大家好，这篇是有关Learning from data第三章习题的详解，这一章主要介绍了线性回归，Logistic回归以及特征转换。

我的github地址：
https://github.com/Doraemonzzz

个人主页：
http://doraemonzzz.com/

参考资料:
https://blog.csdn.net/a1015553840/article/details/51085129
http://www.vynguyen.net/category/study/machine-learning/page/6/
http://book.caltech.edu/bookforum/index.php
http://beader.me/mlnotebook/

Chapter 3 The Linear Model

Part 1:Exercise

Exercise 3.1 (Page 79)

Will PLA ever stop updating if the data is not linearly separable?

如果数据不是线性可分的，那么PLA不会终止，因为PLA每次挑选一个错分的数据做更新。

Exercise 3.2 (Page 80)

Take $d = 2$ and create a data set $\mathcal{D}$ of size $N = 100$ that is not linearly separable. You can do so by first choosing a random line in the plane as your target function and the inputs x_n of the data set as random points in the plane. Then, evaluate the target function on each x_n to get the corresponding output y_n. Finally, flip the labels of $\frac{N}{10}$ randomly selected y_n's and the data set will likely become non separable.

Now, try the pocket algorithm on your data set using $T = 1,000$ iterations. Repeat the experiment 20 times. Then, plot the average $E_{in}(w(t))$ and the average $E_{in}(\hat{w})$(which is also a function of t) on the same figure and see how they behave when t increases. Similarly, use a test set of size 1, 000 and plot a figure to show how $E_{out}(w(t))$ and $E_{out}(\hat{w})$ behave.

首先构造100个线性不可分的点，题目给出的方法是先随意取一条直线（这里我们选择的直线是$y = x$），然后根据这条直线给出$N = 100$个线性可分的点，再随机挑选其中$\frac{N}{10} = 10$个数据，翻转他们的y_n，然后使用pocket PLA进行1000次迭代，画出$E_{in}(w(t))$，以及平均值$E_{in}(\hat{w})$随迭代次数的变化，在这个过程中同样计算1000个测试数据的$E_{out}(w(t))$和平均值 $E_{out}(\hat{w})$ 并作图，这里的测试数据和之前的数据符合的规则应该一致，都是同样有10%的数据经过翻转。

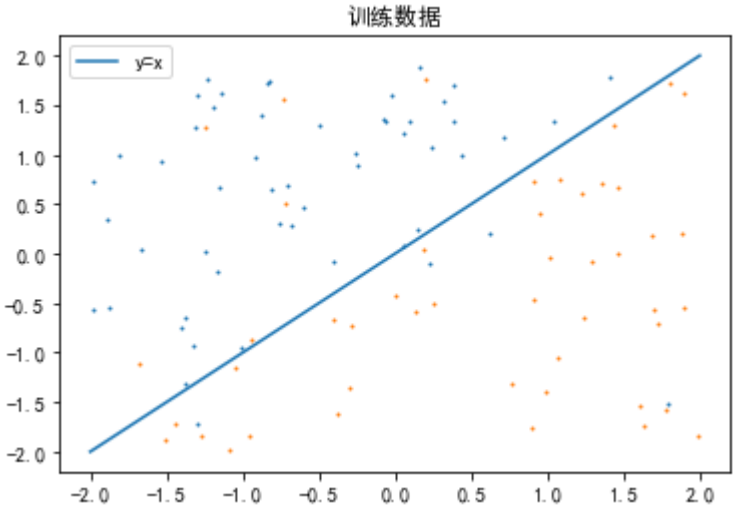
```
import numpy as np

#产生n组数据，10%为噪声，这里直线选择为y=x
def generatedata(n):
    #记录全部数据
    Data=[]
```

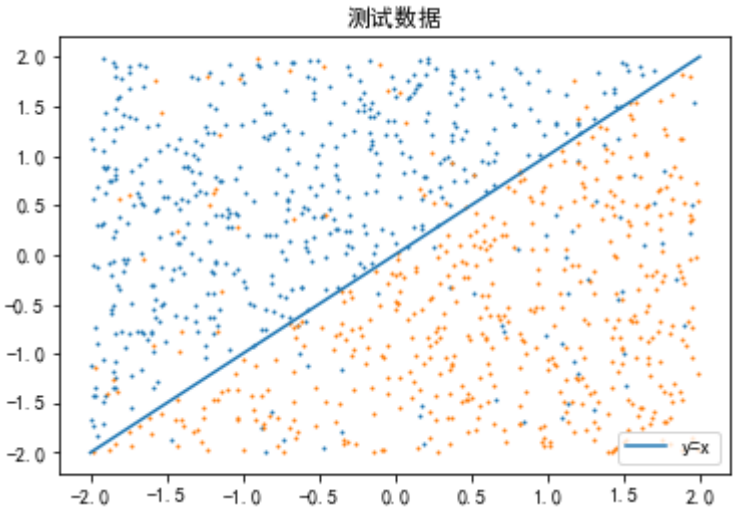
```
    for i in range(n):
        data=np.array(1)
        data=np.append(data,np.random.uniform(-2,2,2))
        if data[2]>data[1]:
            data=np.append(data,1)
        else:
            data=np.append(data,-1)
        #我们将前n/10个数据修改为噪声，最后再打乱数据
        if i<n/10:
            data[-1]*=-1
        Data.append(data)
    np.random.shuffle(Data)
    return Data
```

产生100个以及1000的点，然后分别作图看看。

```
import matplotlib.pylab as plt
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

#生成训练数据
train=generatedata(100)
#后面两步是为了作图
#标签为+1的点
trainpx=[i[1] for i in train if i[-1]>0]
trainpy=[i[2] for i in train if i[-1]>0]
#标签为-1的点
trainnx=[i[1] for i in train if i[-1]<0]
trainny=[i[2] for i in train if i[-1]<0]

#生成测试数据
test=generatedata(1000)
#标签为+1的点
testpx=[i[1] for i in test if i[-1]>0]
testpy=[i[2] for i in test if i[-1]>0]
#标签为-1的点
testnx=[i[1] for i in test if i[-1]<0]
testny=[i[2] for i in test if i[-1]<0]

x=[-2,2]
y=[-2,2]

#训练数据
plt.scatter(trainpx,trainpy,s=1)
plt.scatter(trainnx,trainny,s=1)
plt.title('训练数据')
plt.plot(x,y,label='y=x')
plt.legend()
plt.show()

#测试数据
plt.scatter(testpx,testpy,s=1)
```

```
plt.scatter(testnx,testny,s=1)
plt.title('测试数据')
plt.plot(x,y,label='y=x')
plt.legend()
plt.show()
```





接着定义Pocket PLA

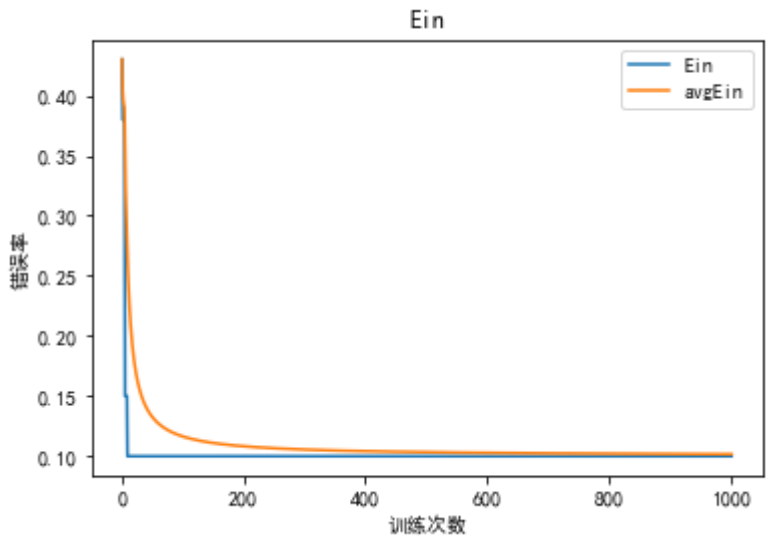
```
#定义sign函数
def sign(x):
    if x>0:
        return 1
    else:
        return -1

#定义计算错误个数的函数,n为数据维度
def CountError(x,w,n):
    count=0
    for i in x:
        if sign(i[:n].dot(w))*i[-1]<0:
```

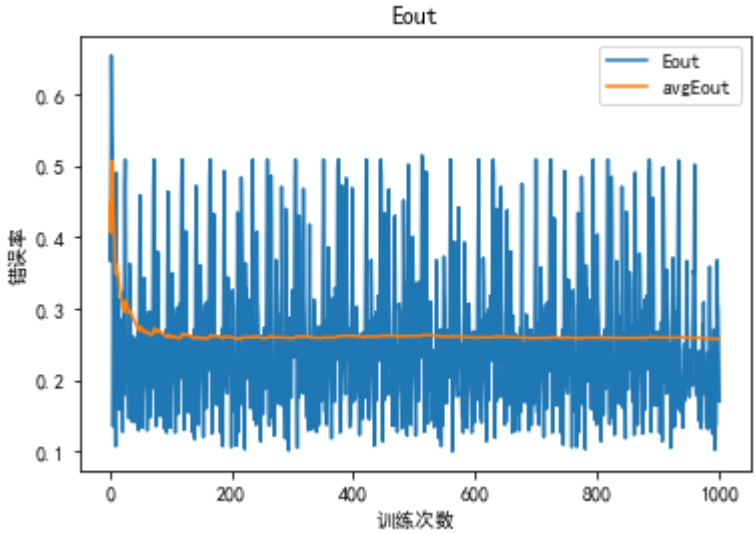
```
            count+=1
    return count

#定义PocketPLA,k为步长,max为最大更新次数
def PocketPLA(train,test,k,maxnum):
    #n为数据维度,m为数据数量
    m=len(train)
    n=len(train[0])-1
    #记录过程中的全部w
    W=[]
    #Ein
    Ein=np.array([])
    #Eout
    Eout=np.array([])
    #初始化向量
    w=np.zeros(n)
    #错误率最小的向量
    w0=np.zeros(n)
    #记录次数
    t=0
    error=CountError(train,w,n)
    if error==0:
        pass
    else:
        #记录取哪个元素
        j=0
        while (t<maxnum or error==0):
            #记录是否更新过
            flag=0
            i=train[j]
            #print(error)
            if sign(i[:n].dot(w))*i[-1]<0:
                w+=k*i[-1]*i[:n]
                t+=1
                flag=1
            error1=CountError(train,w,n)
            if error>error1:
                w0=w[:]
                error=error1
            j+=1
            if(j>=m):
                j=j%m
            #更新才记录数据
            if flag==1:
                Ein=np.append(Ein,error/m)
                Eout=np.append(Eout,CountError(test,w0,n)/len(test))
                W.append(w0)
    return Ein,Eout,W
```

我们看下训练结果

```
#n为训练次数,k为步长
```

```
def show(n,k):
    #计算Ein,Eout
    Ein,Eout,W=PocketPLA(train,test,k,n)

    #计算平均值
    t=np.arange(1,n+1)
    avgEin=np.cumsum(Ein)/t
    avgEout=np.cumsum(Eout)/t

    #Ein作图
    plt.plot(t,Ein,label='Ein')
    plt.plot(t,avgEin,label='avgEin')
    plt.title('Ein')
    plt.xlabel('训练次数')
    plt.ylabel('错误率')
    plt.legend()
    plt.show()

    #Eout作图
    plt.plot(t,Eout,label='Eout')
    plt.plot(t,avgEout,label='avgEout')
    plt.title('Eout')
    plt.xlabel('训练次数')
    plt.ylabel('错误率')
    plt.legend()
    plt.show()

show(1000,1)
```





可以看到$E_{in}(w(t)), E_{in}(\hat{w}), E_{out}(\hat{w})$随着训练次数增加逐渐减少，而$E_{out}(w(t))$随着训练次数增加则波动较大。

Exercise 3.3 (Page 87)

Consider the hat matrix $H = X(X^TX)^{-1}X^T$, where X is an N by $d+1$ matrix, and X^TX is invertible.

(a) Show that H is symmetric.

(b) Show that$H^K = H$ for any positive integer K.

(c) If I is the identity matrix of size N, show that $(I-H)^K = I-H$ for any positive integer K.

(d) Show that $trace(H) = d+1$, where the trace is the sum of diagonal elements. [Hint: $trace(AB) = trace(BA)$]

(a)证明H是对称矩阵

$$\begin{aligned} H^T &= (X(X^TX)^{-1}X^T)^T \\ &= X((X^TX)^{-1})^TX^T \\ &= X((X^TX)^T)^{-1}X^T \\ &= X(X^TX)^{-1}X^T \\ &= H \end{aligned}$$

(b)证明$H^K = H$,直接验证即可，先来看$K=2$的情形

$$\begin{aligned} H^2 &= X(X^TX)^{-1}X^TX(X^TX)^{-1}X^T \\ &= X(X^TX)^{-1}X^T \\ &= H \end{aligned}$$

那么对于任意K

$$\begin{aligned} H^K &= H^2H^{K-2} \\ &= HH^{K-2} \\ &= H^{K-1} \\ &= \dots \\ &= H \end{aligned}$$

(c)利用二项式定理直接打开验证

$$
\begin{aligned}
(I-H)^K &= \sum_{i=0}^{i=K} C_K^i I^{K-i}(-H)^i \\
&= \sum_{i=0}^{i=K} C_K^i (-1)^i H^i \\
&= I + H\sum_{i=1}^{i=K} C_K^i (-1)^i (\text{注意 } H^i = H) \\
&= I + H[(1-1)^K - 1] \\
&= I - H
\end{aligned}
$$

(d)利用迹(trace)的性质$trace(AB) = trace(BA)$

$$
\begin{aligned}
trace(H) &= trace(X(X^TX)^{-1}X^T) \\
&= trace(X^TX(X^TX)^{-1}) \\
&= trace(I_{d+1}) (\text{注意 } H^TH \text{为 } (d+1)\times(d+1) \text{阶矩阵}) \\
&= d+1
\end{aligned}
$$

Exercise 3.4 (Page 88)

Consider a noisy target $y = w^{*T}x + \epsilon$ for generating the data, where ϵ is a noise term with zero mean and σ^2 variance, independently generated for every example (x, y). The expected error of the best possible linear fit to this target is thus σ^2 .

For the data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, denote the noise in yn as ϵ_n and let $\epsilon = [\epsilon_1, \epsilon_2, \ldots, \epsilon_N]^T$; assume that X^TX is invertible. By fllowing the steps below, show that the expected in sample error of linear regression with respect to $\mathcal{D}$ is given by .

$$
E_{\mathcal{D}}[E_{in}(w_{lin})] = \sigma^2(1 - \frac{d+1}{N})
$$

(a) Show that the in sample estimate of is given by $\hat{y} = Xw^* + H\epsilon$.

(b) Show that the in sample error vector $\hat{y} - y$ can be expressed by a matrix times ϵ. What is the matrix?

(c) Express $E_{in}(w_{lin})$ in terms of ϵ using (b), and simplify the expression using Exercise 3.3(c).

(d) Prove that $E_{\mathcal{D}}[E_{in}(w_{lin})] = \sigma^2(1 - \frac{d+1}{N})$ using (c) and the independence of $\epsilon_1, \epsilon_2, \ldots, \epsilon_N$. [Hint: The sum of the diagonal elements of a matrix (the trace) wil play a role. See Exercise 3.3]

For the expected out of sample error, we take a special case which is easy to analyze. Consider a test data set $\mathcal{D}_{test} = \{(x_1, y_1^{'}), \ldots, (x_N, y_N^{'})\}$ which shares the same input vectors x_n with $\mathcal{D}$ but with a different realization of the noise terms. Denote the noise in $y_n^{'}$ as $\epsilon_n^{'}$and let $\epsilon^{'} = [\epsilon_1^{'}, \epsilon_2^{'}, \ldots, \epsilon_N^{'}]^T$. Define Etest(W!in) to be the average squared error on $\mathcal{D}_{test}$.

(e) Prove that $E_{\mathcal{D},\epsilon^{'}}[E_{in}(w_{lin})] = \sigma^2(1 + \frac{d+1}{N})$.

The special test error E_{test} is a very restricted case of the general out-of sample error. Some detailed analysis shows that similar results can be obtained for the general case, as shown in Problem 3.1.

(a)首先将$y = w^{*T}x + \epsilon$改写为向量的形式，记$y = [y_1 \ldots y_N]^T, X = [x_1 \ldots x_N]^T$，注意题目中给出 $\epsilon = [\epsilon_1, \epsilon_2, \ldots, \epsilon_N]^T$

那么

$$y = Xw^* + \epsilon$$

我们知道$w_{lin} = (X^TX)^{-1}X^Ty$

那么

$$\begin{aligned}\hat{y} &= Xw_{lin} \\ &= X(X^TX)^{-1}X^Ty \\ &= X(X^TX)^{-1}X^T(Xw^* + \epsilon) \\ &= X(X^TX)^{-1}X^TXw^* + X(X^TX)^{-1}X^T\epsilon \\ &= Xw^* + H\epsilon\end{aligned}$$

其中$H = X(X^TX)^{-1}X^T$的定义来自于Exercise 3.3

(b)直接计算即可

$$\begin{aligned}\hat{y} - y &= Xw^* + H\epsilon - (Xw^* + \epsilon) \\ &= (H - I)\epsilon\end{aligned}$$

(c)直接计算即可，注意要用到Exercise 3.3证明的性质

$$\begin{aligned}E_{in}(w_{lin}) &= \frac{1}{N}||\hat{y} - y||^2 \\ &= \frac{1}{N}||(H - I)\epsilon||^2 \\ &= \frac{1}{N}((H - I)\epsilon)^T((H - I)\epsilon) \\ &= \frac{1}{N}\epsilon^T(H - I)(H - I)\epsilon(\text{注意}H\text{对称}) \\ &= \frac{1}{N}\epsilon^T(I - H)\epsilon(\text{注意}(I - H)^K = I - H) \\ &= \frac{1}{N}\epsilon^T(I - H)\epsilon\end{aligned}$$

(d)这题也是直接计算，注意要用到trace的性质和上题结论

$$\begin{aligned}E_{\mathcal{D}}[E_{in}(w_{lin})] &= \frac{1}{N}E_{\mathcal{D}}(\epsilon^T(I - H)\epsilon) \\ &= \frac{1}{N}E_{\mathcal{D}}trace(\epsilon^T(I - H)\epsilon)(\text{这一步是由于}\epsilon^T(I - H)\epsilon\text{是一个实数，对于实数}a, a = trace(a)) \\ &= \frac{1}{N}E_{\mathcal{D}}trace(\epsilon^T\epsilon - \epsilon^TH\epsilon)) \\ &= \frac{1}{N}[E_{\mathcal{D}}(\sum_{i=1}^{N}\epsilon_i^2) - E_{\mathcal{D}}(\sum_{i=1}^{N}\epsilon_iH_{ii}\epsilon_i)](H_{ii}\text{为}H\text{第}(i,i)\text{个元素}) \\ &= \frac{1}{N}[N\sigma^2 - (\sum_{i=1}^{N}H_{ii})\sigma^2] \\ &= \frac{1}{N}[N\sigma^2 - trace(H)\sigma^2](\text{注意上一题结论}trace(H) = d + 1) \\ &= \frac{1}{N}[N\sigma^2 - (d + 1)\sigma^2] \\ &= \sigma^2(1 - \frac{d + 1}{N})\end{aligned}$$

