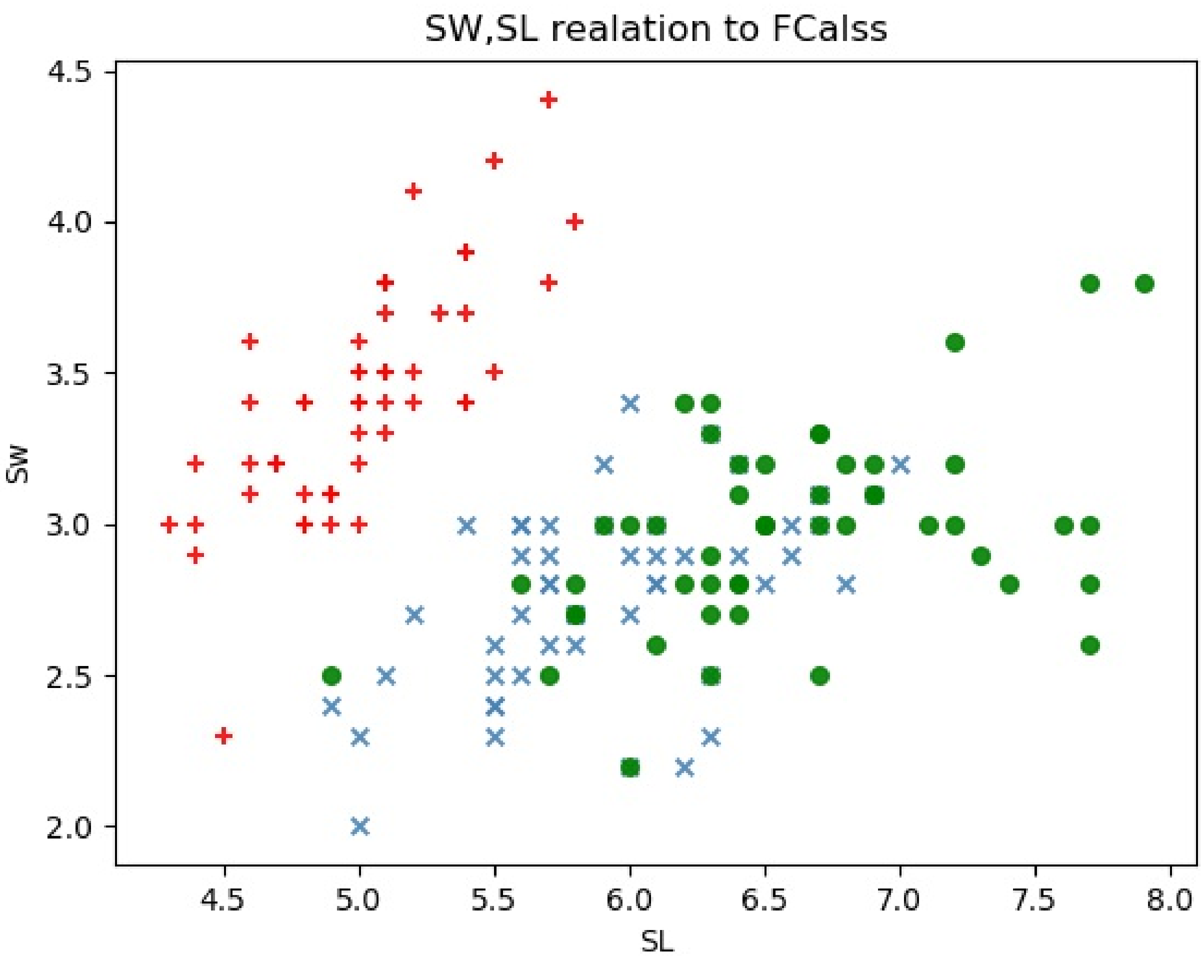
可以看到第一类鸢尾花Iris Setosa可以从后面两种中分离出来。

SL和SW与属性结果的关系

这样还是制作一张散点图

```
import pandas as pd
import numpy as np
from pandas import Series,DataFrame

data_train=pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')
import matplotlib.pyplot as plt
fig = plt.figure()
fig.set(alpha=0.2) #设定图表颜色颜色

plt.scatter(data_train.SepalLength[data_train.FClass==1], # x轴数据为PL
            data_train.SepalWidth[data_train.FClass==1], # y轴数据为PW
            s = 30, # 设置点的大小
            c = 'steelblue', # 设置点的颜色
            marker = 'x', # 设置点的形状
            alpha = 0.9, # 设置点的透明度
            )

plt.scatter(data_train.SepalLength[data_train.FClass==2],
            data_train.SepalWidth[data_train.FClass==2],
            s = 30,
            c = 'green',
            marker = 'o',
            alpha = 0.9,
```

```
            )

plt.scatter(data_train.SepalLength[data_train.FClass==0],
            data_train.SepalWidth[data_train.FClass==0],
            s = 30,
            c = 'red',
            marker = '+',
            alpha = 0.9,
            )
plt.ylabel(u"Sw")
plt.title(u"SW,SL realation to FCalss") #标题
plt.xlabel(u"SL")

plt.show()
```

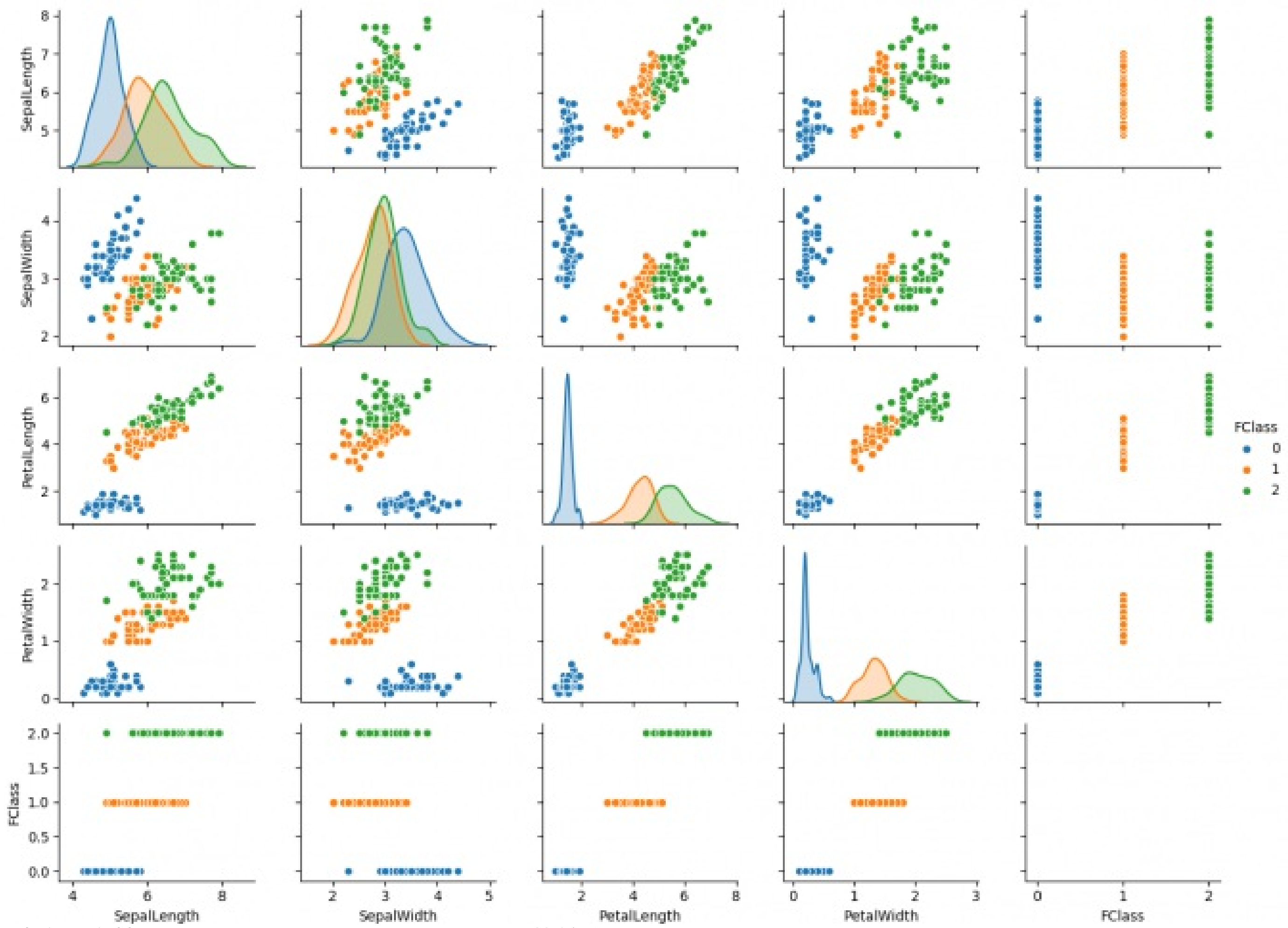


纳尼，看来SW分离不出来什么东西，和第一次是差不多的,看来通过某两对属性组就能分类的想法是不可能了。但是我们可以看出来，第一类鸢尾花无论是花瓣还是花萼都与后面两类差很多，分离出第一种化是比较容易的，重点放在如何分离拆开后两种花朵类型。

