# fisher 线性判别

yfpeng

September 2022

## 1 算法实现

根据 PPT 中提供的公式，我们依次计算出均值和方差，然后利用这两个指标找出最佳投影向量和分类阈值，算法的实现如下：

```python
#fisher算法
#里面乘以-100000是为了防止数值过小
def fisher(train_data):
    u1=np.matrix(np.zeros(2))
    u2=np.matrix(np.zeros(2))
    sigma1=np.matrix(np.zeros((2,2)))
    sigma2=np.matrix(np.zeros((2,2)))
    #计算均值
    pos_num=0.0
    neg_num=0.0
    for i in range(train_data.shape[0]):
        x=np.matrix(train_data[i,0:2])
        y=train_data[i,2]
        if y>0:
            u1+=x
            pos_num+=1
        else:
            u2+=x
            neg_num+=1
    u1=u1/pos_num
    u2=u2/neg_num
    for i in range(train_data.shape[0]):
        x=np.matrix(train_data[i,0:2])
        y=train_data[i,2]
        if y>0:
            sigma1+=(x-u1).T*(x-u1)
        else:
            sigma2+=(x-u2).T*(x-u2)
    s_w=sigma1+sigma2
    w=s_w.I*(u1.T-u2.T)
    threshold=w.T*(u1.T+u2.T)/2
    return np.array(w.T),np.array(float(threshold))
```

图 1: fisher 线性判别算法实现

## 2 实验结果

根据上面的算法，分别求出最佳投影向量和分类阈值如下：

$$\omega^* = \begin{pmatrix} -0.01745735 \\ -0.01618433 \end{pmatrix}$$

s=0.0012377763635528713

## 3 正确率统计

以下是我的正确率统计的算法实现和统计结果

```python
#测试函数
def eval(test_set,weight,threshold):
    error_rate=0
    weight=np.squeeze(weight)
    for j in range(test_set.shape[0]):
        if (np.dot(weight,test_set[j,0:-1])-threshold)*test_set[j][-1]<0:
            error_rate+=1
    print('正确率为:{}%'.format((1-error_rate/test_set.shape[0])*100))
```

```python
eval(train_set,w,threshold)
```

```
正确率为:100.0%
```

+ 代码 + Markdown

```python
eval(test_set,w,threshold)
```

```
正确率为:100.0%
```

图 2: evaluation 算法和实验结果

# 4 可视化

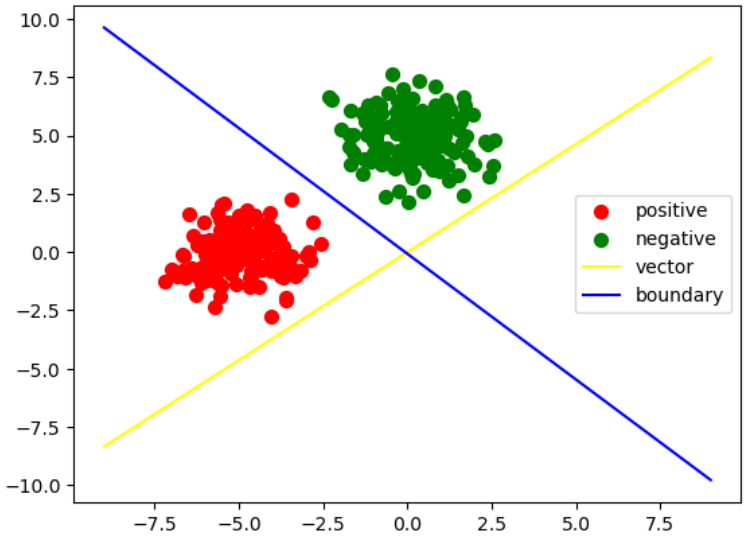


图 3: 训练集可视化

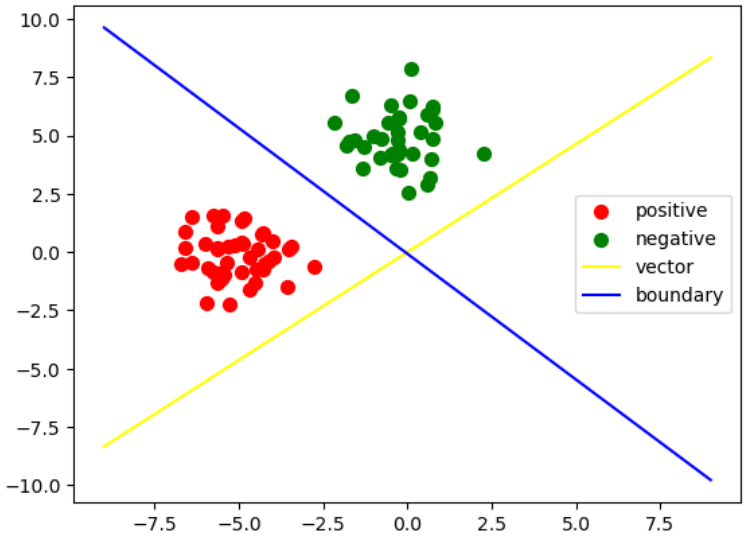


图 4: 测试集可视化

其中的线条的意义是：黄线是我们的最佳投影向量，蓝线是处于分类阈值上的样本点所组成的直线（也可作为分类面），按理来说，两者应该垂直，但图像中似乎不垂直，这也许是因为浮点误差。