为了方便解决下一题，我们把上述计算中的两个结果单独列出

$$E_{\mathcal{D}}(\epsilon^T\epsilon)=N\sigma^2$$
$$E_{\mathcal{D}}(\epsilon^TH\epsilon))=(d+1)\sigma^2$$

(e)首先还是改写为向量的形式$y^{'}=[y_1^{'}\dots y_N^{'}]^T,X=[x_1\dots x_N]^T,\epsilon^{'}=[\epsilon_1^{'},\epsilon_2^{'},\dots,\epsilon_N^{'}]^T$，那么

$$y^{'}=Xw^*+\epsilon^{'}$$

那么由(a)的结论知

$$\hat{y}=Xw_{lin}=Xw^*+H\epsilon$$

因此

$$\begin{aligned}E_{test}(w_{lin})&=\frac{1}{N}||\hat{y}-y^{'}||^2\\&=\frac{1}{N}||Xw^*+H\epsilon-(Xw^*+\epsilon^{'})||^2\\&=\frac{1}{N}||H\epsilon-\epsilon^{'}||^2\\&=\frac{1}{N}(H\epsilon-\epsilon^{'})^T(H\epsilon-\epsilon^{'})\\&=\frac{1}{N}(\epsilon^TH-\epsilon^{'T})(H\epsilon-\epsilon^{'})(注意H对称)\\&=\frac{1}{N}(\epsilon^THH\epsilon-2\epsilon^{'T}H\epsilon+\epsilon^{'T}\epsilon^{'})\\&=\frac{1}{N}(\epsilon^TH\epsilon-2\epsilon^{'T}H\epsilon+\epsilon^{'T}\epsilon^{'})(注意H^K=H)\end{aligned}$$

接着我们计算$E_{\mathcal{D},\epsilon'}[E_{in}(w_{lin})]$

$$\begin{aligned}E_{\mathcal{D},\epsilon'}[E_{in}(w_{lin})]&=E_{\mathcal{D},\epsilon'}[\frac{1}{N}(\epsilon^TH\epsilon-2\epsilon^{'T}H\epsilon+\epsilon^{'T}\epsilon^{'})]\\&=\frac{1}{N}[E_{\mathcal{D},\epsilon'}(\epsilon^TH\epsilon)-2E_{\mathcal{D},\epsilon'}(\epsilon^{'T}H\epsilon)+E_{\mathcal{D},\epsilon'}(\epsilon^{'T}\epsilon^{'})]\end{aligned}$$

回顾上一题单独列出的结论

$$E_{\mathcal{D}}(\epsilon^T\epsilon)=N\sigma^2$$
$$E_{\mathcal{D}}(\epsilon^TH\epsilon))=(d+1)\sigma^2$$

因此

$$E_{\mathcal{D},\epsilon'}[E_{in}(w_{lin})]=\sigma^2(1+\frac{d+1}{N})-\frac{2}{N}E_{\mathcal{D},\epsilon'}(\epsilon^{'T}H\epsilon)$$

后面我们单独计算$E_{\mathcal{D},\epsilon'}(\epsilon^{'T}H\epsilon)$，注意$\epsilon_i,\epsilon_i^{'}$独立且$E(\epsilon_i)=E(\epsilon_i^{'})=0$

$$\begin{aligned}E_{\mathcal{D},\epsilon'}(\epsilon^{'T}H\epsilon)&=E_{\mathcal{D},\epsilon'}(trace(\epsilon^{'T}H\epsilon))\\&=E_{\mathcal{D},\epsilon'}(\sum_{i=1}^{N}\epsilon_i^{'}H_{ii}\epsilon_i)\\&=\sum_{i=1}^{N}(E(\epsilon_i^{'})H_{ii}E(\epsilon_i))(由独立性)\\&=0(数学期望为0)\end{aligned}$$

因此

$$E_{\mathcal{D},\epsilon'}\left[E_{in}(w_{lin})\right] = \sigma^2(1+\frac{d+1}{N})$$

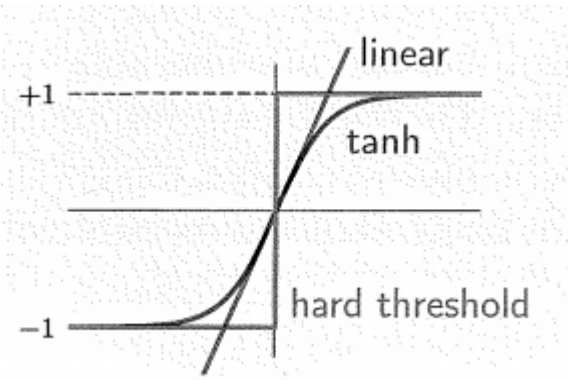
Exercise 3.5 (Page 90)

Another popular soft threshold is the hyperbolic tangent

$$tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

(a) How is $tanh$ related to the logistic function θ? [Hint: shift and scale]

(b) Show that $tanh(s)$ converges to a hard threshold for large $|s|$, and converges to no threshold for small $|s|$ [Hint: Formalize the figure below.]



(a)我们先回顾θ的定义

$$\theta(s) = \frac{e^s}{1+e^s}$$

那么

$$\begin{aligned} tanh(s) &= \frac{e^s - e^{-s}}{e^s + e^{-s}} \\ &= \frac{e^{2s} - 1}{e^{2s} + 1} \\ &= \frac{2e^{2s} - (1+e^{2s})}{e^{2s} + 1} \\ &= 2\frac{e^{2s}}{1+e^{2s}} - 1 \\ &= 2\theta(2s) - 1 \end{aligned}$$

所以$tanh(s)$相当于$\theta(s)$先沿x轴方向压缩2倍，再y轴方向扩张2倍，最右沿y轴向下平移一个单位

(b)这题我不是很理解题目中所说的converges to no threshold for small $|s|$，我看了下论坛上老师的回复以及参考图片，猜测是threshold是水平渐近线的意思，所以我们计算$\frac{tanh(s)}{s}$，这里使用了$Taylor$展开。

$$
\begin{aligned}
limit_{s\to\infty}\frac{tanh(s)}{s} &= limit_{s\to\infty}\frac{1-e^{-2s}}{(1+e^{-2s})s} \\
&= limit_{s\to\infty}\frac{1-(1-2s)}{(1+1-2s)s} \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
limit_{s\to 0}\frac{tanh(s)}{s} &= limit_{s\to 0}\frac{e^{s}-e^{-s}}{e^{s}+e^{-s}} \\
&= limit_{s\to 0}\frac{1+s-(1-s)}{(1+s+(1-s))s} \\
&= limit_{s\to 0}\frac{2}{2} \\
&= 1
\end{aligned}
$$

所以当$|s|\to\infty$，$tanh(s)$有水平渐近线,$|s|\to 0$，$tanh(s)$没有水平渐近线

Exercise 3.6 (Page 92)

[Cross-entropy error measure]

(a) More generally, if we are learning from ± 1 data to predict a noisy target $P(y|x)$ with candidate hypothesis h, show that the maximum likelihood method reduces to the task of finding h that minimizes

$$
E_{in}(w)=\sum_{n=1}^{N}[\![y_n=+1]\!]ln\frac{1}{h(x_n)}+[\![y_n=-1]\!]ln\frac{1}{1-h(x_n)}
$$

(b) For the case $h(x)=\theta(w^Tx)$, argue that minimizing the in sample error in part (a) is equivalent to minimizing the one in (3.9) .

For two probability distributions $p,1-p$ and $q,1-q$ with binary outcomes, the cross entropy (from information theory) is

$$
plog\frac{1}{q}+(1-p)log\frac{1}{1-q}
$$

The in sample error in part (a) corresponds to a cross entropy error measure on the data point (x_n,y_n), with $p=[\![y_n=+1]\!]$ and $q=h(x_n)$.

(a)记最大似然函数为L

$$
\begin{aligned}
L&=\prod_{i=1}^{N}(h(x_n))^{[y_n=+1]}(1-h(x_n))^{[y_n=-1]} \\
lnL&=e^{\sum_{n=1}^{N}[y_n=+1]lnh(x_n)+[y_n=-1]ln(1-h(x_n))}
\end{aligned}
$$

因此要使得L最大，只要使$\sum_{n=1}^{N}[\![y_n=+1]\!]lnh(x_n)+[\![y_n=-1]\!]ln(1-h(x_n))$最大即可，也就是使得$-(\sum_{n=1}^{N}[\![y_n=+1]\!]lnh(x_n)+[\![y_n=-1]\!]ln(1-h(x_n)))$最小，注意以下事实

$$
-(\sum_{n=1}^{N}[\![y_n=+1]\!]lnh(x_n)+[\![y_n=-1]\!]ln(1-h(x_n)))=\sum_{n=1}^{N}[\![y_n=+1]\!]ln\frac{1}{h(x_n)}+[\![y_n=-1]\!]ln\frac{1}{1-h(x_n)}=E_{in}(w)
$$

因此结论成立。

(b)回顾下3.9

$$E_{in}^{'}(w)=\frac{1}{N}\sum_{n=1}^{N}ln(1+e^{-y_nw^Tx_n})$$

注意$\theta(s)=\frac{e^s}{1+e^s}$，那么

$$\theta(-s)=\frac{e^{-s}}{1+e^{-s}}=\frac{1}{1+e^s}=1-\theta(s)$$
$$h(x)=\theta(w^Tx)=\frac{e^{w^Tx}}{1+e^{w^Tx}}$$
$$h(-x)=\theta(w^T(-x))=1-\theta(w^Tx)=1-h(x)$$

我们对这里的$E_{in}(w)$进行一个处理

$$[\![y_n=+1]\!]ln\frac{1}{h(x_n)}=[\![y_n=+1]\!]ln\frac{1}{h(y_nx_n)}$$
$$[\![y_n=-1]\!]ln\frac{1}{1-h(x_n)}=[\![y_n=-1]\!]ln\frac{1}{h(-x_n)}=[\![y_n=-1]\!]ln\frac{1}{h(y_nx_n)}$$

代入我们这里的$E_{in}(w)$

$$\begin{aligned}E_{in}(w)&=\sum_{n=1}^{N}[\![y_n=+1]\!]ln\frac{1}{h(x_n)}+[\![y_n=-1]\!]ln\frac{1}{1-h(x_n)}\\&=\sum_{n=1}^{N}([\![y_n=+1]\!]+[\![y_n=-1]\!])ln\frac{1}{h(y_nx_n)}\\&=\sum_{n=1}^{N}ln\frac{1}{h(y_nx_n)}\\&=\sum_{n=1}^{N}ln(\frac{e^{y_nw^Tx}}{1+e^{y_nw^Tx}})^{-1}\\&=\sum_{n=1}^{N}ln(1+e^{-y_nw^Tx_n})\\&=NE_{in}^{'}(w)\end{aligned}$$

因此最小化$E_{in}(w)$等价于最小化$E_{in}^{'}(w)$

题目最后的意思是这里的结论可以和信息论里的结论类比。

Exercise 3.7 (Page 92)

For logistic regression , show that

$$\begin{aligned}\nabla E_{in}(w)&=-\frac{1}{N}\sum_{n=1}^{N}\frac{y_nx_n}{1+e^{y_nw^Tx_n}}\\&=\frac{1}{N}\sum_{n=1}^{N}-y_nx_n\theta(-y_nw^Tx_n)\end{aligned}$$

Argue that a 'misclassified' example contributes more to the gradient than a correctly classified one.

这题就是对logistic的$E_{in}(w)$求梯度，回顾下$E_{in}(w)$

$$E_{in}(w)=\frac{1}{N}\sum_{n=1}^{N}ln(1+e^{-y_nw^Tx_n})$$

注意$\theta(s)=\frac{e^s}{1+e^s}$

$$\begin{aligned}\frac{\partial E_{in}(w)}{\partial w_i}&=\frac{1}{N}\sum_{n=1}^{N}\frac{ln(1+e^{-y_nw^Tx_n})}{\partial w_i}\\&=\frac{1}{N}\sum_{n=1}^{N}\frac{1}{1+e^{-y_nw^Tx_n}}\frac{\partial e^{-y_nw^Tx_n}}{\partial w_i}\\&=\frac{1}{N}\sum_{n=1}^{N}\frac{1}{1+e^{-y_nw^Tx_n}}(-e^{-y_nw^Tx_n})(y_nx_n^{(i)})(x_n^{(i)}\text{表示}x_n\text{的第}i\text{个分量})\\&=\frac{1}{N}\sum_{n=1}^{N}-y_nx_n^{(i)}\frac{e^{-y_nw^Tx_n}}{1+e^{-y_nw^Tx_n}}\\&=\frac{1}{N}\sum_{n=1}^{N}-y_nx_n^{(i)}\frac{1}{1+e^{y_nw^Tx_n}}\\&=\frac{1}{N}\sum_{n=1}^{N}-y_nx_n^{(i)}\theta(-y_nw^Tx_n)\end{aligned}$$

因此

$$\begin{aligned}\nabla E_{in}(w)&=-\frac{1}{N}\sum_{n=1}^{N}\frac{y_nx_n}{1+e^{y_nw^Tx_n}}\\&=\frac{1}{N}\sum_{n=1}^{N}-y_nx_n\theta(-y_nw^Tx_n)\end{aligned}$$

当一个样例错分时，$y_nw^Tx_n<0$，对应的$\theta(-y_nw^Tx_n)>\frac{1}{2}$；当一个样例分类正确时，$y_nw^Tx_n>0$对应的$\theta(-y_nw^Tx_n)<\frac{1}{2}$，因此错分的样例的权重比正确的样例要大。

Exercise 3.8 (Page 94)

The claim that $\hat{v}$ is the direction which gives largest decrease in E_{in} only holds for small η. Why?

回顾下书上之前的叙述

$$\begin{aligned}\Delta E_{in}&=E_{in}(w(0)+\eta\hat{v})-E_{in}(w(0))\\&=\eta\nabla E_{in}(w(0))^T\hat{v}+O(\eta^2)\\&\geq\eta||\nabla E_{in}(w(0))||\\\hat{v}&=-\frac{\nabla E_{in}(w(0))}{||\nabla E_{in}(w(0))||}\text{时等号成立}\end{aligned}$$

这个式子是泰勒展开，所以$\eta\hat{v}$的模不能太大，如果太大，泰勒展开的偏差会很大。$\hat{v}$是单位向量，因此$||\eta\hat{v}||=\eta$，所以上述式子只有当η较小的时候才有效。

Exercise 3.9 (Page 97)

Consider pointwise error measures $e_{class}(s,y)=[\![y\neq sign(s)]\!]$,$e_{sq}(s,y)=(y-s)^2$,and $e_{log}(s,y)=ln(1+exp(-ys))$,where the signal $s=w^Tx$

(a) For $y=+1$, plot e_{class}, e_{sq} and $\frac{1}{ln2}e_{log}$ versus s, on the same plot.

(b) Show that $e_{class}(s,y)\leq e_{sq}(s,y)$, and hence that the classification error is upper bounded by the squared error.

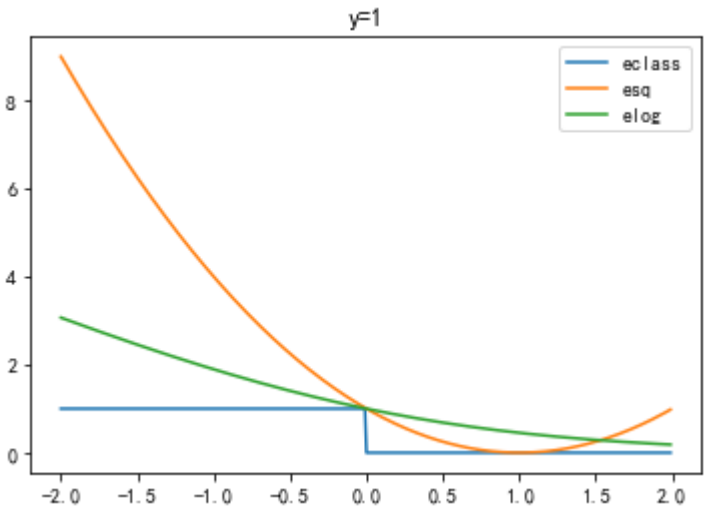
(c) Show that $e_{class}(s,y)\leq \frac{1}{ln2}e_{log}(s,y)$,and, as in part (b), get an upper bound (up to a constant factor) using the logistic regression error. These bounds indicate that minimizing the squared or logistic regression error should also decrease the classification error, which justifies using the weights returned by linear or logistic regression as approximations for classification.

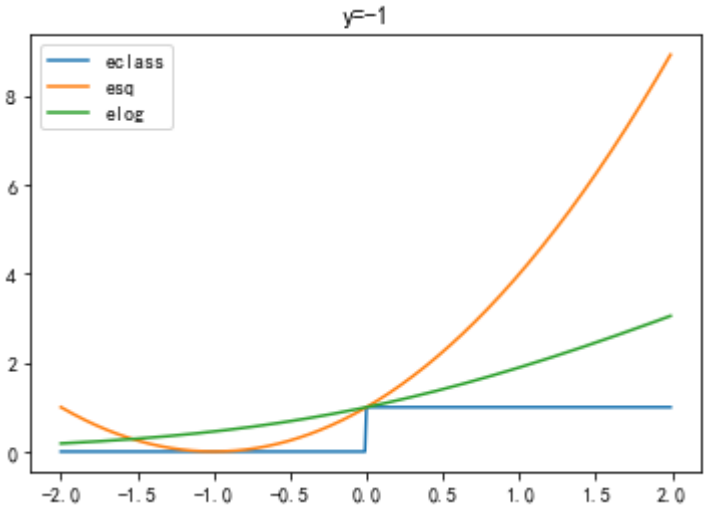
(a),(b),(c)三题作图即可

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

def Class(s,y):
    if s*y>0:
        return 0
    else:
        return 1

def Sq(s,y):
    return (s-y)**2

def Log(s,y):
    return np.log(1+np.exp(-y*s))
```

```
#构造点
x=np.arange(-2,2,0.01)

#y=1
eclass1=[Class(i,1) for i in x]
esq1=[Sq(i,1) for i in x]
elog1=[Log(i,1)/np.log(2) for i in x]

#y=-1
eclass2=[Class(i,-1) for i in x]
esq2=[Sq(i,-1) for i in x]
elog2=[Log(i,-1)/np.log(2) for i in x]

plt.plot(x,eclass1,label='eclass')
plt.plot(x,esq1,label='esq')
plt.plot(x,elog1,label='elog')
plt.title('y=1')
plt.legend()
plt.show()
```