下面就要进入瞎猜的环节来看看还有哪些猜想是成立的，这里使用一个库函数帮我们理出所有的可能

```
import pandas as pd
import numpy as np
from pandas import Series,DataFrame

data_train=pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')
import matplotlib.pyplot as plt
fig = plt.figure()
fig.set(alpha=0.2) #设定图表颜色颜色
```

```python
import seaborn as sns
sns.pairplot(data_train,hue='FClass')

plt.show()
```



看来在所有情况下Iris Versicolour和Iris Virginica都比较难区分。

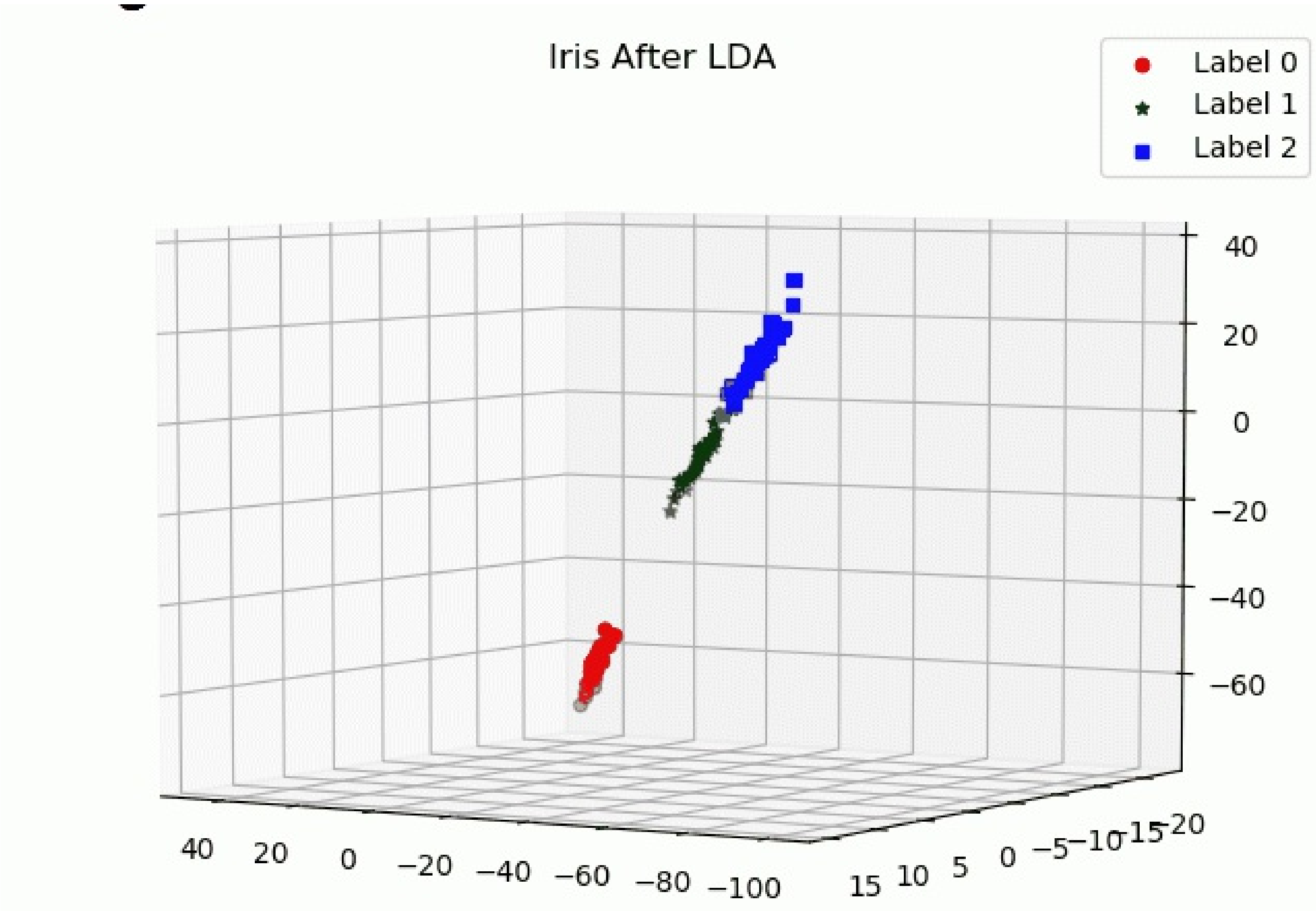
线性判别分析LDA

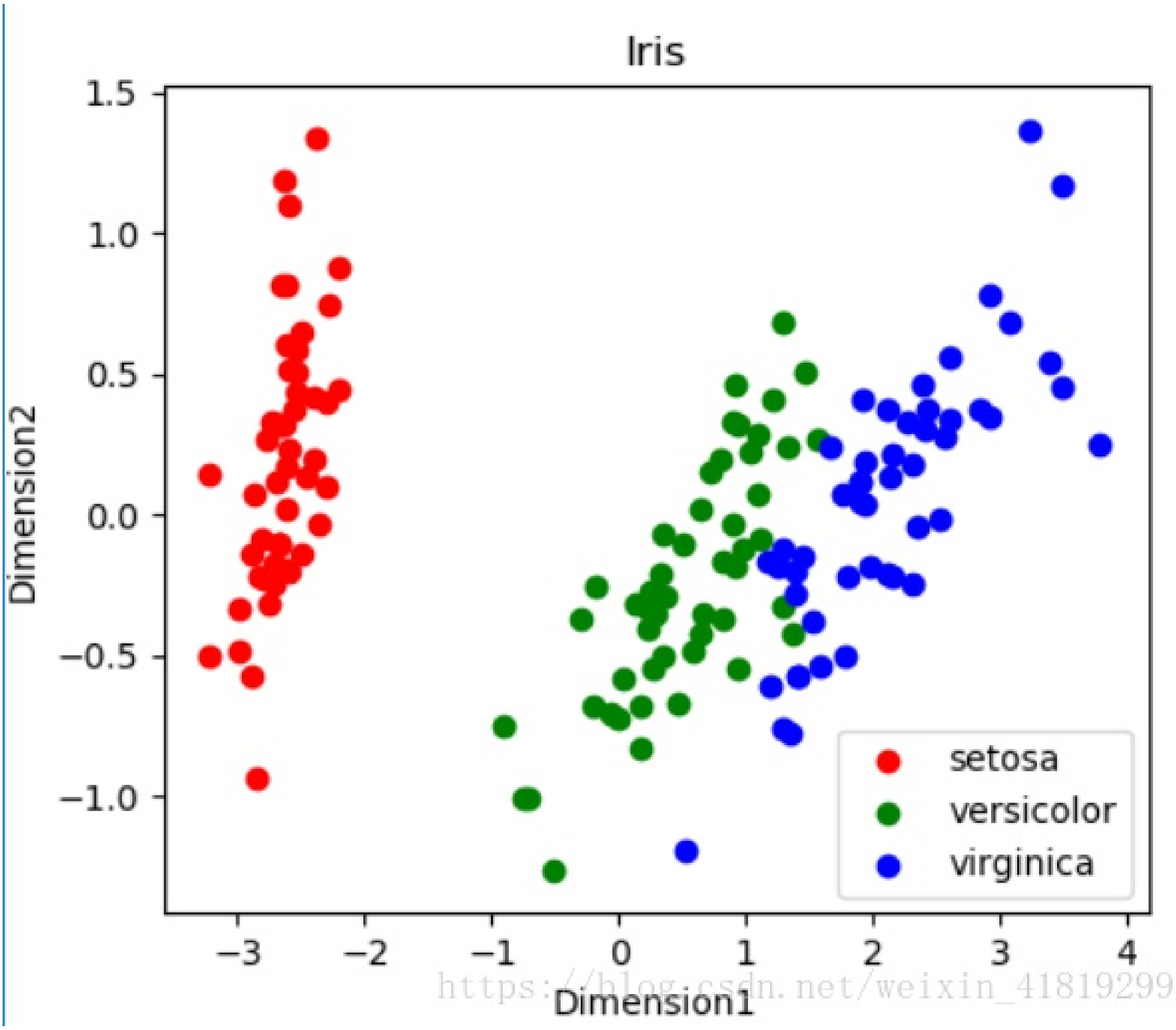
Linear Discriminant Analysis，器最主要的作用我觉得是对数据进行降维，通过将多维数据向更低空间投影，从而能获得一个易于理解的概念模型。