```
plt.plot(x,eclass2,label='eclass')
plt.plot(x,esq2,label='esq')
plt.plot(x,elog2,label='elog')
plt.title('y=-1')
plt.legend()
plt.show()
```





从图像中我们看出， e_{sq} 和 $\frac{1}{ln2}e_{log}$ 都是e_{class}的上界，因此我们可以用线性回归或者logistic回归计算结果，再用产生的结果喂给PLA。

Exercise 3.10 (Page 98)

(a)Define an error for a single data point (x_n, y_n) to be

$$e_n(w) = max(0, -y_n w^T x_n)$$

Argue that PLA can be viewed as SGD on e_n with learning rate $\eta = 1$.

(b) For logistic regression with a very large w, argue that minimizing E_{in} using SGD is similar to PLA. This is another indication that the logistic regression weights can be used as a good approximation for classification .

(a)$e_n(w) = max(0, -y_n w^T x_n)$的意思是对于分类正确的点$e_n(w) = 0$，对于分类不正确的点$e_n(w) = -y_n w^T x_n$，我们来求梯度

$$\frac{\partial(-y_n w^T x_n)}{\partial w_i} = -y_n x_n^{(i)} \left(x_n^{(i)}\text{表示}x_n\text{的第}i\text{个分量}\right)$$

$$\nabla(-y_n w^T x_n) = -y_n x_n$$

所以对于分类错误的点(x_n, y_n)，根据SGD,更新规则为

$$w(t+1) = w(t) + \eta(-\nabla(-y_n w^T x_n)) = w(t) + \eta y_n x_n$$

所以PLA可以被看成$e_n(w) = max(0, -y_n w^T x_n)$的SGD且$\eta = 1$的情形

(b)我们知道logistic的梯度公式有如下形式

$$\nabla E_{in}(w) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n x_n}{1 + e^{y_n w^T x_n}}$$

那么对于SGD梯度即为

$$\nabla e_{in}(w) = \frac{-y_n x_n}{1 + e^{y_n w^T x_n}}$$

带入更新规则$w \leftarrow w - \eta \nabla e_{in}(w)$

$$w(t+1) = w(t) + \frac{\eta y_n x_n}{1 + e^{y_n w^T x_n}}$$

我们注意更新的点一定是错误的，即$y_n w^T x_n < 0$，那么当w非常大时，$e^{y_n w^T x_n} \approx 0$

因此对于非常大的w，上述更新规则可以近似为

$$w(t+1) = w(t) + \eta y_n x_n$$

和PLA一致，这也从另一个角度说明了logistic是分类问题的一个近似

Exercise 3.11 (Page 101)

Consider the feature transform ϕ in (3.12). What kind of boundary in $\mathcal{X}$ does a hyperplane $\hat{w}$ in $\mathcal{Z}$ correspond to in the following cases?

Draw a picture that illustrates an example of each case.

(a) $\hat{w}_1 > 0, \hat{w}_2 < 0$

(b) $\hat{w}_1 > 0, \hat{w}_2 = 0$

(c) $\hat{w}_1 > 0, \hat{w}_2 > 0, \hat{w}_0 < 0$

(d) $\hat{w}_1 > 0, \hat{w}_2 > 0, \hat{w}_0 > 0$

回顾下3.12

$$\phi(x) = (1, x_1^2, x_2^2)$$

因此对应方程为

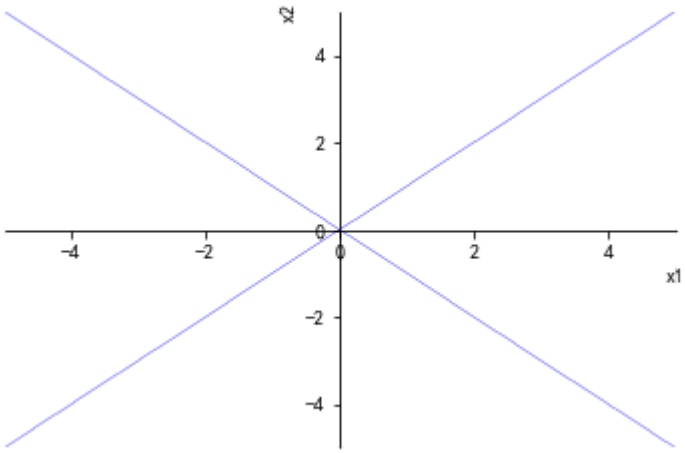
$$\hat{w}_0 + \hat{w}_1 x_1^2 + \hat{w}_2 x_2^2 = 0$$

后面的叙述实际上是高中解析几何的知识。

(a)有三种可能，分别是$\hat{w}_0 = 0, \hat{w}_0 > 0, \hat{w}_0 < 0$

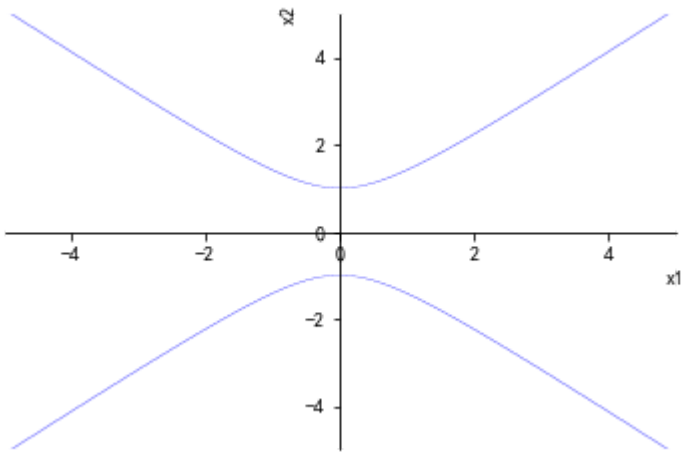
```
from sympy.parsing.sympy_parser import parse_expr
from sympy import plot_implicit
ezplot = lambda exper: plot_implicit(parse_expr(exper))
```

```
ezplot('x1**2-x2**2')
```



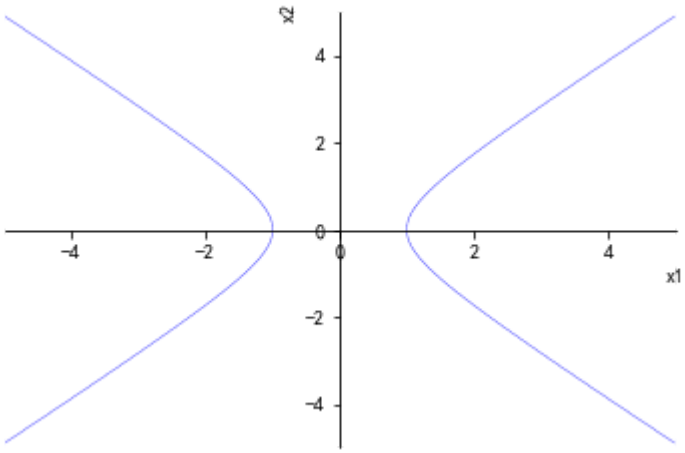

```
<sympy.plotting.plot.Plot at 0x20091b74710>
```

```
ezplot('1+x1**2-x2**2')
```



```
<sympy.plotting.plot.Plot at 0x20091abb5f8>
```

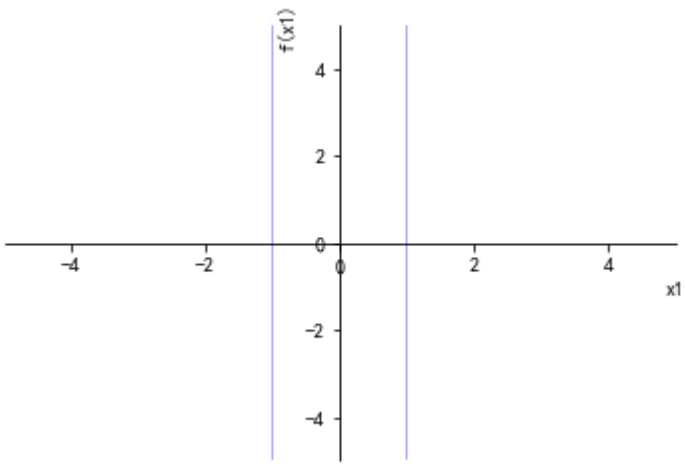
```
ezplot('-1+x1**2-x2**2')
```




```
<sympy.plotting.plot.Plot at 0x20091ca0dd8>
```

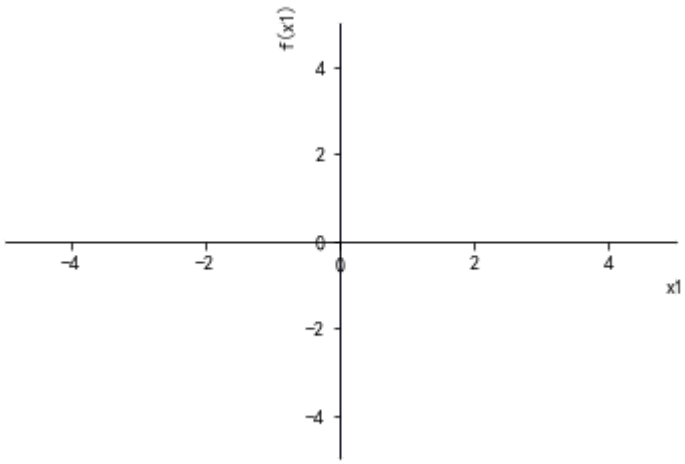
(b)有三种可能，分别是$\hat{w}_0 = 0, \hat{w}_0 > 0, \hat{w}_0 < 0$

```
ezplot('-1+x1**2')
```



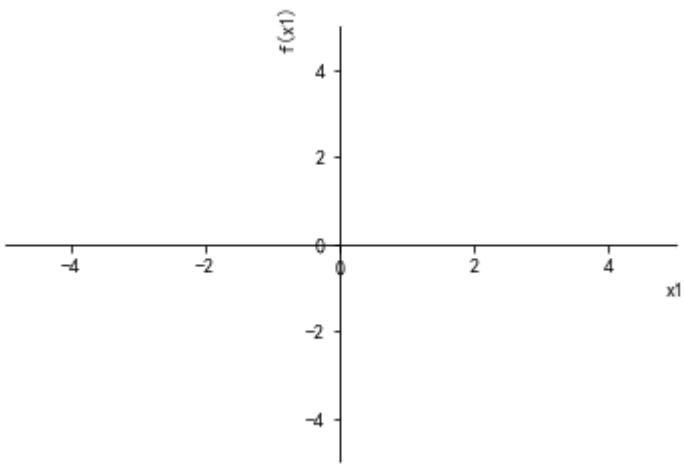
```
<sympy.plotting.plot.Plot at 0x20091cbd828>
```

```
ezplot('0+x1**2')
```




```
<sympy.plotting.plot.Plot at 0x20091d04080>
```

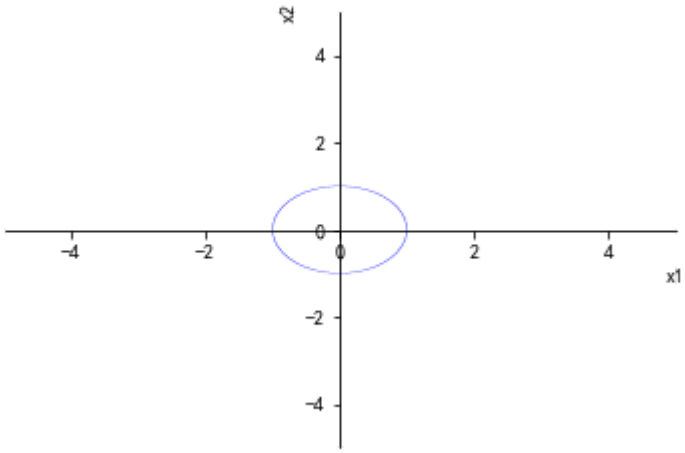
```
ezplot('1+x1**2')
```



```
<sympy.plotting.plot.Plot at 0x2008edd0208>
```

(c)条件已经定死

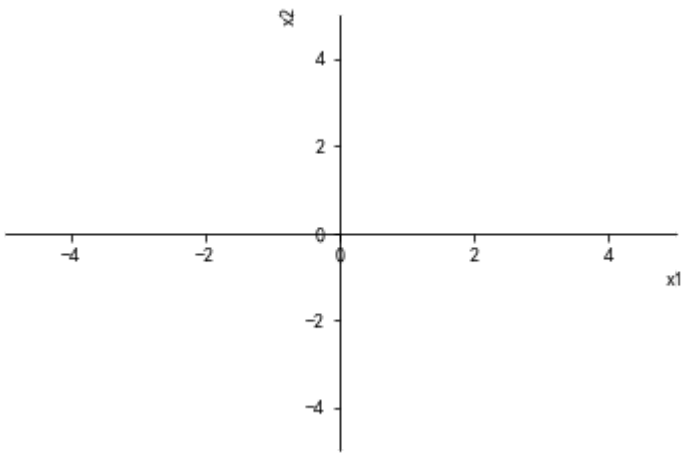
```
ezplot('-1+x1**2+x2**2')
```




```
<sympy.plotting.plot.Plot at 0x20091f64e48>
```

(d)条件已经定死

```
ezplot('1+x1**2+x2**2')
```



```
<sympy.plotting.plot.Plot at 0x20091cbd160>
```

Exercise 3.12 (Page 103)

We know that in the Euclidean plane, the perceptron model $\mathcal{H}$ cannot implement all 16 dichotomies on 4 points. That is, $m_{\mathcal{H}}(4) < 16$. Take the feature transform ϕ in (3.12).

(a) Show that $m_{\mathcal{H}\phi}(3) = 8$.

(b) Show that $m_{\mathcal{H}\phi}(4) < 16$.

(c) Show that $m_{\mathcal{H} \bigcup \mathcal{H}\phi}(4) = 16$.

That is, if you used lines, $d_{vc} = 3$; if you used elipses, $d_{vc} = 3$; if you used lines and elipses, $d_{vc} > 3$.

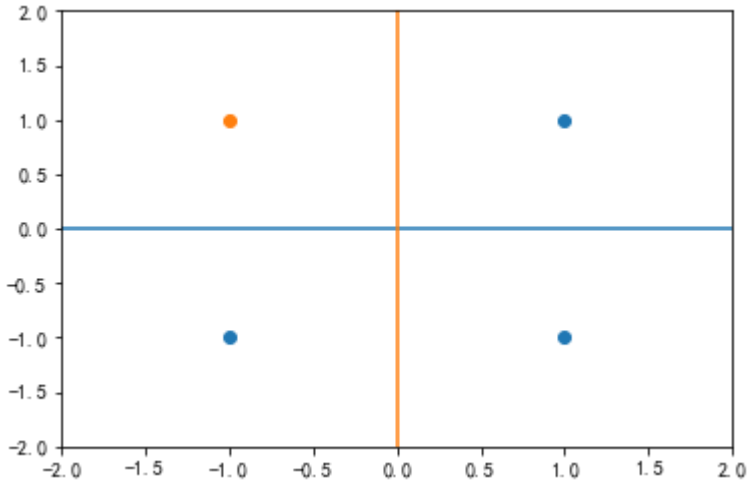
回顾下3.12

$$\phi(x) = (1, x_1^2, x_2^2)$$

(a)作图即可，这里画图比较麻烦，略去了

(b)我们找三个特殊的点

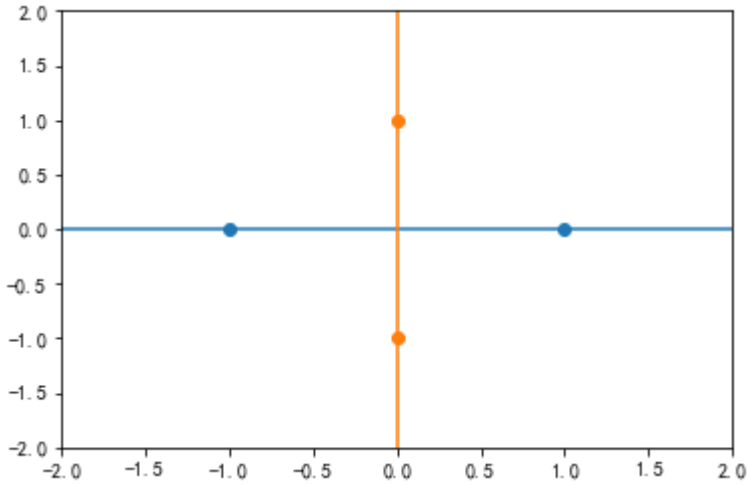
```
import matplotlib.pyplot as plt

x=[-1,1,1]
y=[-1,-1,1]
plt.scatter(x,y)
plt.scatter(-1,1)
plt.xlim(-2,2)
plt.ylim(-2,2)
plt.plot([-2,2],[0,0])
plt.plot([0,0],[-2,2])
plt.xticks()
plt.show()
```



我们用上一题给出的几种二次曲线图形去匹配，可以发现无法区分这种形式。

(c)这里只列一种我们我们之前感知机无法表示的情况

```
import matplotlib.pyplot as plt

plt.scatter([-1,1],[0,0])
plt.scatter([0,0],[-1,1])
plt.xlim(-2,2)
plt.ylim(-2,2)
plt.plot([-2,2],[0,0])
plt.plot([0,0],[-2,2])
plt.xticks()
plt.show()
```



这种形式可以用双曲线进行划分，其余的情形用直线可以轻松划分出来，这里不列出来了。

Exercise 3.13 (Page 104)

Consider the feature transform $z = \phi_2(x)$ in (3.13). How can we use a hyperplane $\hat{w}$ in $\mathcal{Z}$ to represent the following boundaries in $\mathcal{X}$

(a) parabola$(x_1 - 3)^2 + x_2 = 1$

(b) The circle $(x_1-3)^2+(x_2-4)^2=1$

(c) The ellipse $2(x_1-3)^2+(x_2-4)^2=1$

(d) The hyperbola $(x_1-3)^2-(x_2-4)^2=1$

(e) The elipse $2(x_1+x_2-3)^2+(x_1-x_2-4)^2=1$

(f) line $2x_1+x_2=1$

回顾3.13

$$\phi_2(x)=(1,x_1,x_2,x_1^2,x_1x_2,x_2^2)$$

接下来分别打开上述6个式子即可

(a)

$$\begin{gathered}(x_1-3)^2+x_2=1\\x_1^2-6x_1+9+x_2-1=0\\8-6x_1+x_2+x_1^2=0\\\hat{w}=(8,-6,1,1,0,0)\end{gathered}$$

(b)

$$\begin{gathered}(x_1-3)^2+(x_2-4)^2=1\\x_1^2-6x_1+x_2^2-8x_2+24=0\\\hat{w}=(24,-6,-8,1,0,1)\end{gathered}$$

(c)

$$\begin{gathered}2(x_1-3)^2+(x_2-4)^2=1\\2(x_1^2-6x_1+9)+(x_2^2-8x_2+16)-1=0\\2x_1^2-12x_1+x_2^2-8x_2+33=0\\\hat{w}=(33,-12,-8,2,0,1)\end{gathered}$$

(d)

$$\begin{gathered}(x_1-3)^2-(x_2-4)^2=1\\x_1^2-6x_1+9-(x_2^2-8x_2+16)-1=0\\x_1^2-x_2^2+8x_2-6x_1-8=0\\\hat{w}=(-8,-6,8,1,0,-1)\end{gathered}$$

(e)

$$\begin{gathered}2(x_1+x_2-3)^2+(x_1-x_2-4)^2=1\\2[(x_1+x_2)^2+9-6(x_1+x_2)]+(x_1-x_2)^2-8(x_1-x_2)+16-1=0\\2(x_1^2+x_2^2+2x_1x_2+9-6x_1-6x_2)+x_1^2+x_2^2-2x_1x_2-8x_1+8x_2+15=0\\3x_1^2+3x_2^2+33-20x_1-4x_2+2x_1x_2=0\\\hat{w}=(33,-20,-4,3,2,3)\end{gathered}$$

(f)

$$2x_1 + x_2 = 1$$
$$2x_1 + x_2 - 1 = 0$$
$$\hat{w} = (-1, 2, 1, 0, 0, 0)$$

Exercise 3.14 (Page 105)

Consider the Qth order polynomial transform ϕ_Q for $\mathcal{X} = R^d$. What is the dimensionality $\tilde{d}$ of the feature space $\mathcal{Z}$ (excluding the fixed coordinate $z_0 = 1$). Evaluate your result on $d \in \{2, 3, 5, 1\}$and $Q \in \{2, 3, 5, 1\}$.

注意设$x = (x_1 \dots x_d)$，那么多项式转换的一般形式为$\prod_{i=1}^{d} x_i^{n_i}$，那么$Q$次多项式相当于对此加了一个条件($z_0 = 1$不算在内)

$$1 \leq \sum_{i=1}^{d} n_i \leq Q(n_i \geq 0)$$

我们记$\sum_{i=1}^{d} n_i = q(n_i \geq 0)$的解的数量为$f(q)$，那么$1 \leq \sum_{i=1}^{d} n_i \leq Q(n_i \geq 0)$的解的数量为

$$\sum_{q=1}^{Q} f(q)$$

我们接下来求解$f(q)$，对式子稍做变形

$$\sum_{i=1}^{d} n_i = q(n_i \geq 0)$$
$$\sum_{i=1}^{d} (n_i + 1) = q + d(n_i \geq 0)$$

令$n_i + 1 = m_i$，那么上式可化为

$$\sum_{i=1}^{d} (m_i) = q + d(m_i \geq 1)$$

所以求正整数解即可，隔板法可得

$$f(q) = C_{q+d-1}^{d-1}$$

因此

$$\hat{h} = \sum_{q=1}^{Q} f(q) = \sum_{q=1}^{Q} C_{q+d-1}^{d-1}$$

我们将$d = 2$带入

$$\hat{h} = \sum_{q=1}^{Q} f(q) = \sum_{q=1}^{Q} C_{q+1}^{1} = 2 + \dots (Q+1) = \frac{Q(Q+3)}{2}$$

符合课本104页的叙述

Exercise 3.15 (Page 106)

High-dimensional feature transfrms are by no means the only transforms that we can use. We can take the tradeoff in the other direction, and use low dimensional feature transforms as well (to achieve an even lower generalization error bar). Consider the following fature transform, which maps a d-dimensional x to a one-dimensional z, keeping only the kth coordinate of x.

$$\phi_{(k)}(x) = (1, x_k)$$

Let $\mathcal{H}_k$ be the set of perceptrons in the feature space.

(a) Prove that $d_{vc}(\mathcal{H}_k) = 2$.

(b) Prove that $d_{vc}(\bigcup_{k=1}^{d} \mathcal{H}_k) \leq 2(log_2 d + 1)$.

$\mathcal{H}_k$ is called the decision stump model on dimension k.

(a)这个比较简单，$\mathcal{H}_k$是特征空间里的感知机，并且特征空间的维度为1，因此根据感知机的性质，我们知道$d_{vc}(\mathcal{H}_k) = 1 + 1 = 2$

(b)我们来看下$\mathcal{H}_k$的具体形式，设参数为(w_0, w_1)，那么对应的边界平面为

$$w_0 + w_1 x_k = 0$$
$$x_k = -\frac{w_0}{w_1}$$

因此$\mathcal{H}_k$划分方法可以理解为看第k个下标，如果x_k大于阈值$-\frac{w_0}{w_1}$，标记为1，反之标记为-1，或者反过来(大于$-\frac{w_0}{w_1}$标记为-1,小于$-\frac{w_0}{w_1}$标记为1)。

假设现在有N个点，现在来计算$\mathcal{H}_k$能区分的数量，先对这N个点的第k个坐标排序，**先不管全1或者全-1的两种情况**，那么$\mathcal{H}_k$相当于在这N个x_k的$N-1$个间隔挑选，一共可以有$N-1$种选择，那么由于大于阈值可以为1，也可以为-1，所以一共可以区分$2(N-1)$种情形，因此

$$除去全1或者全-1的情况，每个\mathcal{H}_k可以区分2N-2种情形$$

那么$\bigcup_{k=1}^{d} \mathcal{H}_k$一共可以表示$f(N) = 2(N-1) \times d + 2$种情形，注意我们这里为了更加准确，全1或者全-1的情形合并在一起统计了。当$N = 2(log_2 d + 1)$时

$$\begin{aligned} f(N) &= 2(N-1) \times d + 2 \\ &= 2(2log_2 d + 1) \times d + 2 \end{aligned}$$

我们来证明

$$f(N) = 2(2log_2 d + 1) \times d + 2 \leq 2^N = 2^{2(log_2 d + 1)} = 4d^2$$

接着我们来进行一些处理

$$\begin{aligned} 2(2log_2 d + 1) \times d + 2 \leq 4d^2 \Leftrightarrow \\ 2d(2log_2 d + 1) \leq 4d^2 - 2 \Leftrightarrow \\ 2log_2 d + 1 \leq 2d - \frac{1}{d} \Leftrightarrow \\ 2d - \frac{1}{d} - 2log_2 d - 1 \geq 0 \end{aligned}$$

记$g(d) = 2d - \frac{1}{d} - 2log_2 d - 1$，求导得

$$g^{'}(d) = 2 + \frac{1}{d^2} - \frac{2}{dln2}$$

$$g^{'}(d) = (\frac{1}{d} - \frac{1}{ln2})^2 + 2 - (\frac{1}{ln2})^2$$

将$(\frac{1}{d} - \frac{1}{ln2})^2 + 2 - (\frac{1}{ln2})^2$看成关于$\frac{1}{d}$的二次函数，由二次函数的性质可得

$$g^{'}(d) \geq g^{'}(1) = 3 - \frac{2}{ln2} > 0$$

所以$g(d)$在$[1, +\infty)$递增递增，$g(d) \geq g(1) = 0$

因此

$$f(N) \leq 2^N (N = 2(log_2 d + 1))$$

从而可得

$$d_{vc}(\bigcup_{k=1}^{d} \mathcal{H}_k) \leq 2(log_2 d + 1)$$

这是因为在$N = 2(log_2 d + 1)$表示的种类数量小于等于2^N，所以最多shatter $2(log_2 d + 1)$个点。

Exercise 3.16 (Page 106)

Write down the steps of the algorithm that combines ϕ_3 with linear regression. How about using ϕ_{10} instead? Where is the main computational bottleneck of the resulting algorithm?

这部分可以参考课本86页。我们直接对ϕ_k进行总结,记特征空间的维度为$\tilde{d}$，原始数据为$(x_1 \dots x_N), (y_1 \dots y_N)$。

第一步进行特征转换，记得到的新的数据为$(\tilde{x}_1 \dots \tilde{x}_N)$，构成的矩阵为$\widetilde{X}$

第二步计算$(\widetilde{X}^T \widetilde{X})^{-1} \widetilde{X}^T$

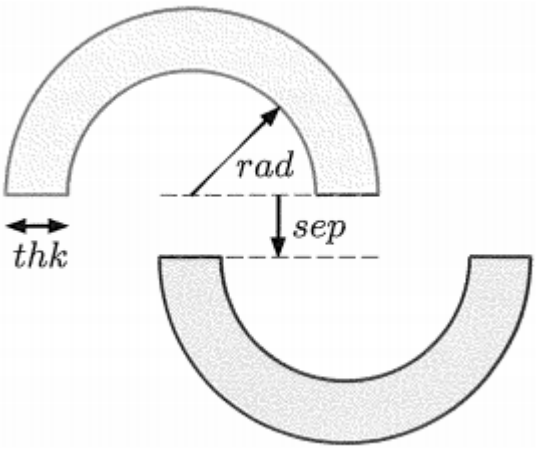
第三步计算$(\widetilde{X}^T \widetilde{X})^{-1} \widetilde{X}^T y$

主要计算时间应该是第二部求逆矩阵。

Part 2:Problems

Problem 3.1 (Page 109)

Consider the double semi-circle "toy" learning task below.



There are two semi circles of width thk with inner radius rad, separated by sep as shown (red is -1 and blue is +1). The center of the top semi circle is aligned with the middle of the edge of the bottom semi circle. This task is linearly separable when $sep \geq 0$, and not so for $sep < 0$. Set $rad = 10, thk = 5$ and $sep = 5$. Then, generate 2,000 examples unifrmly, which means you will have approximately 1, 000 examples for each class.

(a) Run the PLA starting from $w = 0$ until it converges. Plot the data and the final hypothesis.

(b) Repeat part (a) using the linear regression (for classification) to obtain w. Explain your observations.

这题看了非常久，楞是不明白什么意思，后来看了论坛里老师的回复才明白。题目的意思是有如图两个圆环，我们要在圆环内的区域随机生成点，然后用PLA和线性回归进行分类。我们先生成数据，这里我们将上半个圆环对应的圆心放在(a,b)，那么下半个圆环对应的圆心为$(a + rad + \frac{thk}{2}, b - sep)$。为方便叙述，这里记录上半个圆环的圆心为$(X_1, Y_2)$，下半个圆环的圆心为$(X_2, Y_2)$。

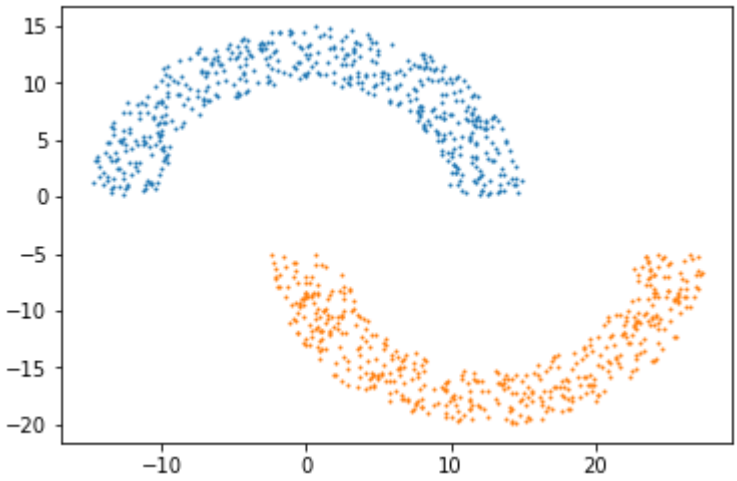
接着介绍生成点的方式，我们在$[-rad - thk, rad + thk] \times [-rad - thk, rad + thk]$随机生成点，如果这个点$(x, y)$落在$rad^2 \leq x^2 + y^2 \leq (rad + thk)^2$范围内，那么当$y > 0$时，我们把这个点给上半个圆环，产生点$(X_1 + x, Y_1 + y)$，当$y < 0$时，我们把这个点给下半个圆环，产生点$(X_2 + x, Y_2 + y)$；如果产生的点不在$rad^2 \leq x^2 + y^2 \leq (rad + thk)^2$范围内，则重新产生点即可。这里产生点的方法有点类似参数方程的思路，**即产生相对于圆心的相对位置**，下面编程实现一下。

```
import numpy as np

#参数
rad=10
thk=5
sep=5

#n为产生点的个数,x1,y1为上半个圆环的坐标
def generatedata(rad,thk,sep,n,x1=0,y1=0):
    #上半个圆的圆心
    X1=x1
    Y1=y1

    #下半个圆的圆心
    X2=X1+rad+thk/2
    Y2=Y1-sep

    #上半个圆环的点
```

```
    top=[]
    #下半个圆环的点
    bottom=[]

    #后面要用到的参数
    r1=rad+thk
    r2=rad

    cnt=1
    while(cnt<=n):
        #产生均匀分布的点
        x=np.random.uniform(-r1,r1)
        y=np.random.uniform(-r1,r1)

        d=x**2+y**2
        if(d>=r2**2 and d<=r1**2):
            if (y>0):
                top.append([X1+x,Y1+y])
                cnt+=1
            else:
                bottom.append([X2+x,Y2+y])
                cnt+=1
        else:
            continue

    return top,bottom
```

接着作图看下

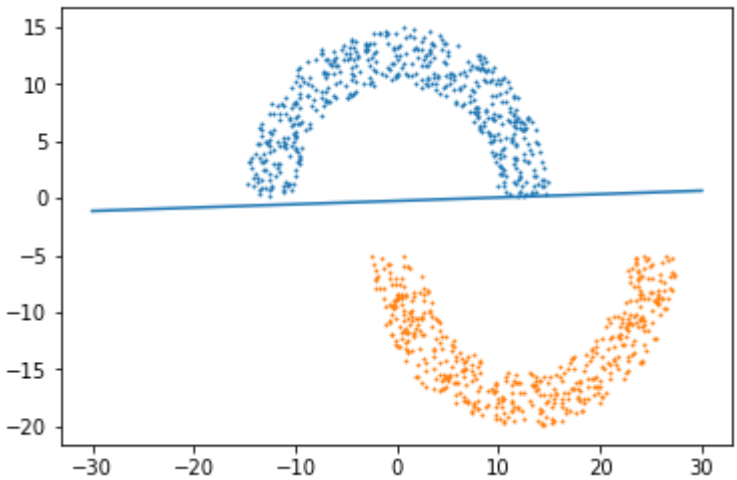
```
import matplotlib.pyplot as plt
top,bottom=generatedata(rad,thk,sep,1000)

X1=[i[0] for i in top]
Y1=[i[1] for i in top]

X2=[i[0] for i in bottom]
Y2=[i[1] for i in bottom]

plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.show()
```



可以看到和圆环形状还是很接近的，接着我们处理(a)

```
#对数据预处理，加上标签和偏移项1
x1=[[1]+i+[1] for i in top]
x2=[[1]+i+[-1] for i in bottom]
data=x1+x2

data=np.array(data)
np.random.shuffle(data)

#PLA
#定义sign函数
def sign(x):
    if x>0:
        return 1
    else:
        return -1

#定义判别函数，判断所有数据是否分类完成
def Judge(x,w):
    #n为数据维度
    n=x.shape[1]-1
    flag=1
    for i in x:
        if sign(i[:n].dot(w))*i[-1]<0:
            flag=0
            break
    return flag

#定义PLA,k为步长
def PLA(x,k):
    #n为数据维度,m为数据数量,
    m,n=x.shape
    n-=1
    #初始化向量
    w=np.zeros(n)
    #记录最后一个更新的向量
    last=0
```

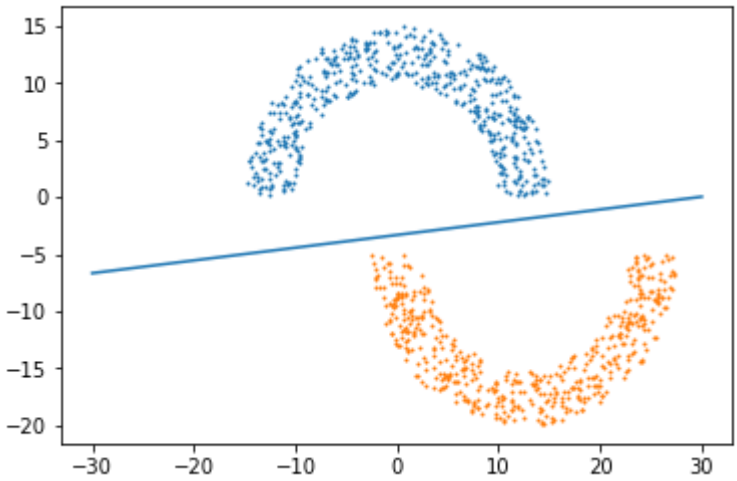
```
    #记录次数
    t=0
    if Judge(x,w):
        pass
    else:
        #记录取哪个元素
        j=0
        while Judge(x,w)==0:
            i=x[j]
            #print(i[:n],i[-1])
            if sign(i[:n].dot(w))*i[-1]<0:
                w+=k*i[-1]*i[:n]
                t+=1
                last=j
            j+=1
            if(j>=m):
                j=j%m
    return t,last,w
t,last,w=PLA(data,1)

#作图
r=2*(rad+thk)
X3=[-r,r]
Y3=[-(w[0]+w[1]*i)/w[2] for i in X3]

plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.plot(X3,Y3)
plt.show()
```



(b)依旧要对数据作预处理，然后直接带入线性回归的公式即可

```
from numpy.linalg import inv

#对数据预处理
x1=[[1]+i for i in top]
y1=[1]*len(top)
```

```
x2=[[1]+i for i in bottom]
y2=[-1]*len(bottom)

X=np.array(x1+x2)
Y=np.array(y1+y2)

w1=inv(X.T.dot(X)).dot(X.T).dot(Y)

#作图
t=2*(rad+thk)
X4=[-t,t]
Y4=[-(w1[0]+w1[1]*i)/w1[2] for i in X4]


plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.plot(X4,Y4)
plt.show()
```



可以看到，线性回归同样能解决这个分类问题，和课本里的描述一致，相比PLA，线性回归得到的直线距离点集的间距更大。

Problem 3.2 (Page 109)

For the double semi circle task in Problem 3.1, vary sep in the range $\{0.2, 0.4, \ldots, 5\}$. Generate 2,000 examples and run the PLA starting with $w = 0$. Record the number of iterations PLA takes to converge.

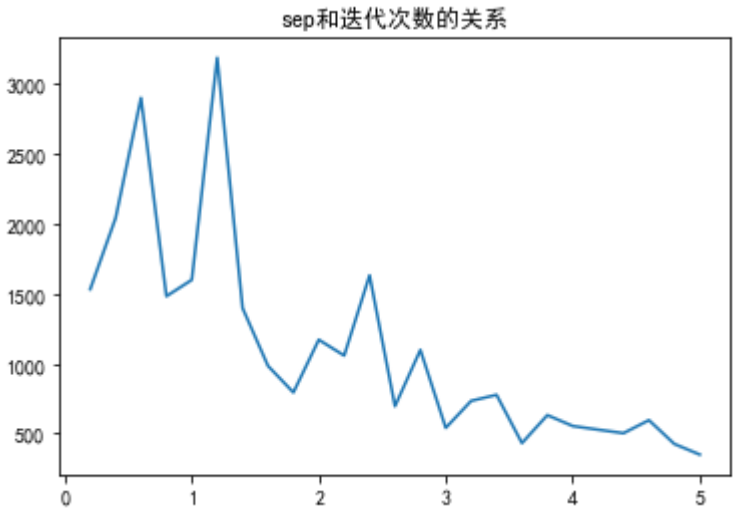
Plot sep versus the number of iterations taken for PLA to converge. Explain your observations. [Hint: Problem 1.3.]

和上题一致，不过这里改变了sep，我们要观察sep和迭代次数的关系，注意这里我们更换下上半个圆环的圆心

```
import numpy as np
#参数
rad=10
thk=5
sep=np.arange(0.2,5.2,0.2)
```

```
T=np.array([])

for k in sep:
    top,bottom=generatedata(rad,thk,k,2000,5,10)
    x1=[[1]+i+[1] for i in top]
    x2=[[1]+i+[-1] for i in bottom]
    data=x1+x2

    data=np.array(data)
    np.random.shuffle(data)

    #维度
    n=len(data[0])-1
    #数据组数
    m=len(data)

    t,last,w=PLA(data,1)