这里我们通过单个二分类问题的概念推到出多类线性判别分析。

低维空间的选择

如何选择一个维度合适的空间，作为我们的特征空间呢？是将一个 d 维数据集投影到一个 k （k<d)维子空间中,如何选择k的大小。比如对于目前我们系哪有的数据集，就有2维或3维这两种降维选择。


Iris After LDA
Label 0
Label 1
Label 2
40
20
0
−20
−40
−60
40 20 0 −20 −40 −60 −80 −100
15 10 5 0 −5 −10 −15 −20



这里用到的方法是求特征向量,然后将器归总到类内散度矩阵和类间散度矩阵。

$$\begin{align*} & S_w=\Sigma_0+\Sigma_1=\sum_{x \in X_0}(x-\mu_0)(x-\mu_0)^T+\sum_{x \in X_1}(x-\mu_1)(x-\mu_1)^T \\ & S_b=(\mu_0-\mu_1)(\mu_0-\mu_1)^T \end{align*}$$

每一个特征向量都对应一个特征值， 如果特征值的大小接近就代表我们投影到的空间维度比较合适。

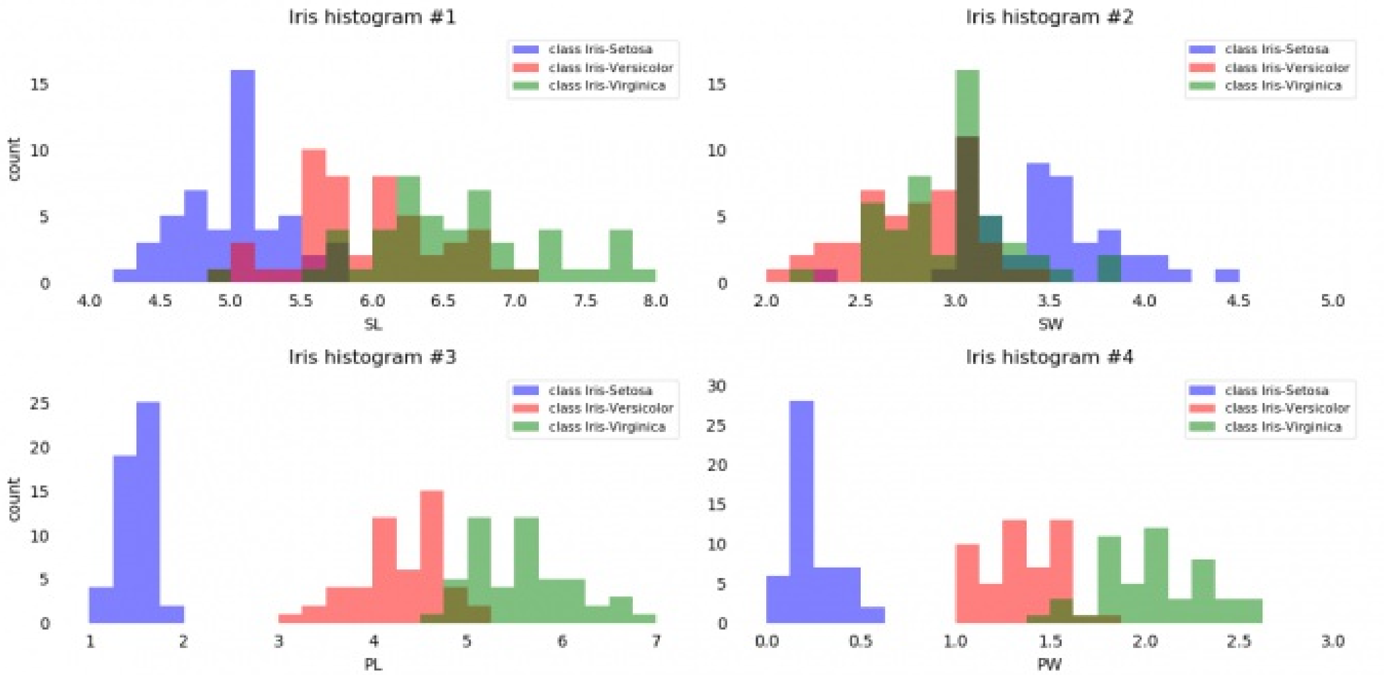
基本方法和步骤

- 计算数据集中不同类别数据的 d 维均值向量。
- 计算散度矩阵，包括类间、类内散度矩阵。
- 计算散度矩阵的特征向量 e1,e2,...,ed 和对应的特征值 λ1,λ2,...,λd。
- 特征向量按特征值大小降序排列，选前 k 个特征值对应的特征向量，组建一个 d×k 维矩阵——每一列就是一个特征向量。
- 用这个 d×k-维特征向量矩阵将样本变换到新的子空间。这一步可以写作矩阵乘法 $Y=X\times W$ 。 X 是 n×d 维矩阵，表示 n 个样本； Y 是变换到子空间后的 n×k 维样本。

conjecture

通过前面的一些基本分析，我们已经知道区分花朵类型的在四种特征里面，花瓣的长度、宽度更适合用来区分三种鸢尾花类别。但这是否正确还是要看一下结果,用直方图做一下映射

```python
import pandas as pd
import math
import numpy as np
from pandas import Series,DataFrame
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

label_dict = {0: 'Iris-Setosa', 1: 'Iris-Versicolor', 2:'Iris-Virginica'}
feature_dict = {i:label for i,label in zip(range(4),('SL','SW','PL','PW', ))}

data_train = pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')

data_train.columns = [l for i,l in sorted(feature_dict.items())] + ['FClass']


X = data_train[['SL','SW','PL','PW']].values
y = data_train['FClass'].values

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12,6))

for ax,cnt in zip(axes.ravel(), range(4)):

    # set bin sizes
    min_b = math.floor(np.min(X[:,cnt]))
    max_b = math.ceil(np.max(X[:,cnt]))
    bins = np.linspace(min_b, max_b, 25)

    # plottling the histograms
    for lab,col in zip(range(1,4), ('blue', 'red', 'green')):
        ax.hist(X[y==lab-1, cnt],
                   color=col,
                   label='class %s' %label_dict[lab-1],
                   bins=bins,
                   alpha=0.5,)
    ylims = ax.get_ylim()

    # plot annotation
    leg = ax.legend(loc='upper right', fancybox=True, fontsize=8)
    leg.get_frame().set_alpha(0.5)
    ax.set_ylim([0, max(ylims)+2])
    ax.set_xlabel(feature_dict[cnt])
    ax.set_title('Iris histogram #%s' %str(cnt+1))

    # hide axis ticks
    ax.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

axes[0][0].set_ylabel('count')
axes[1][0].set_ylabel('count')

fig.tight_layout()

plt.show()
```



仅凭这些简单的图形化展示，已经足以让我们得出结论：在四种特征里面，花瓣的长度、宽度更适合用来区分三种鸢尾花类别。

实际应用中，比起通过投影降维（此处即LDA），另一种比较好的办法是做特征筛选。像鸢尾花这样的低维数据集，看一眼直方图就能得到很多信息了。

LDA

- 计算数据的 d 维均值向量

```
import numpy as np
import math
import pandas as pd
from pandas import Series,DataFrame
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

feature_dict = {i:label for i,label in zip(range(4),('SL','SW','PL','PW', ))}

data_train = pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')

data_train.columns = [l for i,l in sorted(feature_dict.items())] + ['FClass']

#print (data_train.tail())

X_src = data_train[['SL','SW','PL','PW']].values
y_src = data_train['FClass'].values
X, X_spl, y, y_spl = train_test_split(X_src, y_src, test_size=0.3,
random_state=42,stratify=y_src)

#print (X.tail(),y.tail())
np.set_printoptions(precision=4)

mean_vectors = []
for clo in range(1,4):
    mean_vectors.append(np.mean(X[y==clo-1],axis=0))
    print('Mean Vector FClass %s: %s\n' %(clo-1,mean_vectors[clo-1]))
```

py一下

```
Jason@X1:~/flower/Data$ py3 LDA.py
Mean Vector FClass 0: [4.9886 3.4114 1.4886 0.2371]

Mean Vector FClass 1: [5.9486 2.7314 4.2371 1.3086]

Mean Vector FClass 2: [6.6829 3.0086 5.6314 2.0686]
```