    T=np.append(T,t)
```

```
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

plt.plot(sep,T)
plt.title('sep和迭代次数的关系')
plt.show()
```



可以看到sep越大，迭代次数总体来说在下降。现在来简单分析下原因，回顾Problem 1.3(Page 33)

$$\rho = min_{1\leq n\leq N} y_n(w^{*T}x_n)$$
$$R = max_{1\leq n\leq N}||xn||$$
$$t \leq \frac{R^2||w^*||^2}{\rho^2}$$

这里圆环的位置固定，所以可以认为R是一个常数，我们来分析$\frac{||w^*||}{\rho}$。

由解析几何知识，我们知道$\frac{||w^T x_n||}{||w||}$为$x_n$到平面$w^T x = 0$的距离，而题目中的$\frac{||w^*||}{\rho}$相当于点集到平面$w^{*T} x = 0$的最小距离的倒数，因此如果$sep$越大，最小距离也越大，从而$\frac{||w^*||}{\rho}$越小，迭代次数相应也会减少

Problem 3.3 (Page 109)

For the double semi circle task in Problem 3.1, set $sep = -5$ and generate 2, 000 examples.

(a) What will happen if you run PLA on those examples?

(b) Run the pocket algorithm for 100,000 iterations and plot E_{in} versus the iteration number t.

(c) Plot the data and the final hypothesis in part (b).

(d) Use the linear regression algorithm to obtain the weights w, and compare this result with the pocket algorithm in terms of computation time and quality of the solution .

(e) Repeat (b) - (d) with a 3rd order polynomial fatu re transform.

这题的$sep < 0$，所以数据不可分

(a)如果运行PLA，那么算法不会停下来

(b)回顾下Pocket PLA

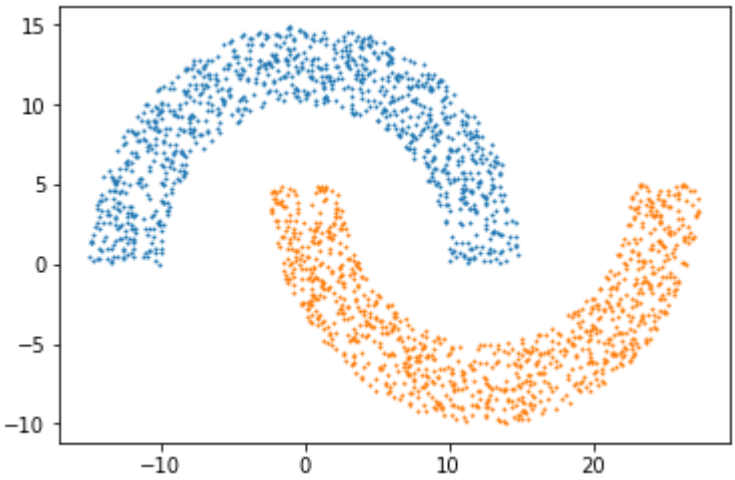
The pocket algorithm:

1: Set the pocket weight vector $\hat{\mathbf{w}}$ to $\mathbf{w}(0)$ of PLA.
2: **for** $t = 0, \ldots, T \quad 1$ **do**
3: Run PLA for one update to obtain $\mathbf{w}(t+1)$.
4: Evaluate $E_{\text{in}}(\mathbf{w}(t+1))$.
5: If $\mathbf{w}(t+1)$ is better than $\hat{\mathbf{w}}$ in terms of E_{in}, set $\hat{\mathbf{w}}$ to $\mathbf{w}(t+1)$.
6: Return $\hat{\mathbf{w}}$.

编程实现，题目要求迭代十万次，我发现十万次实在是太慢了，所以改为10000次，先作图看一下

```
#参数
rad=10
thk=5
sep=-5

#产生数据
top,bottom=generatedata(rad,thk,sep,2000)

#作图
X1=[i[0] for i in top]
Y1=[i[1] for i in top]

X2=[i[0] for i in bottom]
Y2=[i[1] for i in bottom]

plt.scatter(X1,Y1,s=1)
```

```
plt.scatter(X2,Y2,s=1)
plt.show()
```



接着编写Pocket PLA

```
#Pocket PLA
#定义sign函数
def sign(x):
    if x>0:
        return 1
    else:
        return -1

#定义计算错误个数的函数
def CountError(x,w):
    #数据维度
    n=x.shape[1]-1
    #记录错误次数
    count=0
    for i in x:
        if sign(i[:n].dot(w))*i[-1]<0:
            count+=1
    return count


#定义PocketPLA,k为步长,max为最大更新次数
def PocketPLA(x,k,maxnum):
    #n为数据维度,m为数据数量
    m,n=x.shape
    #注意多了一个标志位
    n-=1
    #初始化向量
    w=np.zeros(n)
    #错误率最小的向量
    w0=np.zeros(n)
    error=CountError(x,w)
    #记录每次的错误
    Error=[]
```

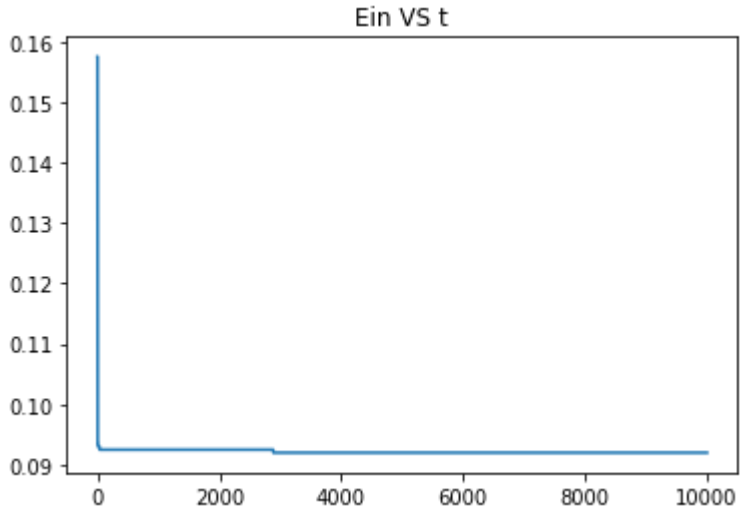
```
    if error==0:
        pass
    else:
        #记录次数
        j=0
        while (j<maxnum or error==0):
            #随机选取数据
            k=np.random.randint(0,m)
            i=x[k]
            #得到w(t+1)
            w=w0+k*i[-1]*i[:n]
            error1=CountError(x,w)
            #如果w(t+1)比w(t)效果好，则更新w(t)
            if error>error1:
                w0=w[:]
                error=error1
            Error.append(error)
            j+=1
    return w0,Error
```

带入算法

```
#对数据预处理，加上标签和偏移项1
x1=[[1]+i+[1] for i in top]
x2=[[1]+i+[-1] for i in bottom]
data=x1+x2

data=np.array(data)
np.random.shuffle(data)

#迭代次数
num=10000

w,error=PocketPLA(data,1,num)
```

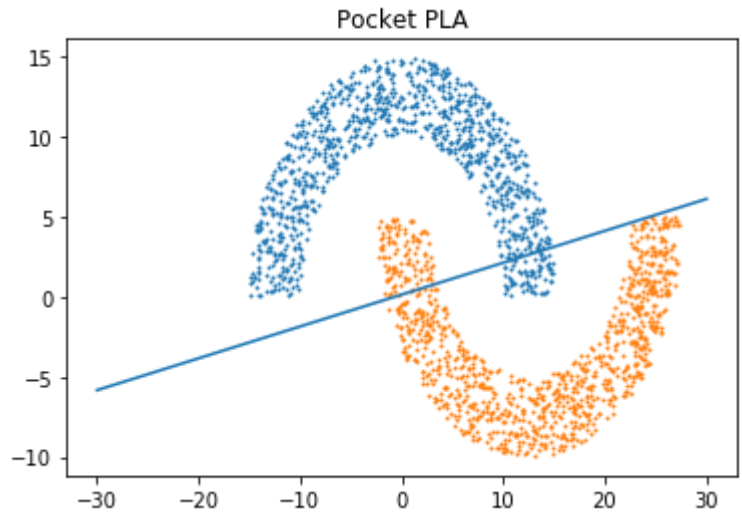
(b)的图

```
t=np.arange(num)
plt.plot(t,np.array(error)/data.shape[0])
plt.title('Ein VS t')
plt.show()
```



(c)的图

```
plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
t=2*(rad+thk)
X3=[-t,t]
Y3=[-(w[0]+w[1]*i)/w[2] for i in X3]
plt.plot(X3,Y3)
plt.title('Pocket PLA')
plt.show()
print('Pocket PLA的错误率'+str(CountError(data,w)/data.shape[0]))
```

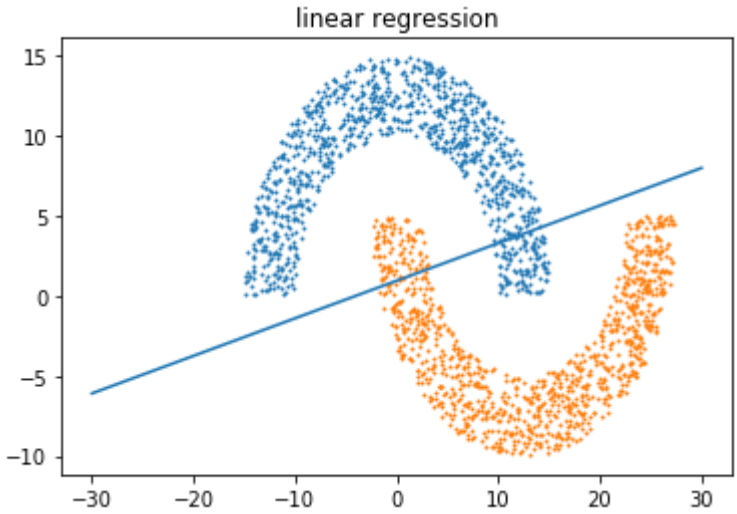



```
Pocket PLA的错误率0.092
```

(d)linear regression，之前有处理过，直接带公式即可

```
#对数据预处理
x1=[[1]+i for i in top]
y1=[1]*len(top)
```

```
x2=[[1]+i for i in bottom]
y2=[-1]*len(bottom)

X=np.array(x1+x2)
Y=np.array(y1+y2)

w1=inv(X.T.dot(X)).dot(X.T).dot(Y)

#作图
t=2*(rad+thk)
X4=[-t,t]
Y4=[-(w1[0]+w1[1]*i)/w1[2] for i in X4]

plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.plot(X4,Y4)
plt.title('linear regression')
plt.show()
print('linear regression的错误率'+str(CountError(data,w1)/data.shape[0]))
```




```
linear regression的错误率0.1005
```

(e)先做三次特征转换，再重复(b)到(d)，注意到三次特征转换为

$$\phi_3(x) = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$$

后续作曲线的图用到了plt.contour函数，原本是用来绘制等高线的，这里可以用来画隐函数的图像

```
#data形式为[  1.        ,   3.05543009,  -3.72519952,  -1.        ]
#特征转换
def transform(data):
    result=[]
    for i in data:
        x1=i[1]
        x2=i[2]
        flag=i[-1]
```

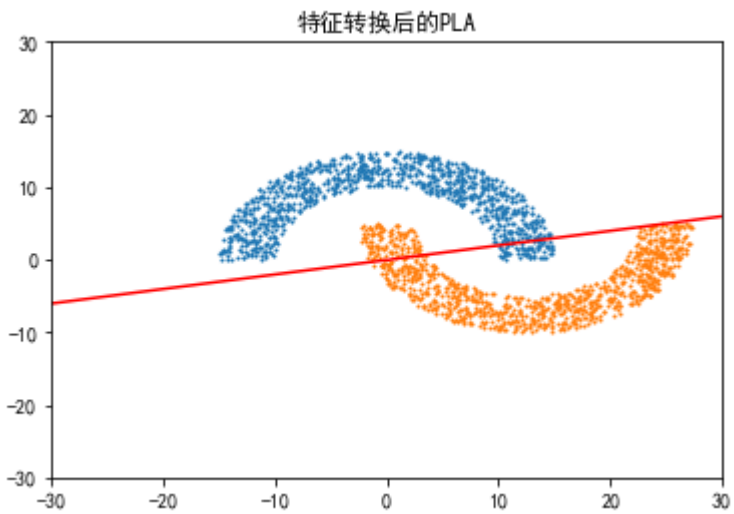
```
            x=[1,x1,x2,x1*x2,x1**2,x2**2,x1**3,(x1**2)*x2,x1*(x2**2),x2**3,flag]
            result.append(x)
    return np.array(result)

newdata=transform(data)

# 定义等高线高度函数
def f(x,y,w):
    return w[0]+w[1]*x+w[2]*y+w[3]*x*y+w[4]*(x**2)+w[5]*(y**2)+w[6]*(x**3)+w[7]*(x**2)*y+w[8]*x*
(y**2)+w[9]*y**3

# 数据数目
n = 2000
# 定义x, y
x = np.linspace(-t, t, n)
y = np.linspace(-t, t, n)

# 生成网格数据
X, Y = np.meshgrid(x, y)
```

将转换过的数据带入Pocket PLA

```
#迭代次数
num=10000

w,error=PocketPLA(newdata,1,num)
```

作图

```
plt.contour(X, Y, f(X, Y,w), 1, colors = 'red')
plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.title('特征转换后的PLA')
plt.show()
print('特征转换后的错误率为'+str(CountError(newdata,w)/newdata.shape[0]))
```



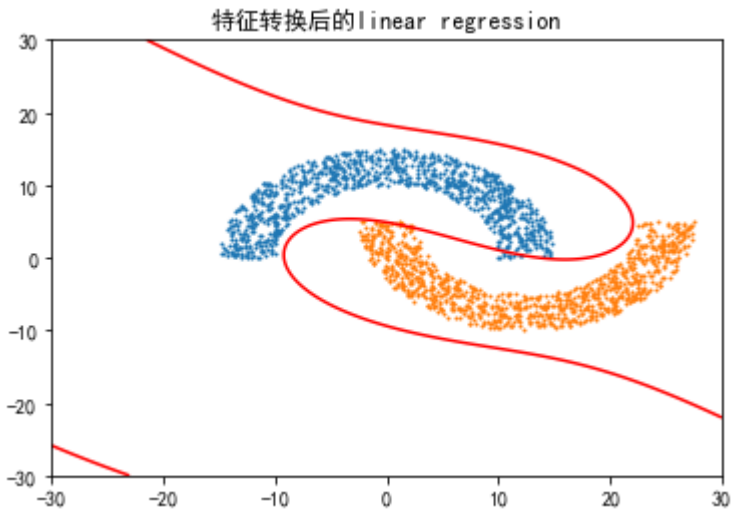
```
特征转换后的错误率为0.093
```

```
t=1000
n=10000
x = np.linspace(-t, t, n)
y = np.linspace(-t, t, n)

# 生成网格数据
X, Y = np.meshgrid(x, y)
plt.contour(X, Y, f(X, Y,w), 1, colors = 'red')
plt.show()
```

```
# 数据数目
n = 10000
# 定义x, y
x = np.linspace(-100, 100, n)
y = np.linspace(-100, 100, n)

# 生成网格数据
X, Y = np.meshgrid(x, y)
```

将转换后的数据带入linear regression

```
#对数据预处理
Xnew=newdata[:,:-1]
Ynew=newdata[:,-1]

w1=inv(Xnew.T.dot(Xnew)).dot(Xnew.T).dot(Ynew)

plt.contour(X, Y, f(X, Y,w1), 1, colors = 'red')
plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.title('特征转换后的linear regression')
plt.show()
print('特征转换后的错误率为'+str(CountError(newdata,w1)/newdata.shape[0]))
```



特征转换后的错误率为0.009

Problem 3.4 (Page 110)

In Problem 1.5, we introduced the Adaptive Linear Neuron (Ada line) algorithm for classification. Here, we derive Ada line from an optimization perspective.

(a) Consider $E_n(w) = (\max(0, 1 - y_n w^T x_n))^2$. Show that $E_n(w)$ is continuous and differentiable. Write down the gradient $\nabla E_n(w)$.

(b) Show that $E_n(w)$ is an upper bound for $[\![sign(w^T x_n) \neq y_n]\!]$. Hence, $\frac{1}{N}\sum_{n=1}^{N} E_n(w)$is an upper bound for the in sample classification error $E_{in}(w)$.

(c) Argue that the Adaline algorithm in Problem 1.5 performs stochastic gradient descent on $\frac{1}{N}\sum_{n=1}^{N} E_n(w)$.

(a)$1 - y_n w^T x_n$关于w是连续的，$max(a, x)$关于x是连续的，所以$\max(0, 1 - y_n w^T x_n)$关于w是连续的，连续函数的平方也是连续的，所以$E_n(w) = (\max(0, 1 - y_n w^T x_n))^2$ 关于w连续。再来看可导性，令$s(w) = 1 - y_n w^T x_n$,$s(w)$显然关于w可导性,我们再来看下$f(s) = (max(0, s))^2$

$$f(s) = \begin{cases} s^2 (s \geq 0) \\ 0 (s < 0) \end{cases}$$

显然这个函数也是可导的。所以$E_n(w) = f(s(w))$也可导。接下来我们求梯度

$$\frac{\partial E_{n(w)}}{w_k} = \begin{cases} \frac{\partial (1 - y_n w^T x_n)^2}{\partial w_k} = (1 - y_n w^T x_n)(-y_n x_n^k) & (y_n w^T x_n \leq 1) \\ 0 & (y_n w^T x_n > 1) \end{cases}$$

因此

$$\nabla E_n(w) = \begin{cases} (1 - y_n w^T x_n)(-y_n x_n) & (y_n w^T x_n \leq 1) \\ 0 & (y_n w^T x_n > 1) \end{cases}$$

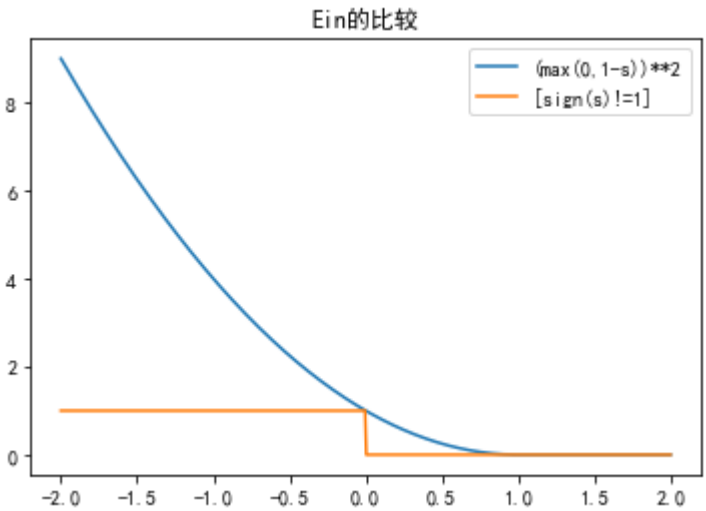
(b)作图，我们先对式子做点变形

$$[\![sign(w^T x_n) \neq y_n]\!] \Leftrightarrow$$
$$[\![y_n \times sign(w^T x_n) \neq y_n \times y_n]\!] \Leftrightarrow$$
$$[\![sign(y_n w^T x_n) \neq 1]\!]$$

令$s = sign(y_n w^T x_n)$

说以上述两个式子可以化为$(max(0, 1-s))^2$以及$[\![sign(s) \neq 1]\!]$

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

def f1(s):
    a=max(0,1-s)
    return a**2

def f2(s):
    if s>0:
        return 0
    else:
        return 1

x=np.linspace(-2,2,500)
y1=[f1(i) for i in x]
y2=[f2(i) for i in x]

plt.plot(x,y1,label="(max(0,1-s))**2")
plt.plot(x,y2,label="[sign(s)!=1]")
plt.legend()
plt.title('Ein的比较')
plt.show()
```



(c)回顾Problem 1.5的更新规则

$$s(t) = w^T(t)x(t), \text{当} y(t) \cdot s(t) \leq 1 \text{时}$$
$$w(t+1) = w(t) + \eta(y(t) - s(t)).x(t)$$

再来看下我们的梯度，稍微做下变形

$$(1-y_nw^Tx_n)(-y_nx_n)=-(y_n-w^Tx_n)x_n$$
$$\nabla E_n(w)=\begin{cases}-(y_n-w^Tx_n)x_n & (y_nw^Tx_n\leq 1)\\ 0 & (y_nw^Tx_n>1)\end{cases}$$

所以随机梯度下降法的更新规则为

$$w(t+1)=w(t)-\eta\nabla E_n(w)=\begin{cases}w(t)+\eta(y_n-w(t)^Tx_n)x_n & (y_nw(t)^Tx_n\leq 1)\\ w(t) & (y_nw(t)^Tx_n>1)\end{cases}$$

我们使用Problem 1.5一样的符号$s(t)=w^T(t)x(t)$，那么随机梯度下降法的更新规则即为Problem 1.5的更新规则

Problem 3.5 (Page 110)

(a) Consider

$$E_n(w)=\max(0,1-y_nw^Tx_n)$$

Show that $E_n(w)$ is continuous and differentiable except when $y_n=w^Tx_n$.