- 计算散度矩阵 $$ S_w=\Sigma_0+\Sigma_1=\sum_{x \in X_0}(x-\mu_0)(x-\mu_0)^T+\sum_{x \in X_1}(x-\mu_1)(x-\mu_1)^T $$

```
S_W = np.zeros((4,4)) #4x4的矩阵
for clo,mv in zip(range(1,4), mean_vectors):
    class_sc_mat = np.zeros((4,4))
    for row in X[y == clo-1]:
        row, mv = row.reshape(4,1), mv.reshape(4,1)
        class_sc_mat += (row-mv).dot((row-mv).T)
    S_W += class_sc_mat
print('类内散度矩阵:\n', S_W)
```

py一下

```
Jason@X1:~/flower/Data$ py3 LDA.py
类内散度矩阵:
 [[26.9126  9.7063 17.5711  3.2614]
 [ 9.7063 13.2583  5.7343  3.0851]
 [17.5711  5.7343 19.4926  4.0283]
 [ 3.2614  3.0851  4.0283  3.8246]]
```

$$ S_b=(\mu_0-\mu_1)(\mu_0-\mu_1)^T $$

```
overall_mean = np.mean(X, axis=0)

S_B = np.zeros((4,4))
for i,mean_vec in enumerate(mean_vectors):
    n = X[y==i,:].shape[0]
    mean_vec = mean_vec.reshape(4,1)
    overall_mean = overall_mean.reshape(4,1)
    S_B += n * (mean_vec - overall_mean).dot((mean_vec - overall_mean).T)

print('类间散度矩阵:\n', S_B)
```

py一下

```
Jason@X1:~/flower/Data$ py3 LDA.py
类间散度矩阵:
 [[ 50.5328 -13.205  124.6189  54.7119]
 [-13.205    8.1842 -36.7686 -14.6504]
 [124.6189 -36.7686 311.056  135.2389]
 [ 54.7119 -14.6504 135.2389  59.263 ]]
```

- 求解矩阵的广义特征值

```
eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))

for i in range(len(eig_vals)):
    eigvec_sc = eig_vecs[:,i].reshape(4,1)
    print('\n特征向量 {}: \n{}'.format(i+1, eigvec_sc.real))
    print('特征值 {:}: {:.2e}'.format(i+1, eig_vals[i].real))
```

py一下

```
Jason@X1:~/flower/Data$ py3 LDA.py
特征向量 1:
[[-0.1895]
 [-0.3302]
 [ 0.5051]
 [ 0.7746]]
特征值 1: 3.23e+01

特征向量 2:
[[ 0.0448]
 [ 0.5551]
 [-0.2891]
 [ 0.7786]]
特征值 2: 3.72e-01

特征向量 3:
[[ 0.4887]
 [ 0.1004]
 [ 0.1842]
 [-0.8468]]
特征值 3: -4.92e-15

特征向量 4:
[[ 0.7403]
 [-0.3929]
 [-0.4662]
 [ 0.2834]]
特征值 4: 2.78e-15
```

特征向量和特征值代表了一个线性变换的形变程度，特征向量是形变的方向，特征值是形变的大小。

- 选择线性判别器构成新的特征子空间

将特征向量根据特征值的大小从高到低排序，然后选择前 k 个本征向量：

```
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)

print('特征值:\n')
for i in eig_pairs:
    print(i[0])
```

py一下

```
Jason@X1:~/flower/Data$ py3 LDA.py
特征值:

32.34182135648737
0.37171378445332187
4.915564356595094e-15
2.7775787346756613e-15
```

我们将4d空间投影到2d空间上，所以选择前两个特征向量。在LDA中，线性判别器的数目最多是 c−1，c 是总的类别数，这是因为类内散布矩阵 SB 是 c 个秩为1或0的矩阵的和。

按特征值的大小得到降序排列的本征对之后，现在就可以组建我们的 k×d-维特征向量矩阵 W 了（此时大小为 4×2），

这样就从最初的4维特征空间降到了2维的特征空间。

```
W = np.hstack((eig_pairs[0][1].reshape(4,1), eig_pairs[1][1].reshape(4,1)))
print('矩阵 W:\n', W.real)
```

py一下

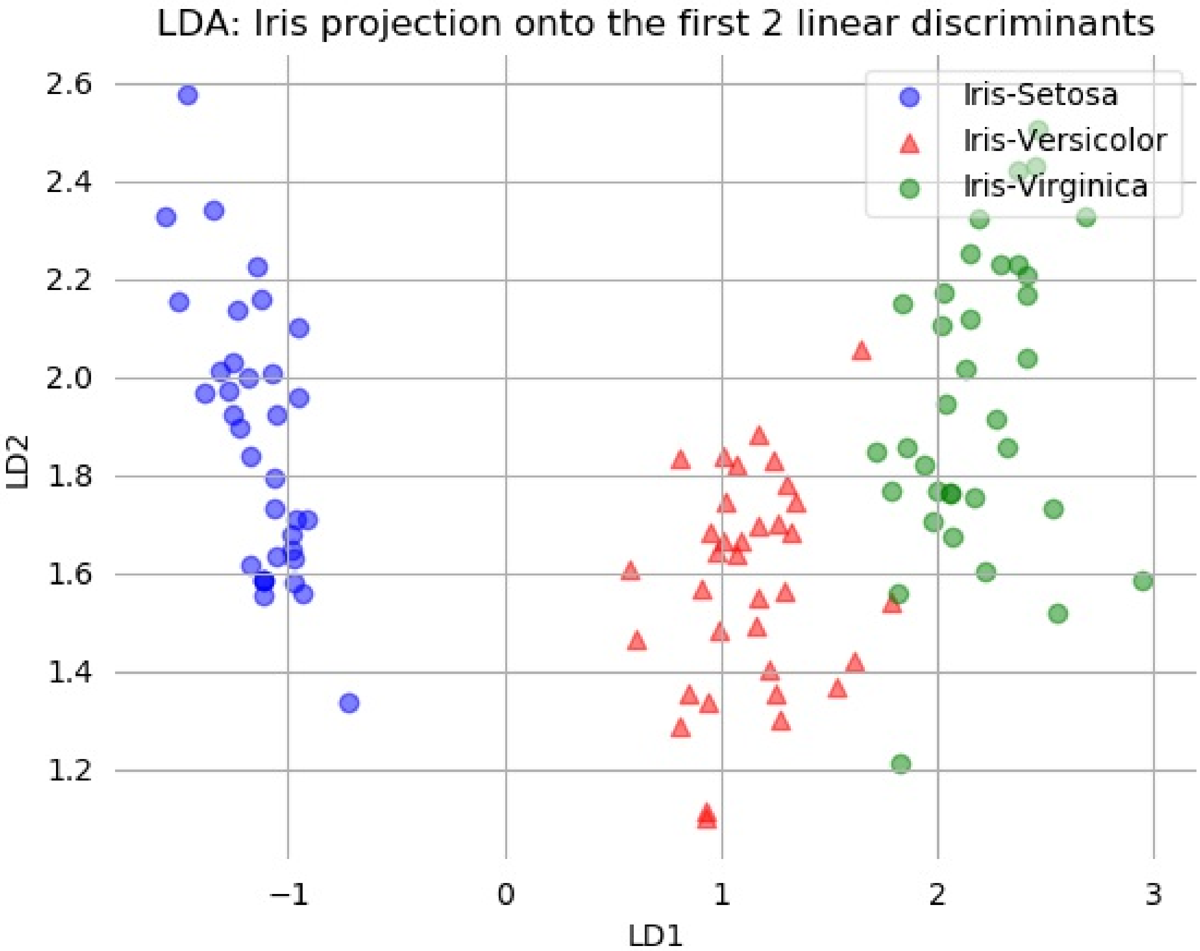
```
Jason@X1:~/flower/Data$ py3 LDA.py
矩阵 W:
 [[-0.1895  0.0448]
 [-0.3302  0.5551]
 [ 0.5051 -0.2891]
 [ 0.7746  0.7786]]
```

- 将样本变换到新的子空间中

使用上一步刚刚计算出的 4×2-维矩阵 W， 将样本变换到新的特征空间上：

```
X_lda = X.dot(W)
label_dict = {0: 'Iris-Setosa', 1: 'Iris-Versicolor', 2:'Iris-Virginica'}
def plot_step_lda():

    ax = plt.subplot(111)
    for label,marker,color in zip(
        range(1,4),('o', '^', 'o'),('blue', 'red', 'green')):

        plt.scatter(x=X_lda[:,0].real[y == label-1],
                y=X_lda[:,1].real[y == label-1],
                marker=marker,
                color=color,
                alpha=0.5,
                label=label_dict[label-1]
                )

    plt.xlabel('LD1')
    plt.ylabel('LD2')

    leg = plt.legend(loc='upper right', fancybox=True)
    leg.get_frame().set_alpha(0.5)
    plt.title('LDA: Iris projection onto the first 2 linear discriminants')

    # hide axis ticks
    plt.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

    plt.grid()
    plt.tight_layout
    plt.show()

plot_step_lda()
```

结果绘图



上方散点图展示了我们通过 LDA 构建的新的特征子空间。可以看到第一个线性判别器“LD1”把不同类数据区分得不错，第二个线性判别器就不行了。其原因在上面已经做了简单解释。

预测

然后就是对样本结果集的预测了。

```
X_lda = X.dot(W)
label_dict = {0: 'Iris-Setosa', 1: 'Iris-Versicolor', 2:'Iris-Virginica'}
def plot_step_lda():

    ax = plt.subplot(111)
    for label,marker,color in zip(
        range(1,4),('o', '^', 'o'),('blue', 'red', 'green')):

        plt.scatter(x=X_lda[:,0].real[y == label-1],
                y=X_lda[:,1].real[y == label-1],
                marker=marker,
                color=color,
                alpha=0.5,
                label=label_dict[label-1]
                )

    plt.xlabel('LD1')
    plt.ylabel('LD2')

    leg = plt.legend(loc='upper right', fancybox=True)
    leg.get_frame().set_alpha(0.5)
    plt.title('LDA: Iris projection onto the first 2 linear discriminants')
```

```
    # hide axis ticks
    plt.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

    plt.grid()
    plt.tight_layout
    plt.show()

plot_step_lda()
```

比较

下面比较一下我们的模型和实际的测试集有多大的误差

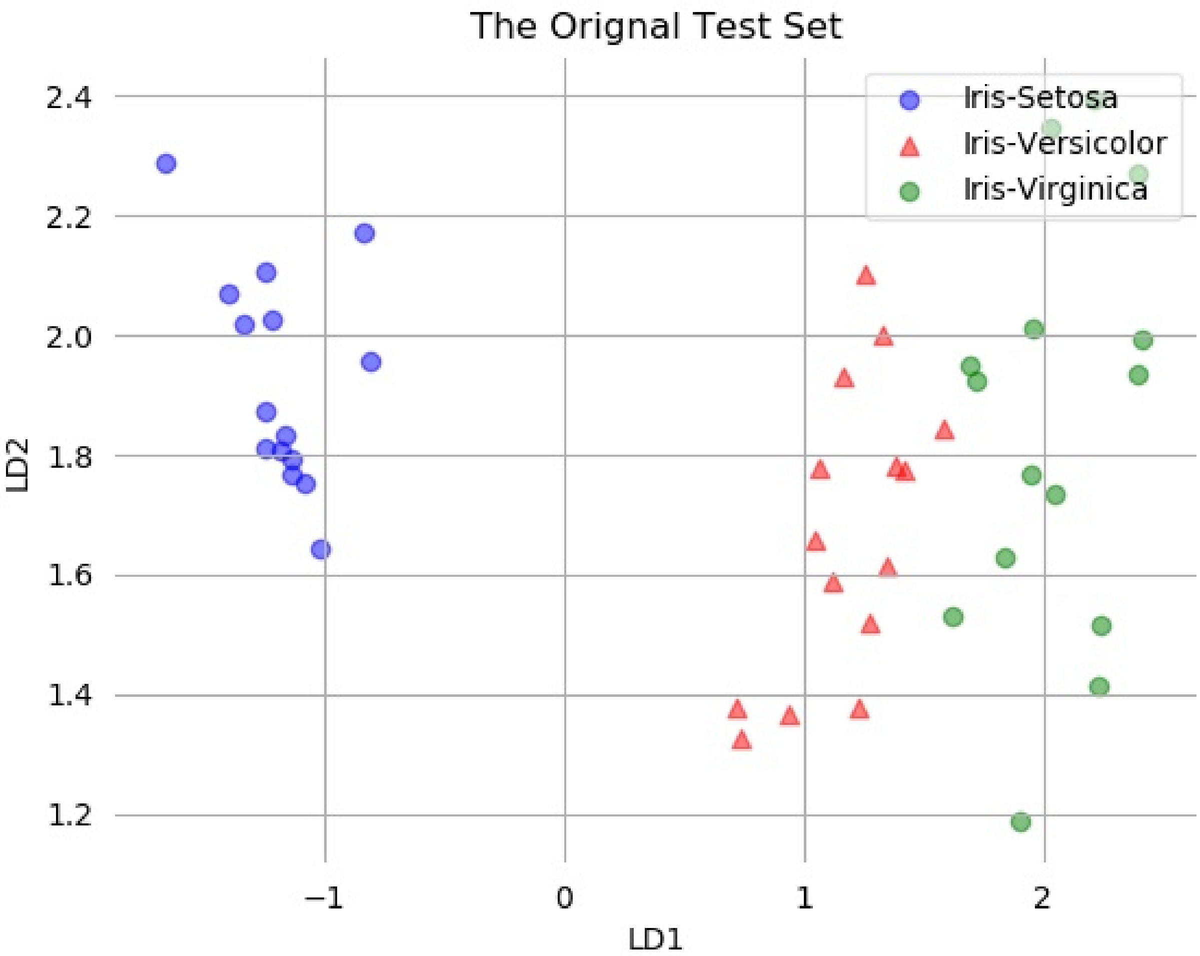
```
import numpy as np
import math
import pandas as pd
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

feature_dict = {i:label for i,label in zip(range(4),('SL','SW','PL','PW', ))}

data_train = pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')

data_train.columns = [l for i,l in sorted(feature_dict.items())] + ['FClass']

#print (data_train.tail())

X_src = data_train[['SL','SW','PL','PW']].values
y_src = data_train['FClass'].values
X, X_spl, y, y_spl = train_test_split(X_src, y_src, test_size=0.3,
random_state=42,stratify=y_src)

np.set_printoptions(precision=4)

mean_vectors = []
for clo in range(1,4):
    mean_vectors.append(np.mean(X[y==clo-1],axis=0))
    print('Mean Vector FClass %s: %s\n' %(clo-1,mean_vectors[clo-1]))

S_W = np.zeros((4,4)) #4x4的矩阵
for clo,mv in zip(range(1,4), mean_vectors):
    class_sc_mat = np.zeros((4,4))
    for row in X[y == clo-1]:
        row, mv = row.reshape(4,1), mv.reshape(4,1)
        class_sc_mat += (row-mv).dot((row-mv).T)
    S_W += class_sc_mat
print('类内散度矩阵:\n', S_W)

overall_mean = np.mean(X, axis=0)

S_B = np.zeros((4,4))
for i,mean_vec in enumerate(mean_vectors):
    n = X[y==i,:].shape[0]
```