(b) Show that $E_n(w)$ is an upper bound for $[\![sign(w^Tx_n)\neq y_n]\!]$. Hence, $\frac{1}{N}\sum_{n=1}^{N}E_n(w)$is an upper bound for the in sample classification error $E_{in}(w)$.

(c) Apply stochastic gradient descent on $\frac{1}{N}\sum_{n=1}^{N}E_n(w)$(ignoring the singular case of $y_n=w^Tx_n$) and derive a new perceptron learning algorithm.

(a)我们用上一题一样的思路，令$s=y_nw^Tx_n$，$f(s)=max(0,1-s)$关于s连续,s关于w连续，因此$E_n(w)=f(s(w))$关于w连续。

s关于w处处可导，但$f(s)=max(0,1-s)$在$s=1$处不可导，其余点均可导。我们来看下$s=1$的特点，注意$y_n\in\{1,-1\}$，那么

$$s=1\Leftrightarrow$$
$$y_nw^Tx_n=1\Leftrightarrow$$
$$y_n\times y_nw^Tx_n=y_n\Leftrightarrow$$
$$w^Tx_n=y_n$$

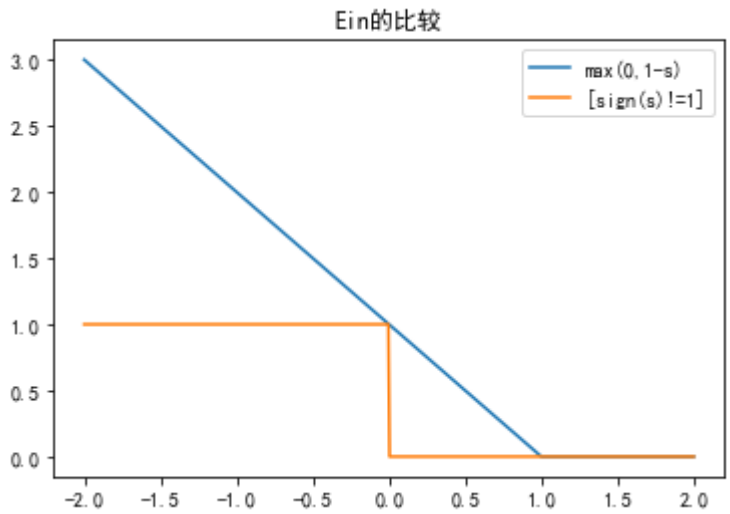
所以$E_n(w)=f(s(w))$在$s=1$即$y_n=w^Tx_n$处不可导，其余点均可导

(b)同Problem 3.4方法，$s=y_nw^Tx_n$

$$E_n(w)=\max(0,1-y_nw^Tx_n)=max(0,1-s)$$
$$[\![sign(s)\neq 1]\!]$$

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
```

```
def f1(s):
    a=max(0,1-s)
    return a

def f2(s):
    if s>0:
        return 0
    else:
        return 1

x=np.linspace(-2,2,500)
y1=[f1(i) for i in x]
y2=[f2(i) for i in x]

plt.plot(x,y1,label="max(0,1-s)")
plt.plot(x,y2,label="[sign(s)!=1]")
plt.legend()
plt.title('Ein的比较')
plt.show()
```



(c)不管不可导点，我们来求梯度，只考虑$y_n w^T x_n < 1$的情形，此时

$$E_n(w) = 1 - y_n w^T x_n$$
$$\frac{\partial E_n(w)}{\partial w_i} = \frac{\partial(1 - y_n w^T x_n)}{\partial w_i} = -y_n x_n^i$$

所以

$$\text{当} y_n w^T x_n < 1\text{时}, \ \nabla E_n(w) = -y_n x_n$$
$$\text{当} y_n w^T x_n \geq 1\text{时}, \ \nabla E_n(w) = 0$$

所以SGD（随机梯度下降法）的更新规则为

$$\text{当} y_n w(t)^T x_n < 1\text{时}$$
$$w(t+1) = w(t) - \eta \nabla E_n(w) = w(t) + \eta y_n x_n$$
$$\text{当} y_n w(t)^T x_n \geq 1\text{时不更新}$$

Problem 3.6 (Page 110)

Derive a linear programming algorithm to fit a linear model for classification using the following steps. A linear program is an optimization problem of the following form:

$$\begin{aligned} \min_{z} \quad & c^T z \\ subject\ to \quad & Az \leq b \end{aligned}$$

A, b and c are parameters of the linear program and z is the optimization variable. This is such a well studied optimization problem that most mathematics software have canned optimization functions which solve linear programs.

(a) For linearly separable data,show that for some w, $y_n(w^T x_n) \geq 1$ for $n = 1, \ldots, N$.

(b) Formulate the task of finding a separating w for separable data as a linear program. You need to specify what the parameters A, b, c are and what the optimization variable z is.

(c) If the data is not separable, the condition in (a) cannot hold for every n. Thus introduce the violation $\xi_n \geq 0$ to capture the amount of violation for example x_n. So, for $n = 1, \ldots, N$,

$$\begin{aligned} y_n(w^T x_n) &\geq 1 - \xi_n \\ \xi_n &\geq 0 \end{aligned}$$

Naturally, we would like to minimize the amount of violation. One intuitive a pproach is to minimize $\sum_{n=1}^{N} \xi_n$, i .e., we want w that solves

$$\begin{aligned} min_{w,\xi_n} \quad & \sum_{n=1}^{N} \xi_n \\ subject\ to \quad & y_n(w^T x_n) \geq 1 - \xi_n \\ & \xi_n \geq 0 \end{aligned}$$

where the inequalities must hold for $n = 1, \ldots, N$. Formulate this problem as a linear program .

(d) Argue that the linear program you derived in (c) and the optimization problem in Problem 3.5 are equivalent.

这题的任务是要把我们之前讨论的线性分类问题转化为线性规划问题。

(a)由第一章的结论，对于线性可分的数据，存在w_1，使得$y_n w_1^T x_n > 0 (n = 1, \ldots, N)$，设$\rho = min_{1 \leq n \leq N} y_n w_1^T x_n$，显然$\rho > 0$，现在取$w = \frac{w_1}{\rho}$，那么

$$y_n w^T x_n = y_n (\frac{w_1}{\rho})^T x_n = \frac{y_n w_1^T x_n}{\rho} \geq 1$$

因此结论成立

(b)这题的限制条件就是刚刚所说的$y_n w^T x_n \geq 1$，因此$z = w$，比较让人费解的是c应该取什么，实际上思考下，我们这里只要找到满足$y_n w^T x_n \geq 1, n = 1, \ldots, N$这个条件的$w$即可，所以这里$c$可以取任意值。结合以上几点，下面把$A, b, c$分别写出，不妨设$w, x_n \in R^d, n = 1, \ldots, N$。

$$z = w = (w_1, \dots w_d)^T$$
$$x_n = (x_n^1, \dots x_n^d)^T$$
$$A = \begin{pmatrix} -y_1 x_1^T \\ \dots \\ -y_N x_N^T \end{pmatrix} = \begin{pmatrix} -y_1 x_1^1 & \dots & -y_1 x_1^d \\ \dots & \dots & \dots \\ -y_N x_N^1 & \dots & -y_N x_N^d \end{pmatrix} \in R^{N \times d}$$
$$b = \begin{pmatrix} -1 \\ \dots \\ -1 \end{pmatrix} \in R^N$$
c为R^d中任意向量

我们看下这里的Az, b

$$Az = \begin{pmatrix} -y_1 x_1^T \\ \dots \\ -y_N x_N^T \end{pmatrix} w = \begin{pmatrix} -y_1 x_1^T w \\ \dots \\ -y_N x_N^T w \end{pmatrix} = \begin{pmatrix} -y_1 w^T x_1 \\ \dots \\ -y_N w^T x_N \end{pmatrix}$$
$$b = \begin{pmatrix} -1 \\ \dots \\ -1 \end{pmatrix}$$

因此$Az \leq b$即为$y_n w^T x_n \geq 1$

(c)依旧设$w, x_n \in R^d, n = 1, \dots, N$，和上一题类似的思路

$$z = (w_1, .. w_d, \xi_1, \dots, \xi_N)^T \in R^{N+d}$$
记$A_1 = \begin{pmatrix} -y_1 x_1^T \\ \dots \\ -y_N x_N^T \end{pmatrix} \in R^{N \times d}, I_{N \times N}$为$N \times N$阶单位矩阵
$$A2 = \left(\begin{array}{c|c} A_1 & -I_{N \times N} \end{array} \right) \in R^{N \times (N+d)}$$
$$A3 = \left(\begin{array}{c|c} 0 & -I_{N \times N} \end{array} \right) \in R^{N \times (N+d)} (0\text{为}N \times d\text{阶}0\text{矩阵})$$
$$A = \begin{pmatrix} A1 \\ A2 \end{pmatrix} \in R^{(2N) \times (N+d)}$$
$b = (-1 ... -1, 0 ... 0)^T \in R^{2N}$，其中前$N$个分量为$-1$，其余为$0$
$c = (0, ..0, 1, \dots 1)^T \in R^{N+d}$，其中$c$的前$d$个分量为$0$，后$N$的分量为$1$

同上一题的验证方法可以知此问题即为原来的问题。

(d)回顾下3.5

$$E_n(w) = \max(0, 1 - y_n w^T x_n)$$

我们的目标是最小化$\frac{1}{N} \sum_{n=1}^N E_n(w)$

这里我们令$\xi_n = E_n(w) = \max(0, 1 - y_n w^T x_n)$，那么

$$1 - y_n w^T x_n \leq \xi_n$$
$$0 \leq \xi_n$$
$$\frac{1}{N} \sum_{n=1}^N E_n(w) = \frac{1}{N} \sum_{n=1}^N \xi_n$$

注意N为常数，所以3.5即为

$$\text{在条件}1-y_n w^T x_n \leq \xi_n \text{和} 0 \leq \xi_n \text{下最小化}$$
$$\sum_{n=1}^{N} \xi_n$$

这就是我们刚刚考虑的问题。其实这就是著名的支持向量机算法（SVM）

Problem 3.7 (Page 111)

Use the linear programming algorithm from Problem 3.6 on the learning task in Problem 3.1 for the separable ($sep = 5$) and the non separable ($sep = -5$) cases. Compare your results to the linear regression approach with and without the 3rd order polynomial feature transform.

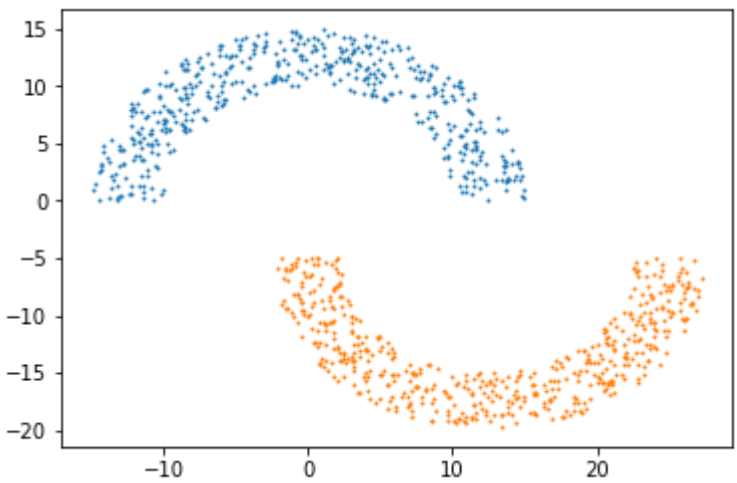
这题的意思是使用线性规划方法对3.1再实践一遍，这里遇到个坑，一开始使用scipy的linprog函数做优化，发现一直显示无解，后来网上一查发现有人说这个函数如果数据量大于100就会求不出解，然后我将我的数据量由1000改为50，果然一下有解了，最后的解决方法是cvxopt包，这个包的效果非常好。后面记第一个算法为算法1，第二个算法为算法2,分别给出结果。

```
import numpy as np

#参数
rad=10
thk=5
sep=5

#n为产生点的个数,x1,y1为上半个圆环的坐标
def generatedata(rad,thk,sep,n,x1=0,y1=0):
    #上半个圆的圆心
    X1=x1
    Y1=y1

    #下半个圆的圆心
    X2=X1+rad+thk/2
    Y2=Y1-sep

    #上半个圆环的点
    top=[]
    #下半个圆环的点
    bottom=[]

    #后面要用到的参数
    r1=rad+thk
    r2=rad

    cnt=1
    while(cnt<=n):
        #产生均匀分布的点
        x=np.random.uniform(-r1,r1)
        y=np.random.uniform(-r1,r1)
```

```
        d=x**2+y**2
        if(d>=r2**2 and d<=r1**2):
            if (y>0):
                top.append([X1+x,Y1+y])
                cnt+=1
            else:
                bottom.append([X2+x,Y2+y])
                cnt+=1
        else:
            continue

    return top,bottom
```

作图

```
import matplotlib.pyplot as plt
n=1000
top,bottom=generatedata(rad,thk,sep,n)

X1=[i[0] for i in top]
Y1=[i[1] for i in top]

X2=[i[0] for i in bottom]
Y2=[i[1] for i in bottom]

plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.show()
```

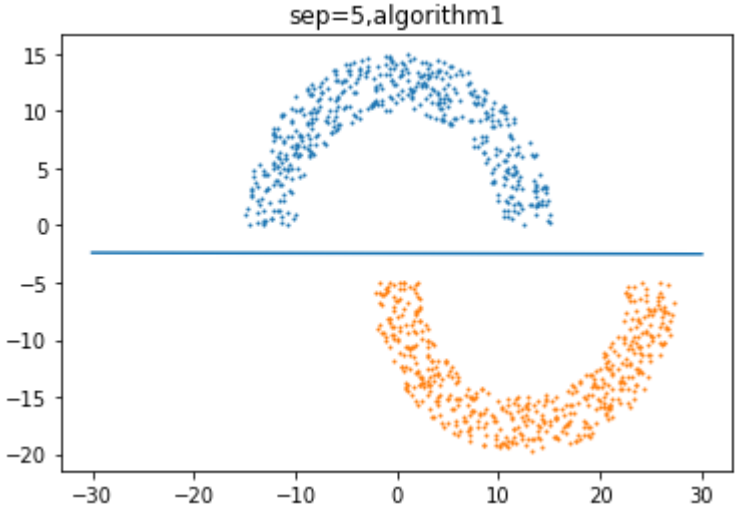


(1)特征转换之前，算法1，sep=5

```
from cvxopt import matrix, solvers

#加上偏移项
x1=[[1]+i for i in top]
x2=[[1]+i for i in bottom]
```

```
c=np.array([1.0,1.0,1.0])
A=np.concatenate((np.array(x1),np.array(x2)*(-1)))*(-1)
b=np.ones(n)*(-1.0)

c=matrix(c)
A=matrix(A)
b=matrix(b)

sol = solvers.lp(c,A,b)

w=list((sol['x']))

#作图
r=2*(rad+thk)
X3=[-r,r]
Y3=[-(w[0]+w[1]*i)/w[2] for i in X3]

plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.plot(X3,Y3)
plt.title('sep=5,algorithm1')
plt.show()
print(w)
```

```
     pcost       dcost       gap    pres   dres   k/t
 0:  3.0133e-01  1.0016e+03  2e+03  2e+00  7e+03  1e+00
 1:  2.9917e+01  4.8276e+04  9e+06  9e+01  3e+05  2e+02
 2:  6.1117e-01  4.3696e+02  6e+02  7e-01  3e+03  5e+01
 3:  7.9437e-01  5.2302e+02  1e+03  9e-01  3e+03  6e+01
 4:  1.8369e+00  7.0321e+01  1e+02  7e-02  3e+02  3e+01
 5:  1.8318e+00  3.3501e+00  4e+00  2e-03  8e+00  4e-01
 6:  1.4266e+00  1.9433e+00  1e+00  7e-04  3e+00  1e-01
 7:  1.4291e+00  1.8611e+00  1e+00  6e-04  2e+00  1e-01
 8:  1.3934e+00  1.5405e+00  3e-01  2e-04  8e-01  3e-02
 9:  1.3881e+00  1.4650e+00  2e-01  1e-04  5e-01  1e-02
10:  1.3829e+00  1.3850e+00  6e-03  4e-06  1e-02  2e-04
11:  1.3827e+00  1.3827e+00  6e-05  4e-08  1e-04  2e-06
12:  1.3827e+00  1.3827e+00  6e-07  4e-10  1e-06  2e-08
13:  1.3827e+00  1.3827e+00  6e-09  4e-12  1e-08  2e-10
Optimal solution found.
```




```
[0.9826141306811725, 0.0007542102316126465, 0.39934445503821364]
```

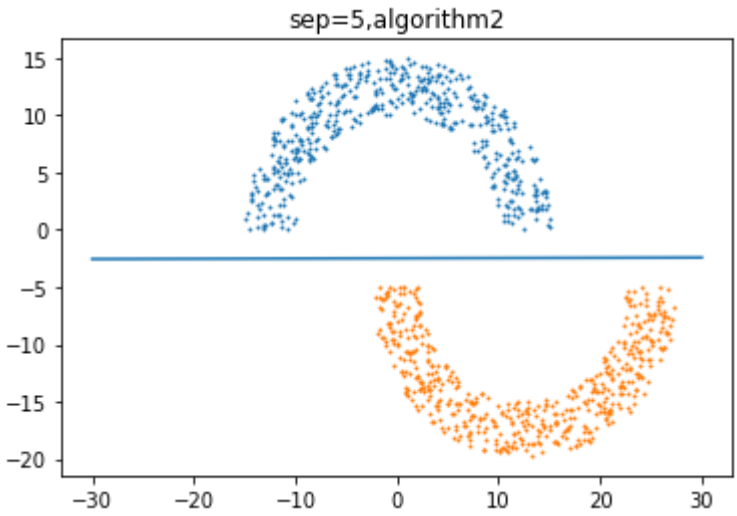
可以看到线性规划产生的直线比感知机的结果更好一些，因为离两组数据更远。

(2)特征转换之前，算法2，sep=5

```
#根据之前所述构造矩阵
a1=np.concatenate((A,(-1)*np.eye(n)),axis=1)
a2=np.concatenate((np.zeros([n,3]),(-1)*np.eye(n)),axis=1)