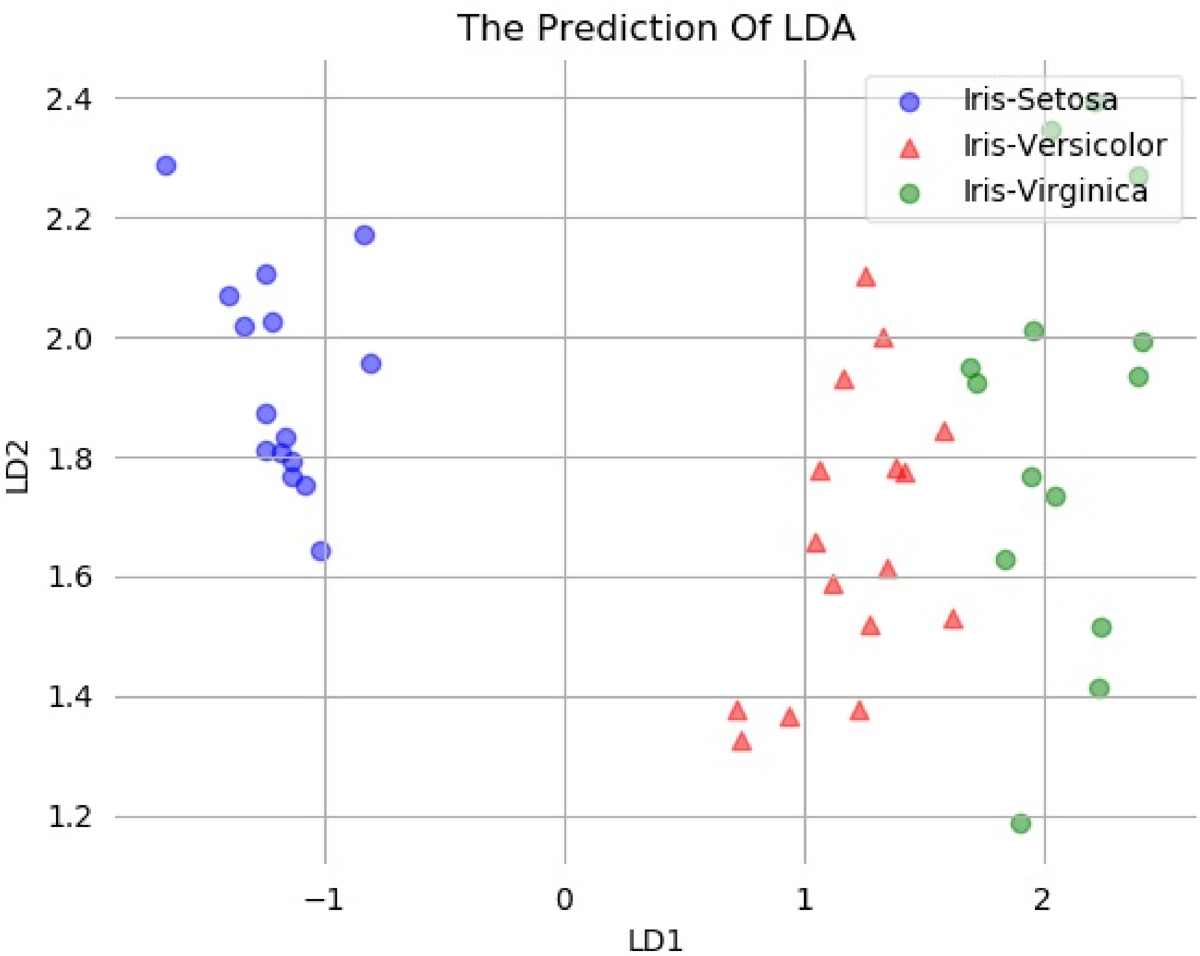


```
    mean_vec = mean_vec.reshape(4,1)
    overall_mean = overall_mean.reshape(4,1)
    S_B += n * (mean_vec - overall_mean).dot((mean_vec - overall_mean).T)

print('类间散度矩阵:\n', S_B)

eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))

for i in range(len(eig_vals)):
    eigvec_sc = eig_vecs[:,i].reshape(4,1)
    print('\n特征向量 {}: \n{}'.format(i+1, eigvec_sc.real))
    print('特征值 {:}: {:.2e}'.format(i+1, eig_vals[i].real))

eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)

print('特征值:\n')
for i in eig_pairs:
    print(i[0])

W = np.hstack((eig_pairs[0][1].reshape(4,1), eig_pairs[1][1].reshape(4,1)))
print('矩阵 W:\n', W.real)

X_lda = X_spl.dot(W)
label_dict = {0: 'Iris-Setosa', 1: 'Iris-Versicolor', 2:'Iris-Virginica'}

def plot_step_lda(argX,argy,title):

    ax = plt.subplot(111)
    for label,marker,color in zip(
        range(1,4),('o', '^', 'o'),('blue', 'red', 'green')):

        plt.scatter(x=argX[:,0].real[argy == label-1],
                y=argX[:,1].real[argy == label-1],
                marker=marker,
                color=color,
                alpha=0.5,
                label=label_dict[label-1]
                )

    plt.xlabel('LD1')
    plt.ylabel('LD2')

    leg = plt.legend(loc='upper right', fancybox=True)
    leg.get_frame().set_alpha(0.5)
    plt.title(title)

    # hide axis ticks
    plt.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

    plt.grid()
    plt.tight_layout
    plt.show()

y_pred=y_spl
```



```
#print (data_train.tail())
#print (X.tail())

plot_step_lda(X_lda,y_spl,"The Orignal Test Set")
for i in range(0,45):
    if X_lda[i][0]<=0:
        y_pred[i]=0
    elif X_lda[i][0]>0 and X_lda[i][0]<=1.69:
        y_pred[i]=1
    elif X_lda[i][0]>1.69:
        y_pred[i]=2

#print(y_pred)
plot_step_lda(X_lda,y_pred,"The Prediction Of LDA")
```