A1=np.concatenate((a1,a2))
c1=np.array([0.0]*3+[1.0]*1000)
b1=np.array([-1.0]*1000+[0.0]*1000)

#带入算法求解
c1=matrix(c1)
A1=matrix(A1)
b1=matrix(b1)

sol1 = solvers.lp(c1,A1,b1)

#作图
w=list((sol1['x']))
#作图
r=2*(rad+thk)
X3=[-r,r]
Y3=[-(w[0]+w[1]*i)/w[2] for i in X3]

plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.plot(X3,Y3)
plt.title('sep=5,algorithm2')
plt.show()
```

```
     pcost       dcost       gap    pres   dres   k/t
```

```
 0:  5.9187e+01  1.2658e+03  5e+03  2e+00  5e+02  1e+00
 1:  9.6869e+01  2.5471e+02  4e+02  3e-01  6e+01  1e+00
 2:  6.5506e+01  1.4718e+02  2e+02  1e-01  3e+01  7e-01
 3:  4.7019e+01  9.9537e+01  2e+02  9e-02  2e+01  4e-01
 4:  3.5215e+01  7.3439e+01  1e+02  6e-02  2e+01  3e-01
 5:  2.6599e+01  5.4888e+01  1e+02  5e-02  1e+01  2e-01
 6:  1.9412e+01  3.9803e+01  8e+01  3e-02  8e+00  2e-01
 7:  1.3697e+01  2.7965e+01  6e+01  2e-02  6e+00  1e-01
 8:  9.9267e+00  2.0330e+01  4e+01  2e-02  4e+00  7e-02
 9:  7.3586e+00  1.5211e+01  3e+01  1e-02  3e+00  4e-02
10:  3.8519e+00  7.7915e+00  2e+01  7e-03  2e+00  8e-03
11:  1.1403e+00  2.3409e+00  5e+00  2e-03  5e-01  3e-03
12:  3.3103e-02  6.8245e-02  2e-01  6e-05  1e-02  7e-05
13:  3.3161e-04  6.8366e-04  2e-03  6e-07  1e-04  7e-07
14:  3.3161e-06  6.8366e-06  2e-05  6e-09  1e-06  7e-09
15:  3.3161e-08  6.8366e-08  2e-07  6e-11  1e-08  7e-11
16:  3.3161e-10  6.8366e-10  2e-09  6e-13  1e-10  7e-13
Optimal solution found.
```



(3)特征转换之前，算法1，sep=-5

```
sep=-5
top,bottom=generatedata(rad,thk,sep,n)

X1=[i[0] for i in top]
Y1=[i[1] for i in top]

X2=[i[0] for i in bottom]
Y2=[i[1] for i in bottom]

#加上偏移项
x1=[[1]+i for i in top]
x2=[[1]+i for i in bottom]

c=np.array([1.0,1.0,1.0])
```

```
A=np.concatenate((np.array(x1),np.array(x2)*(-1)))*(-1)
b=np.ones(n)*(-1.0)

c=matrix(c)
A=matrix(A)
b=matrix(b)

sol = solvers.lp(c,A,b)
```

```
     pcost       dcost       gap    pres   dres   k/t
 0: -2.8054e-02  1.0019e+03  2e+03  2e+00  5e+03  1e+00
 1: -2.3018e+00  6.2068e+04  1e+07  2e+02  3e+05  3e+02
 2: -9.6404e-02  9.2573e+02  2e+03  2e+00  4e+03  1e+02
 3: -1.8134e-01  1.0723e+03  3e+03  2e+00  4e+03  2e+02
 4: -6.7500e-01  1.7910e+03  8e+03  3e+00  6e+03  5e+02
 5: -5.9176e-01  1.9644e+03  9e+03  3e+00  6e+03  7e+02
 6: -1.9019e+00  7.0119e+03  4e+04  5e+00  1e+04  5e+03
 7: -1.5197e+00  8.6818e+03  5e+04  5e+00  1e+04  7e+03
 8: -2.0117e+00  1.5067e+05  6e+05  5e+00  1e+04  1e+05
 9: -2.0611e+00  1.5086e+07  6e+07  5e+00  1e+04  2e+07
10: -2.0611e+00  1.5087e+09  6e+09  5e+00  1e+04  2e+09
11: -2.0611e+00  1.5087e+11  6e+11  5e+00  1e+04  2e+11
Certificate of primal infeasibility found.
```

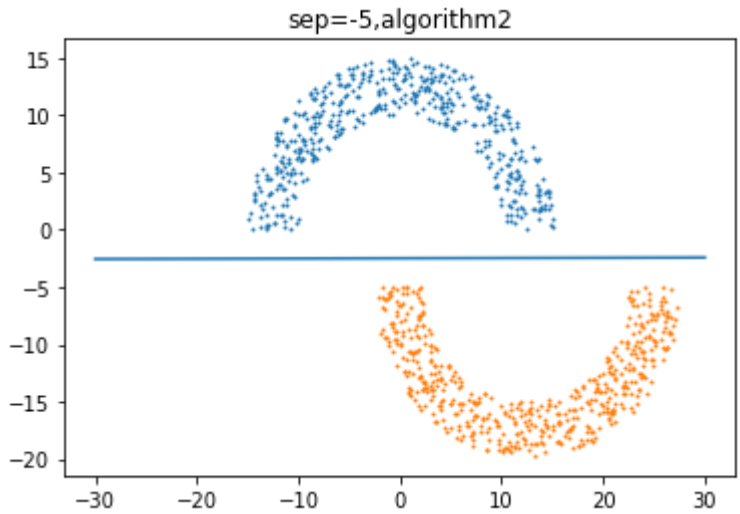
此时无解，因为不可分。

(4)特征转换之前，算法2，sep=-5

```
#根据之前所述构造矩阵
a1=np.concatenate((A,(-1)*np.eye(n)),axis=1)
a2=np.concatenate((np.zeros([n,3]),(-1)*np.eye(n)),axis=1)

A1=np.concatenate((a1,a2))
c1=np.array([0.0]*3+[1.0]*1000)
b1=np.array([-1.0]*1000+[0.0]*1000)

#带入算法求解
c1=matrix(c1)
A1=matrix(A1)
b1=matrix(b1)

sol1 = solvers.lp(c1,A1,b1)

#作图
w=list((sol1['x']))
#作图
r=2*(rad+thk)
X3=[-r,r]
Y3=[-(w[0]+w[1]*i)/w[2] for i in X3]

plt.scatter(X1,Y1,s=1)
```

```
plt.scatter(X2,Y2,s=1)
plt.plot(X3,Y3)
plt.title('sep=-5,algorithm2')
plt.show()
```

```
     pcost       dcost       gap    pres   dres   k/t
 0:  5.9187e+01  1.2658e+03  5e+03  2e+00  5e+02  1e+00
 1:  9.6869e+01  2.5471e+02  4e+02  3e-01  6e+01  1e+00
 2:  6.5506e+01  1.4718e+02  2e+02  1e-01  3e+01  7e-01
 3:  4.7019e+01  9.9537e+01  2e+02  9e-02  2e+01  4e-01
 4:  3.5215e+01  7.3439e+01  1e+02  6e-02  2e+01  3e-01
 5:  2.6599e+01  5.4888e+01  1e+02  5e-02  1e+01  2e-01
 6:  1.9412e+01  3.9803e+01  8e+01  3e-02  8e+00  2e-01
 7:  1.3697e+01  2.7965e+01  6e+01  2e-02  6e+00  1e-01
 8:  9.9267e+00  2.0330e+01  4e+01  2e-02  4e+00  7e-02
 9:  7.3586e+00  1.5211e+01  3e+01  1e-02  3e+00  4e-02
10:  3.8519e+00  7.7915e+00  2e+01  7e-03  2e+00  8e-03
11:  1.1403e+00  2.3409e+00  5e+00  2e-03  5e-01  3e-03
12:  3.3103e-02  6.8245e-02  2e-01  6e-05  1e-02  7e-05
13:  3.3161e-04  6.8366e-04  2e-03  6e-07  1e-04  7e-07
14:  3.3161e-06  6.8366e-06  2e-05  6e-09  1e-06  7e-09
15:  3.3161e-08  6.8366e-08  2e-07  6e-11  1e-08  7e-11
16:  3.3161e-10  6.8366e-10  2e-09  6e-13  1e-10  7e-13
Optimal solution found.
```



第二种算法没有要求可分，所以是有解的。

(5)特征转换后，算法1，sep=-5

```
#data形式为[  1.        ,   3.05543009,  -3.72519952,  -1.        ]
#特征转换
def transform(data):
    result=[]
    for i in data:
        x1=i[1]
```

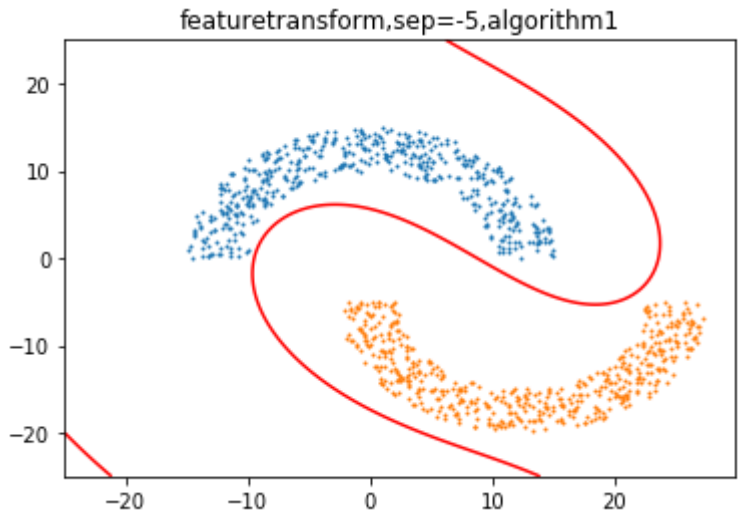
```
        x2=i[2]
        flag=i[-1]
        x=np.array([1,x1,x2,x1*x2,x1**2,x2**2,x1**3,(x1**2)*x2,x1*(x2**2),x2**3,flag])
        result.append(x)
    return result

#对数据预处理，加上标签和偏移项1
x1=[[1]+i+[1] for i in top]
x2=[[1]+i+[-1] for i in bottom]
data=x1+x2
newdata=transform(data)

finaldata=[]
for i in newdata:
    temp=list(i[:-1]*i[-1])
    finaldata.append(temp)

c=np.array([1.0]*10)
A=np.array(finaldata)*(-1)
b=np.ones(n)*(-1.0)

c=matrix(c)
A=matrix(A)
b=matrix(b)

sol = solvers.lp(c,A,b)
```

```
     pcost       dcost       gap    pres   dres   k/t
 0:  2.3620e-01  1.0116e+03  2e+03  2e+00  1e+06  1e+00
 1:  2.9224e+01  2.3607e+04  4e+06  4e+01  3e+07  2e+02
 2:  3.8486e+01  3.7031e+03  6e+05  6e+00  4e+06  2e+00
 3: -9.9194e-02  2.7886e+01  3e+01  4e-02  3e+04  1e+00
 4: -5.3991e-01  2.4845e+01  4e+01  4e-02  3e+04  2e+00
 5: -8.4799e-01  2.8108e+01  7e+01  4e-02  3e+04  2e+00
 6: -8.3387e+00  2.8715e+01  4e+02  4e-02  3e+04  9e+00
 7: -8.7869e+02  3.8457e+01  5e+04  6e-02  5e+04  9e+02
 8: -8.7920e+04  3.8459e+01  5e+06  6e-02  5e+04  9e+04
 9: -8.7921e+06  3.8459e+01  5e+08  6e-02  5e+04  9e+06
10: -8.7921e+08  3.8459e+01  5e+10  6e-02  5e+04  9e+08
Certificate of dual infeasibility found.
```

接着作图看下，利用之前等高线的方法。

```
# 定义等高线高度函数
def f(x,y,w):
    return w[0]+w[1]*x+w[2]*y+w[3]*x*y+w[4]*(x**2)+w[5]*(y**2)+w[6]*(x**3)+w[7]*(x**2)*y+w[8]*x*
(y**2)+w[9]*y**3

# 数据数目
m = 2000
#定义范围
```

```
t=25
# 定义x, y
x = np.linspace(-t, t, m)
y = np.linspace(-t, t, m)

# 生成网格数据
X, Y = np.meshgrid(x, y)


w=list((sol['x']))


plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.contour(X, Y, f(X, Y,w), 1, colors = 'red')
plt.title('featuretransform,sep=-5,algorithm1')
plt.show()
print(w)
```




```
[-1.2908892881068699, 0.06563261448389339, 0.19659674176761507, 0.00803495373549107,
0.015065396638491895, 0.008146061450369876, -0.0006577304715118114, -0.0006010514387000703,
-0.0008964076525334598, -0.0004312904062460984]
```

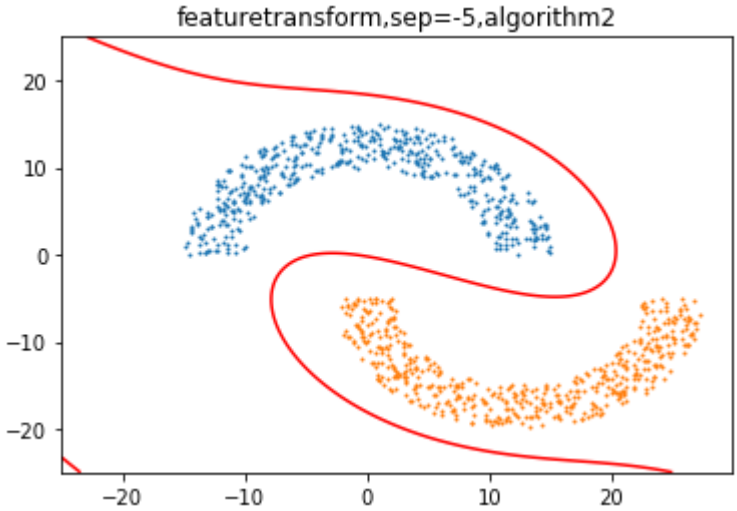
效果还是相当不错的，最后使用算法2

(6)特征转换后，算法2，sep=-5

```
#根据之前所述构造矩阵
a1=np.concatenate((A,(-1)*np.eye(n)),axis=1)
a2=np.concatenate((np.zeros([n,10]),(-1)*np.eye(n)),axis=1)

A1=np.concatenate((a1,a2))
c1=np.array([0.0]*10+[1.0]*1000)
b1=np.array([-1.0]*1000+[0.0]*1000)

#带入算法求解
```

```
c1=matrix(c1)
A1=matrix(A1)
b1=matrix(b1)

sol1 = solvers.lp(c1,A1,b1)

#作图
w=list((sol1['x']))

# 定义x, y
x = np.linspace(-t, t, m)
y = np.linspace(-t, t, m)

# 生成网格数据
X, Y = np.meshgrid(x, y)


plt.scatter(X1,Y1,s=1)
plt.scatter(X2,Y2,s=1)
plt.contour(X, Y, f(X, Y,w), 1, colors = 'red')
plt.title('featuretransform,sep=-5,algorithm2')
plt.show()
```