可以看到我们只有一个数据点出现了误差。

猜想验证

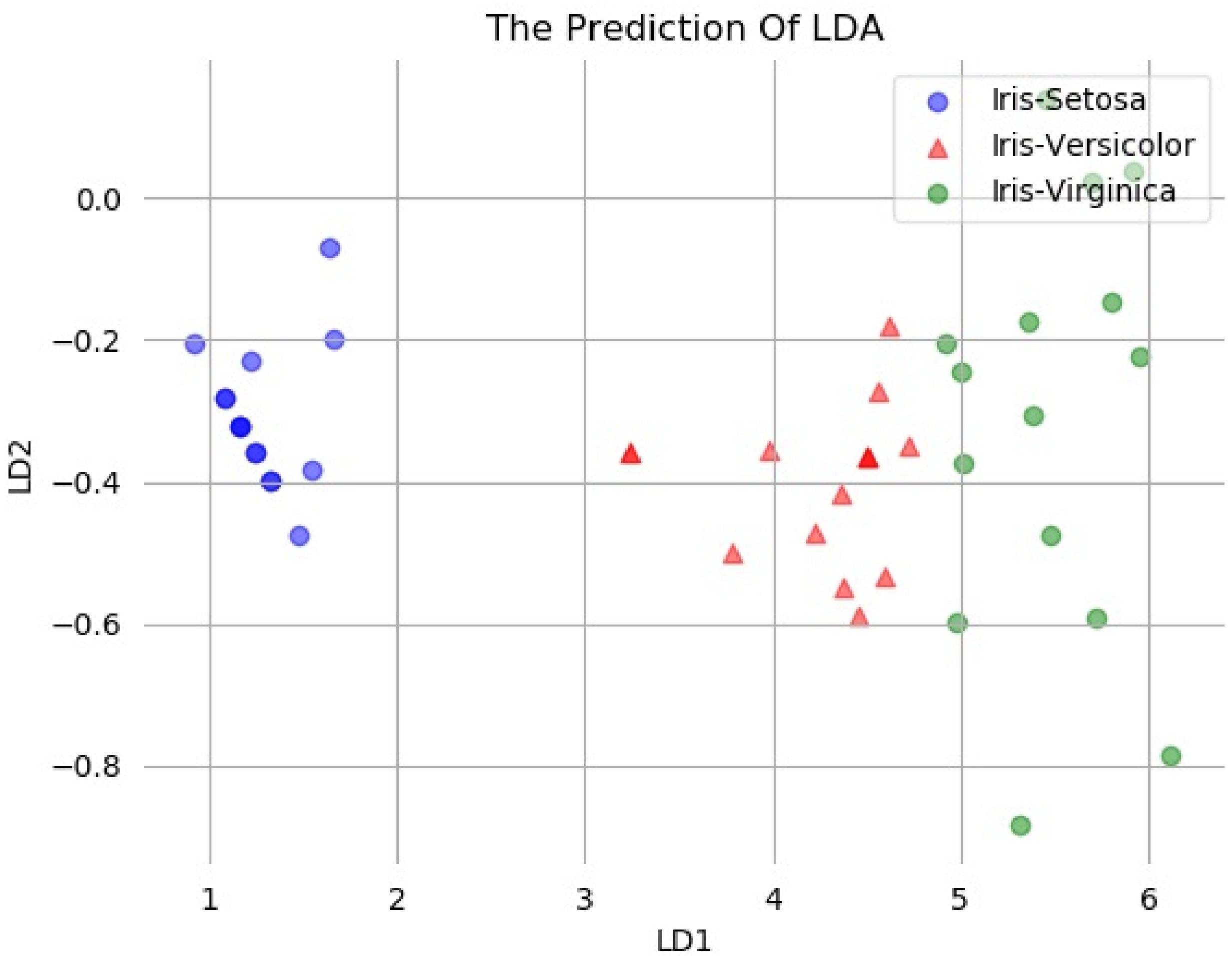
```
import numpy as np
import math
import pandas as pd
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

feature_dict = {i:label for i,label in zip(range(4),('SL','SW','PL','PW', ))}

data_train = pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')

data_train.columns = [l for i,l in sorted(feature_dict.items())] + ['FClass']

#print (data_train.tail())

X_src = data_train[['PL','PW']].values
y_src = data_train['FClass'].values
X, X_spl, y, y_spl = train_test_split(X_src, y_src, test_size=0.3,
random_state=42,stratify=y_src)

np.set_printoptions(precision=4)

mean_vectors = []
for clo in range(1,2):
    mean_vectors.append(np.mean(X[y==clo-1],axis=0))
    print('Mean Vector FClass %s: %s\n' %(clo-1,mean_vectors[clo-1]))
```

```python
S_W = np.zeros((2,2)) #2x2的矩阵
for clo,mv in zip(range(1,2), mean_vectors):
    class_sc_mat = np.zeros((2,2))
    for row in X[y == clo-1]:
        row, mv = row.reshape(2,1), mv.reshape(2,1)
        class_sc_mat += (row-mv).dot((row-mv).T)
    S_W += class_sc_mat
print('类内散度矩阵:\n', S_W)

overall_mean = np.mean(X, axis=0)

S_B = np.zeros((2,2))
for i,mean_vec in enumerate(mean_vectors):
    n = X[y==i,:].shape[0]
    mean_vec = mean_vec.reshape(2,1)
    overall_mean = overall_mean.reshape(2,1)
    S_B += n * (mean_vec - overall_mean).dot((mean_vec - overall_mean).T)

print('类间散度矩阵:\n', S_B)

eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))

for i in range(len(eig_vals)):
    eigvec_sc = eig_vecs[:,i].reshape(2,1)
    print('\n特征向量 {}: \n{}'.format(i+1, eigvec_sc.real))
    print('特征值 {:}: {:.2e}'.format(i+1, eig_vals[i].real))

eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)

print('特征值:\n')
for i in eig_pairs:
    print(i[0])

W = np.hstack((eig_pairs[0][1].reshape(2,1), eig_pairs[1][1].reshape(2,1)))
print('矩阵 W:\n', W.real)

X_lda = X_spl.dot(W)
label_dict = {0: 'Iris-Setosa', 1: 'Iris-Versicolor', 2:'Iris-Virginica'}

def plot_step_lda(argX,argy,title):

    ax = plt.subplot(111)
    for label,marker,color in zip(
        range(1,4),('o', '^', 'o'),('blue', 'red', 'green')):

        plt.scatter(x=argX[:,0].real[argy == label-1],
                y=argX[:,1].real[argy == label-1],
                marker=marker,
                color=color,
                alpha=0.5,
                label=label_dict[label-1]
                )

    plt.xlabel('LD1')
    plt.ylabel('LD2')

    leg = plt.legend(loc='upper right', fancybox=True)
    leg.get_frame().set_alpha(0.5)
    plt.title(title)
```

```
    # hide axis ticks
    plt.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

    plt.grid()
    plt.tight_layout
    plt.show()

y_pred=y_spl
#print (data_train.tail())
#print (X.tail())

plot_step_lda(X_lda,y_spl,"The Orignal Test Set")
for i in range(0,45):
    if X_lda[i][0]<=2.5:
        y_pred[i]=0
    elif X_lda[i][0]>2.5 and X_lda[i][0]<=4.9:
        y_pred[i]=1
    elif X_lda[i][0]>4.9:
        y_pred[i]=2

#print(y_pred)
plot_step_lda(X_lda,y_pred,"The Prediction Of LDA")
```

```
Jason@X1:~/flower/Dat$ py3 conj.py
Mean Vector FClass 0: [1.4886 0.2371]

类内散度矩阵:
 [[0.7954 0.1149]
 [0.1149 0.3417]]
类间散度矩阵:
 [[184.6903  77.7966]
 [ 77.7966  32.77  ]]

特征向量 1:
[[0.7997]
 [0.6003]]
特征值 1: 2.76e+02

特征向量 2:
[[-0.3882]
 [ 0.9216]]
特征值 2: 0.00e+00
特征值:

275.7224210820924
0.0
矩阵 W:
 [[ 0.7997 -0.3882]
 [ 0.6003  0.9216]]
```


The Orignal Test Set
Iris-Setosa
Iris-Versicolor
Iris-Virginica
0.0
−0.2
−0.4
−0.6
−0.8
LD2
1
2
3
4
5
6
LD1



这次分类错误了两个数据点，看来SW，SL对于分类也有一定的作用（虽然很小）

PCA

主成分分析（PCA）是一种统计过程，它使用正交变换将可能相关变量的一组观察值（每个实体都采用各种数值）转换为一组称为主成分的线性不相关变量值。如果有 ñ 观察与 p 变量，然后是不同主成分的数量 min(n-1,p)。这种转换的定义方式是第一主成分具有尽可能大的方差（即，尽可能多地考虑数据的可变性），并且每个后续成分依次在约束下具有最高的方差。它与前面的组件正交。得到的矢量（每个是变量的线性组合并包含n个观测值）是不相关的正交基组。PCA对原始变量的相对缩放敏感。

```
import pandas as pd
import math
import numpy as np
from pandas import Series,DataFrame
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt

label_dict = {0: 'Iris-Setosa', 1: 'Iris-Versicolor', 2:'Iris-Virginica'}
feature_dict = {i:label for i,label in zip(range(4),('SL','SW','PL','PW', ))}

data_train = pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')

data_train.columns = [l for i,l in sorted(feature_dict.items())] + ['FClass']


X = data_train[['SL','SW','PL','PW']].values
y = data_train['FClass'].values
#X_src = data_train[['SL','SW','PL','PW']].values
```

```
#y_src = data_train['FClass'].values
#X, X_spl, y, y_spl = train_test_split(X_src, y_src, test_size=0.3,
random_state=42,stratify=y_src)

from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

target_names = label_dict

pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)
#X_p = pca.predict(X_spl)

lda = LDA(n_components=2)
X_r2 = lda.fit(X, y).transform(X)
X_p = lda.predict(X)

label_dict = {0: 'Iris-Setosa', 1: 'Iris-Versicolor', 2:'Iris-Virginica'}
def plot_step_lda(X,title):

    ax = plt.subplot(111)
    for label,marker,color in zip(
        range(1,4),('o', '^', 'o'),('blue', 'red', 'green')):

        plt.scatter(x=X[:,0].real[y == label-1],
                y=X[:,1].real[y == label-1],
                marker=marker,
                color=color,
                alpha=0.5,
                label=label_dict[label-1]
                )

    plt.xlabel('LD1')
    plt.ylabel('LD2')

    leg = plt.legend(loc='upper right', fancybox=True)
    leg.get_frame().set_alpha(0.5)
    plt.title(title)

    # hide axis ticks
    plt.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

    plt.grid()
    plt.tight_layout
    plt.show()

plot_step_lda(X_r,'PCA')
plot_step_lda(X_r2,'LDA')
# Percentage of variance explained for each components
print('explained variance ratio (first two components): %s'
      % str(pca.explained_variance_ratio_))

plt.figure()
```

支持向量机SVM

数据集分割

这里每个化的种类由50个item，所以我们抽出40个进行训练，剩下的10个做测试集，这里使用train_test_split函数

```
import pandas as pd
from pandas import Series,DataFrame
from sklearn.model_selection import train_test_split

data_train=pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')


x = data_train[["SepalLength","SepalWidth","PetalLength","PetalWidth"]]
y = data_train["FClass"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42,stratify=y)
```

回归

直接调用sklearn中的函数即可，完全不需要写逻辑。其实LDA也是，但是为了理解内容我们还是稍微写一下LDA的。

```
import pandas as pd
from pandas import Series,DataFrame
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

data_train=pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')


x =data_train[["SepalLength","SepalWidth","PetalLength","PetalWidth"]]
y = data_train["FClass"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42,stratify=y)


classifier = LogisticRegression()
classifier.fit(x_train, y_train)
```

最后py运行一下

性能评估

```
import pandas as pd
from pandas import Series,DataFrame
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

data_train=pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')

x =data_train[["SepalLength","SepalWidth","PetalLength","PetalWidth"]]
y = data_train["FClass"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42,stratify=y)
# 这里本来是像使用40-10来分类的，结果发现这样使用后直接精度到0.98没太有优化空间了，所以改为35-15
classifier = LogisticRegression()
classifier.fit(x_train, y_train)
```

```
prdt_y = classifier.predict(x_test)
print (metrics.classification_report(y_test,prdt_y))
print (metrics.accuracy_score(y_test,prdt_y))
```

py一下

```
Jason@X1:~/flower/Data$ py3 sdfs.py
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.73      0.85        15
           2       0.79      1.00      0.88        15

   micro avg       0.91      0.91      0.91        45
   macro avg       0.93      0.91      0.91        45
weighted avg       0.93      0.91      0.91        45

0.9111111111111111
```

可以看到对于Iris Setosa - Iris Versicolour - Iris Virginica三种不同的花，我们分类的精度，召回率， $F1$ ，最后显示的正确率。

模型改进

正确率只有0.9啊，太捞了。我们能不能继续优化一下来。

软间隔与正则化

women知道优化目标中的第一项用来描述超平面的间隔大小，另一项 $\sum^{m}_{i=1}\iota(f(x_i),y_i)$ 用来表示训练集上的误差，课些微更一般的形式

$$ \min \limits_f \Omega(f) + C \sum_{i=1}^{m} \iota(f(x_i),y_i) $$

线性判别中， C 这个正则化常数，用于对经验风险和结构风险进行折中，而我们的 `LogisticRegression()` 方法中也可以设置这个参数。我们长将正则化的程度降低，看看会有什么不一样的结果。

```
>>> classifier = LogisticRegression(C=1e3)
```

再次py一下我们可以发现正确率果然提高了一些

```
Jason@X1:~/flower/Dat$ py3 sdfs.py
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.83      1.00      0.91        15
           2       1.00      0.80      0.89        15

   micro avg       0.93      0.93      0.93        45
   macro avg       0.94      0.93      0.93        45
weighted avg       0.94      0.93      0.93        45

0.9333333333333333
```

solver参数

`LogisticRegression()` 包含的参数当然不仅仅只有C，我们还可以选择其他的优化方法，这里就要用到我们的solver参数了。

```
>>> classifier = LogisticRegression(C=1e3, solver='lbfgs')
#将优化器改为L-BFGS梯度下降优化
```

可以自己py一下看看结果，这里就不再赘述了，但是需要注意的是，方法之间没有高低，根据第一章中的“没有免费午餐”定理，只有适合的才是最好的。下面是不同优化器:

```
'liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga'
```

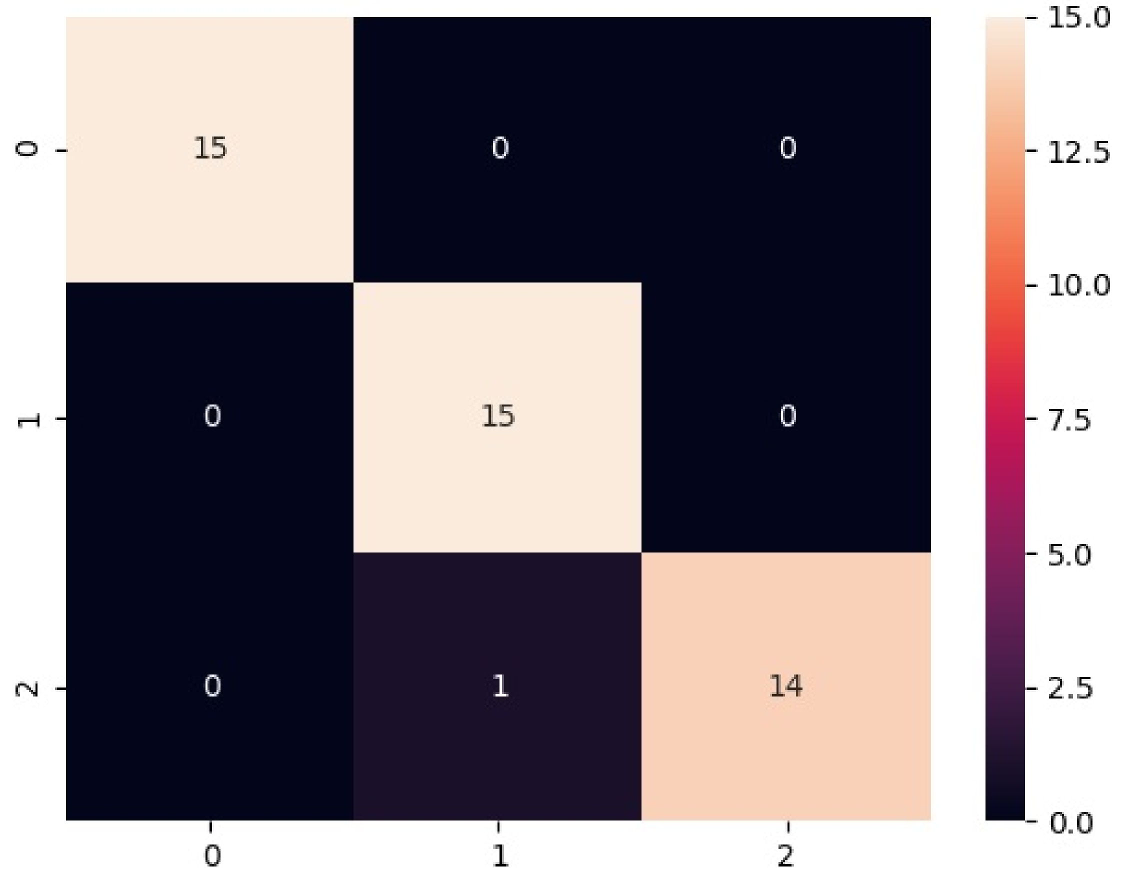
multi_class参数

这个参数的默认值为'ovr',也就是将一个类的样例当作正例，其它类作为反例，来训练多个二分类器，和我们的思路是一样的；'multinomial'表示最小化多项式损失满足整个概率分布，也就是Softmax分类器。

```
>>> classifier = LogisticRegression(C=1e3, solver='sags',multi_class='multinomial')
#优化器改为随机平均梯度下降，multi改为Softmax
```

这样处理之后，我们的精度达到了0.98，这样就十分可以了，继续处理有可能会出现过拟合的情况。

观察



观察一下混淆矩阵，我们的模型只在一个测试上分类错误！

复制参数

将我们训练的模型放入一个3*4的矩阵中，通过这个矩阵我们可以得到三个二元逻辑回归模型，系数矩阵就是coef，截距就是intercept。

```
import pandas as pd
from pandas import Series,DataFrame
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

data_train=pd.read_csv('/home/jason/Documents/ML/flower/Data/data.csv')

x =data_train[["SepalLength","SepalWidth","PetalLength","PetalWidth"]]
y = data_train["FClass"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42,stratify=y)

classifier = LogisticRegression(C=1e3)
classifier.fit(x_train, y_train)

prdt_y = classifier.predict(x_test)

coef = pd.DataFrame(classifier.coef_,columns=data_train.columns[0:4])
coef["intercept"] = classifier.intercept_
print (coef.round(2))
```

py一下，

```
Jason@X1:~/flower/Dat$ py3 args.py
   SepalLength  SepalWidth  PetalLength  PetalWidth  intercept
0         1.31        2.91        -3.99       -1.96       0.68
1         0.97       -0.22        -0.25       -1.77       1.61
2        -2.28       -2.69         4.24        3.72      -2.29
```

也就得到了三个线性回归方程 $$ \begin{align*} & P(FClass==0)=\sigma(1.31SL+2.91SW-3.99PL-1.96PW+0.68) \\ & P(FClass==0)=\sigma(0.97SL-0.22SW-0.25PL-1.77PW+1.61) \\ & P(FClass==0)=\sigma(-2.28SL-2.69SW+4.24PL+3.72PW-2.29) \\ \end{align*} $$

相关论文与参考资料：

- Fisher，RA“在分类学问题中使用多次测量”年度优生学，7，第二部分，179-188（1936）；同样在“对数学统计的贡献”（John Wiley，NY，1950）中。 网站链接
- Duda，RO，＆Hart，PE（1973）模式分类和场景分析。（Q327.D83）John Wiley＆Sons。国际标准书号0-471-22361-1。见第218页。 网站链接
- Dasarathy，BV（1980）“邻近地区：在部分暴露环境中识别的新系统结构和分类规则”。IEEE模式分析和机器智能交易，卷。PAMI-2，No.1,67-71。 网站链接
- Gates，GW（1972）“缩减的最近邻规则”。IEEE信息理论学报，1972年5月，431-433。 网站链接
- scikit learn
- sebastianraschka's blog