```
     pcost       dcost       gap    pres   dres   k/t
 0:  9.8921e+00  1.1827e+03  4e+03  2e+00  1e+05  1e+00
 1:  1.2573e+01  1.7458e+02  3e+02  2e-01  2e+04  1e+00
 2:  5.1100e+00  6.5809e+01  1e+02  9e-02  7e+03  4e-01
 3:  2.0670e+00  2.6862e+01  5e+01  4e-02  3e+03  2e-01
 4:  1.1901e-01  2.2473e+00  4e+00  3e-03  3e+02  1e-02
 5:  1.2202e-03  2.2998e-02  4e-02  3e-05  3e+00  1e-04
 6:  1.2202e-05  2.2998e-04  4e-04  3e-07  3e-02  1e-06
 7:  1.2202e-07  2.2998e-06  4e-06  3e-09  3e-04  1e-08
 8:  1.2202e-09  2.2998e-08  4e-08  3e-11  3e-06  1e-10
 9:  1.2202e-11  2.2998e-10  4e-10  3e-13  3e-08  1e-12
Optimal solution found.
```



算法2的效果更加的好，因为它限制了距离。

Problem 3.8 (Page 111)

For linear regression, the out of sample error is

$$E_{out}(h) = E[(h(x) - y)^2]$$

Show that among all hypotheses, the one that minimizes $E_{out}(h)$ is given by

$$h^*(x) = E[y|x]$$

The function h^* can be treated as a deterministic target function, in which case we can write $y = h^*(x) + \epsilon(x)$ where $\epsilon(x)$ is an (input dependent) noise variable. Show that $\epsilon(x)$ has expected value zero.

这题其实是统计学习里一个比较常见的结论。

$$\begin{aligned}
E_{out}(h) &= E[(h(x) - y)^2] \\
&= E[(h(x) - E[y|x] + E[y|x] - y)^2] \\
&= E[(h(x) - E[y|x])^2 + (E[y|x] - y)^2 + 2(h(x) - E[y|x])(E[y|x] - y)] \\
&= E[(h(x) - E[y|x])^2] + E[(E[y|x] - y)^2] + 2E[(h(x) - E[y|x])(E[y|x] - y)] \\
&= E[(h(x) - h^*(x))^2] + E[(h^*(x) - y)^2] + 2E[(h(x) - h^*(x))(h^*(x) - y)]
\end{aligned}$$

下面分析$E[(h(x) - h^*(x))(h^*(x) - y)]$，注意$E(E(y|x)) = E(y)$，因此

$$\begin{aligned}
E[(h(x) - h^*(x))(h^*(x) - y)] &= E[E[(h(x) - h^*(x))(h^*(x) - y)|x]] \\
&= E[(h(x) - h^*(x))E[(h^*(x) - y)|x]]
\end{aligned}$$

来分析下$E[(h^*(x) - y)|x]$

$$\begin{aligned}
E[(h^*(x) - y)|x] &= E(h^*(x)|x) - E(y|x) \\
&= h^*(x) - h^*(x) \\
&= 0
\end{aligned}$$

所以

$$E[(h(x) - h^*(x))(h^*(x) - y)] = E[(h(x) - h^*(x))E[(h^*(x) - y)|x]] = E[(h(x) - h^*(x)) \times 0] = 0$$

综上

$$E_{out}(h) = E[(h(x) - h^*(x))^2] + E[(h^*(x) - y)^2] \geq E[(h^*(x) - y)^2]$$
$$\text{当且仅当} h(x) = h^*(x) \text{时等号成立}$$

接着证明另一个结论，首先$y = y - h^*(x) + h^*(x) = h^*(x) + \epsilon(x)$，计算$\epsilon(x)$的数学期望，注意$E(E(y|x)) = E(y)$

$$\begin{aligned} E(\epsilon(x)) &= E[E(\epsilon(x)|x)] \\ &= E[E[(y - h^*(x))|x]] \\ &= E[E(y|x) - E(h^*(x)|x)] \\ &= E[h^*(x) - h^*(x)] \\ &= 0 \end{aligned}$$

所以$\epsilon(x)$满足条件，因此结论成立。

Problem 3.9 (Page 112)

Assuming that $X^T X$ is invertible, show by direct comparison with Equation (3.4) that $E_{in}(w)$ can be written as

$$E_{in}(w) = (w - (X^T X)^{-1} X^T y)^T (X^T X)(w - (X^T X)^{-1} X^T y) + y^T (I - X(X^T X)^{-1} X^T) y$$

Use this expression for E_{in} to obtain w_{lin}. What is the in sample error? [Hint: The matrix $X^T X$ is positive definite.]

回顾等式3.4，这里把$\frac{1}{N}$这个常数略去

$$E_{in}(w) = w^T X^T X w - 2 w^T X^T y + y^T y$$

令$u = (X^T X)^{-1} X^T y, v = w - u$，那么$w = v + u$，所以

$$\begin{aligned} E_{in}(w) &= w^T X^T X w - 2 w^T X^T y + y^T y \\ &= (v + u)^T X^T X (v + u) - 2(v + u)^T X^T y + y^T y \\ &= v^T X^T X v + u^T X^T X v + v^T X^T X u + u^T X^T X u - 2 v^T X^T y - 2 u^T X^T y + y^T y \\ &= v^T X^T X v + 2 v^T X^T X u - 2 v^T X^T y + u^T X^T X u - 2 u^T X^T y + y^T y \end{aligned}$$

先看下$2 v^T X^T X u - 2 v^T X^T y$，将$u = (X^T X)^{-1} X^T y$带入

$$\begin{aligned} 2 v^T X^T X u - 2 v^T X^T y &= 2 v^T (X^T X u - y) \\ &= 2 v^T (X^T X (X^T X)^{-1} X^T y - X^T y) \\ &= 2 v^T (X^T y - X^T y) \\ &= 0 \end{aligned}$$

再看下$u^T X^T X u - 2 u^T X^T y$，将$u = (X^T X)^{-1} X^T y$带入

$$\begin{aligned} u^T X^T X u - 2 u^T X^T y &= y^T X (X^T X)^{-1} (X^T X) (X^T X)^{-1} X^T y - 2 y^T X (X^T X)^{-1} X^T y \\ &= y^T X^T (X^T X)^{-1} X y - 2 y^T X (X^T X)^{-1} X^T y \\ &= -y^T X (X^T X)^{-1} X^T y \end{aligned}$$

所以

$$
\begin{aligned}
E_{in}(w) &= v^T X^T X v - y^T X (X^T X)^{-1} X^T y + y^T y \\
&= (w - (X^T X)^{-1} X^T y)^T (X^T X)(w - (X^T X)^{-1} X^T y) + y^T (I - X(X^T X)^{-1} X^T) y
\end{aligned}
$$

接着从这个式子来分析最优解，因为$X^T X$半正定，所以

$$
\begin{gathered}
E_{in}(w) \geq y^T (I - X(X^T X)^{-1} X^T) y \\
\text{当且仅当} w - (X^T X)^{-1} X^T y = 0 \text{时等号成立} \\
\text{即} w_{lin} = (X^T X)^{-1} X^T y
\end{gathered}
$$

可以看到和之前求导的结果一样，显然求导要快很多。

Problem 3.10 (Page 112)

Exercise 3.3 studied some properties of the hat matrix $H = X(X^T X)^{-1} X^T$, where X is a N by $d+1$ matrix, and $X^T X$ is invertible. Show the following additional properties.

(a) Every eigenvalue of H is either 0 or 1. [Hint: Exercise 3.3(b).]

(b) Show that the trace of a symmetric matrix equals the sum of its eigenvalues. [Hint: Use the spectral theorem and the cyclic property of the trace. Note that the same result holds for non-symmetric matrices, but is a little harder to prove.]

(c) How many eigenvalues of H are 1? What is the rank of H? [Hint: Exercise 3.3(d).]

(a)由3.3(b)我们知道$H^K = H$，所以对于H的任意特征值λ

$$
\begin{gathered}
\lambda^K = \lambda \\
\lambda = 0 \text{或} 1
\end{gathered}
$$

(b)直接对一般的矩阵证明结论，利用$Jordan$标准型的结论即可

$$
\begin{gathered}
\text{任意方阵} A \text{可以相似于} J，\ A = PJP^{-1}，\text{其中} J \text{可以表示为如下形式} \\
J = diag(J_1, J_2, \ldots, J_k) \\
J_i = \begin{bmatrix}
\lambda_i & 1 & 0 & \ldots & 0 \\
0 & \lambda_i & 1 & \ldots & 0 \\
0 & 0 & \lambda_i & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots & 1 \\
0 & 0 & 0 & \ldots & \lambda_i
\end{bmatrix} \in R^{n_i * n_i}
\end{gathered}
$$

更具体的部分可以参考维基百科

下面仅从$trace$的角度利用这个结论

$$
\begin{aligned}
trace(A) &= trace(PJP^{-1}) \\
&= trace(P^{-1} PJ) \\
&= trace(J) \\
&= trace(diag(J_1, J_2, \ldots, J_k)) \\
&= \sum_{i=1}^{k} trace(J_i)
\end{aligned}
$$

由特征值的性质我们知道λ_i为A的特征值，而$\sum_{i=1}^{k} trace(J_i)$即为特征值之和，所以矩阵的$trace$等于特征值之和

(c)由3.3(d),特征值之和$trace(H) = d+1 =$特征值之和，因为特征值只能取0或1，所以特征值中一共有$d+1$个1。

Problem 3.11 (Page 112)

Consider the linear regression problem setup in Exercise 3.4, where the data comes from a genuine linear relationship with added noise. The noise for the different data points is assumed to be iid with zero mean and variance σ^2 . Assume that the 2nd moment matrix $\sum = E_x[xx^T]$ is non-singular. Follow the steps below to show that, with high probability, the out-of-sample error on average is

$$E_{out}(w_{lin}) = \sigma^2(1 + \frac{d+l}{N} + O(\frac{1}{N}))$$

(a) For a test point x, show that the error $y - g(x)$ is

$$\epsilon^{'} - x^T(X^TX)^{-1}X^T\epsilon$$

where $\epsilon^{'}$ is the noise realization for the test point and ϵ is the vector of noise realizations on the data .

(b) Take the expectation with respect to the test point, i.e., x and $\epsilon^{'}$, to obtain an expression for E_{out}.Show that

$$E_{out} = \sigma^2 + trace(\sum(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1})$$

[Hints: a = trace(a) for any scalar a; trace(AB) = trace(BA); expectation and trace commute.]

(c) What is $E_\epsilon[\epsilon\epsilon^T]$?

(d) Take the expectation with respect to ϵ to show that, on average,

$$E_{out} = \sigma^2 + \frac{\sigma^2}{N}trace(\sum(\frac{1}{N}X^TX)^{-1})$$

Note that : $\frac{1}{N}X^TX = \frac{1}{N}\sum_{n=1}^{N}x_nx_n^T$ is an N sample estimate of $\sum$. So : $\frac{1}{N}X^TX \approx \sum$. If : $\frac{1}{N}X^TX = \sum$, then what is E_{out} on average?

(e) Show that (after taking the expectation over the data noise) with high probability,

$$E_{out} = \sigma^2(1 + \frac{d+1}{N} + O(\frac{1}{N}))$$

[Hint: By the law of large numbers : $\frac{1}{N}X^TX$ converges in probability to $\sum$, and so by continuity of the inverse at $\sum$, $(\frac{1}{N}X^TX)^{-1}$ converges in probability to $\sum^{-1}$.]

这题实际上是对Exercise 3.4的推广，注意这里X为训练数据，x为测试数据，为了方便区分，我将x的噪音记录为$\epsilon^{'}$，原题为ϵ。

(a)同Exercise 3.4，记$y = [y_1 \dots y_N]^T, X = [x_1 \dots x_N]^T$，注意题目中给出$\epsilon = [\epsilon_1, \epsilon_2, \dots, \epsilon_N]^T$

那么

$$y = Xw^* + \epsilon$$

我们知道$w_{lin} = (X^TX)^{-1}X^Ty$

那么

$$
\begin{aligned}
g(x) &= x^T w_{lin} \\
&= x^T (X^T X)^{-1} X^T y \\
&= x^T (X^T X)^{-1} X^T (Xw^* + \epsilon) \\
&= x^T (X^T X)^{-1} X^T X w^* + X (X^T X)^{-1} X^T \epsilon \\
&= x^T w^* + x^T (X^T X)^{-1} X^T \epsilon
\end{aligned}
$$

从而

$$
\begin{aligned}
y - g(x) &= x^T w^* + \epsilon^{'} - (x^T w^* + x^T (X^T X)^{-1} X^T \epsilon) \\
&= \epsilon^{'} - x^T (X^T X)^{-1} X^T \epsilon
\end{aligned}
$$

(b)利用定义计算即可，注意这题是关于$\epsilon^{'}$, x求期望

$$
\begin{aligned}
E_{out} &= E(||y - g(x)||^2) \\
&= E(||\epsilon^{'} - x^T (X^T X)^{-1} X^T \epsilon||^2) \\
&= E((\epsilon^{'} - x^T (X^T X)^{-1} X^T \epsilon)^T (\epsilon^{'} - x^T (X^T X)^{-1} X^T \epsilon)) \\
&= E[(-\epsilon^T X (X^T X)^{-1} x + \epsilon^{'T})(\epsilon^{'} - x^T (X^T X)^{-1} X^T \epsilon)] \\
&= E[-\epsilon^T X (X^T X)^{-1} x \epsilon^{'} + \epsilon^{'T} \epsilon^{'} + \epsilon^T X (X^T X)^{-1} x x^T (X^T X)^{-1} X^T \epsilon - \epsilon^{'T} x^T (X^T X)^{-1} X^T \epsilon)] (\text{注意} \epsilon^{'} \backsim N(0, \sigma^2)) \\
&= -2E[\epsilon^T X (X^T X)^{-1} x \epsilon^{'}] + E((\epsilon^{'}))^2 + trace(E(\epsilon^T X (X^T X)^{-1} x x^T (X^T X)^{-1} X^T \epsilon)) \\
&= -2E[\epsilon^T X (X^T X)^{-1} x] E(\epsilon^{'}) + \sigma^2 + E[trace(\epsilon^T X (X^T X)^{-1} x x^T (X^T X)^{-1} X^T \epsilon)] (\text{注意} trace(AB) = trace(BA), E(\epsilon^{'}) = 0) \\
&= \sigma^2 + E[trace(x x^T (X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1})] \\
&= \sigma^2 + trace(E(x x^T (X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1})) \\
&= \sigma^2 + trace(E(x x^T)(X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1}) \\
&= \sigma^2 + trace(\sum (X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1}) (\sum = E_x[x x^T])
\end{aligned}
$$

(c)直接计算

$$
\text{注意} E(\epsilon_i \epsilon_j) = \begin{cases} \sigma^2 (i = j) \\ 0 (i \neq j) \end{cases}
$$

$$
\begin{aligned}
E_\epsilon[\epsilon \epsilon^T] &= E_\epsilon[(\epsilon_1, \epsilon_2, \dots, \epsilon_N)(\epsilon_1, \epsilon_2, \dots, \epsilon_N)^T] \\
&= E_\epsilon \begin{pmatrix} \epsilon_1 \epsilon_1 & \epsilon_1 \epsilon_2 & \dots & \epsilon_1 \epsilon_N \\ \epsilon_2 \epsilon_1 & \epsilon_2 \epsilon_2 & \dots & \epsilon_2 \epsilon_N \\ \dots & \dots & \dots & \dots \\ \epsilon_N \epsilon_1 & \epsilon_N \epsilon_2 & \dots & \epsilon_N \epsilon_N \end{pmatrix} \\
&= \begin{pmatrix} E(\epsilon_1 \epsilon_1) & E(\epsilon_1 \epsilon_2) & \dots & E(\epsilon_1 \epsilon_N) \\ E(\epsilon_2 \epsilon_1) & E(\epsilon_2 \epsilon_2) & \dots & E(\epsilon_2 \epsilon_N) \\ \dots & \dots & \dots & \dots \\ E(\epsilon_N \epsilon_1) & E(\epsilon_N \epsilon_2) & \dots & E(\epsilon_N \epsilon_N) \end{pmatrix} \\
&= \sigma^2 I
\end{aligned}
$$

(d)利用c，对b计算的E_{out}关于ϵ取数学期望可得

$$
\begin{aligned}
E_{out}^{'} &= E_\epsilon(E_{out}) \\
&= E_\epsilon[\sigma^2 + trace(\sum (X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1})] \\
&= \sigma^2 + trace[E_\epsilon(\sum (X^T X)^{-1} X^T \epsilon \epsilon^T X (X^T X)^{-1})] \\
&= \sigma^2 + \sigma^2 trace[E_\epsilon(\sum (X^T X)^{-1} X^T X (X^T X)^{-1})] \\
&= \sigma^2 + \sigma^2 trace[E_\epsilon(\sum (X^T X)^{-1})] \\
&= \sigma^2 + \frac{\sigma^2}{N} trace(\sum (\frac{1}{N} X^T X)^{-1})
\end{aligned}
$$

由计算我们知道$\frac{1}{N}X^TX=\frac{1}{N}\sum_{n=1}^{N}x_nx_n^T$是$\sum$的极大似然估计，因此$\frac{1}{N}X^TX\approx\sum$。如果$\frac{1}{N}X^TX=\sum$

$$E_{out}^{'}=\sigma^2+\frac{\sigma^2}{N}trace(I_{d+1})=\sigma^2(1+\frac{d+1}{N})$$

(e)由大数定律我们我知道$\frac{1}{N}X^TX$依概率收敛于$\sum$，所以$(\frac{1}{N}X^TX)^{-1}$依概率收敛于$\sum^{-1}$，从而有很高的的概率

$$\sum(\frac{1}{N}X^TX)^{-1}=\sum(\sum^{-1}+S)=I_{d+1}+\sum S$$

其中S为一个矩阵，那么有很高的概率

$$trace(\sum(\frac{1}{N}X^TX)^{-1})=\sigma^2(d+1+O(1))$$

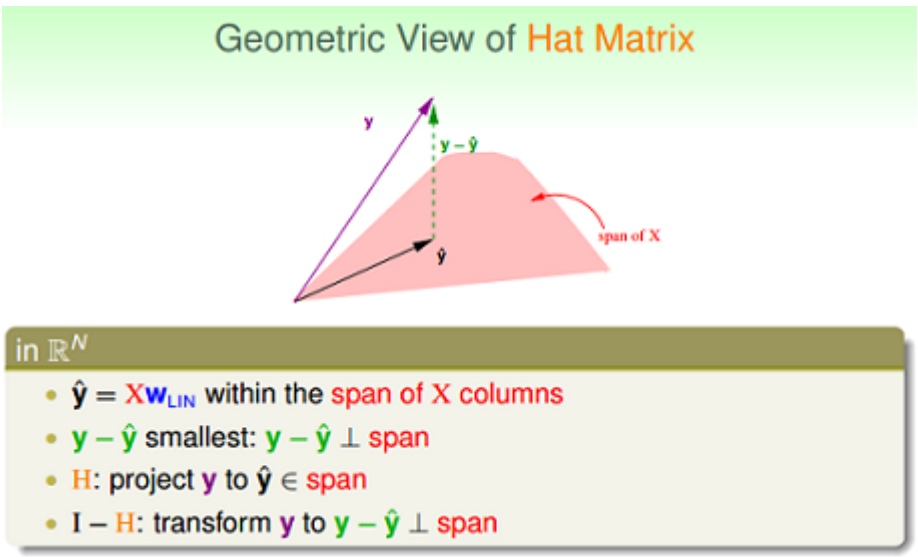
从而有很高的概率

$$E_{out}=\sigma^2+\frac{\sigma^2}{N}trace(\sum(\frac{1}{N}X^TX)^{-1})=\sigma^2(1+\frac{d+1}{N}+O(\frac{1}{N}))$$

Problem 3.12 (Page 113)

In linear regression , the in sample predictions are given by $\hat{y}=Hy$, where $H=X(X^TX)^{-1}X^T$. Show that H is a projection matrix, i.e. $H^2=H$. So $\hat{y}$ is the projection of y onto some space. What is this space?

回忆Exercies 3.3可知，$H^K=H$。关于投影可以参考林老师的课件



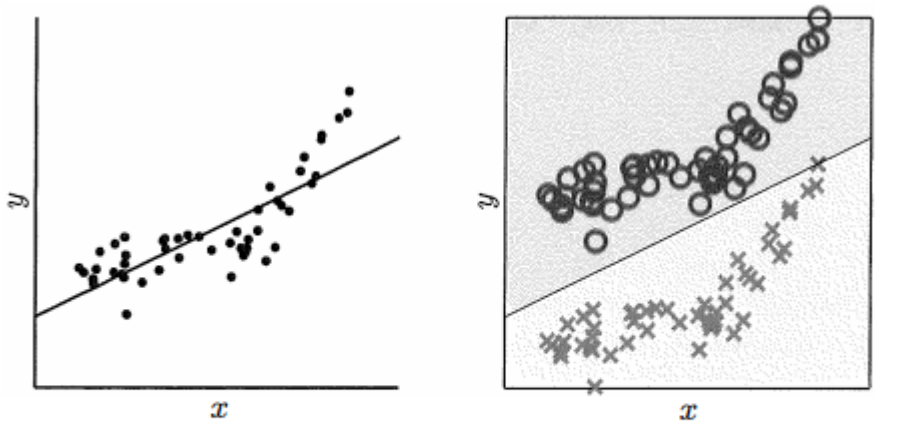
这里简单说明这几个结论。$\hat{y}=Hy=X(X^TX)^{-1}X^Ty=Xw_{lin}$，由线性代数知识我们知道$\hat{y}$属于$X$的列张成的子空间。接着考虑$y-\hat{y}$，注意$H$为对称矩阵

$$\hat{y}^T(y-\hat{y})=y^TH^T(I-H)y=y^T(H^T-H^TH)y=y^T(H-H^2)y=0$$

所以$y-\hat{y}$垂直于$\hat{y}$。

Problem 3.13 (Page 113)

This problem creates a linear regression algorithm from a good algorithm for linear classification. As illustrated , the idea is to take the original data and shift it in one direction to get the +1 data points; then , shift it in the opposite direction to get the -1 data points.



Original data for the one dimensional regression problem

Shifted data viewed as a two dimensional classification problem

More generally, The data (x_n, y_n) can be viewed as data points in R^{d+1} by treating the y value as the $(d+1)$ th coordinate.

Now, construct positive and negative points

$$D_+ = (x_1, y_1) + a, \ldots, (x_N, y_N) + a$$
$$D_- = (x_1, y_1) - a, \ldots, (x_N, y_N) - a$$

where a is a perturbation parameter. You can now use the linear programming algorithm in Problem 3.6 to separate D_+ from D_- . The resulting separating hyperplane can be used as the regression 'fit' to the original data.

(a) How many weights are learned in the classification problem? How many weights are needed for the linear fit in the regression problem?

(b) The linear fit requires weights w, where $h(x) = w^T x$. Suppose the weights returned by solving the classification problem are w_{class} . Derive an expression for w as a function of w_{class} .

(c) Generate a data set $y_n = x_n^2 + \sigma\epsilon_n$ with $N = 50$, where x_n is uniform on $[0, 1]$ and ϵ_n is zero mean Gaussian noise; set $\sigma = 0.1$. Plot D_+ and D_- for $a = [0, 0.1]^T$.

(d) Give comparisons of the resulting fits from running the classification approach and the analytic pseudo-inverse algorithm for linear regression .

题目的意思是对于回归问题的点(x_n, y_n)，有个一个偏移量a，构造两个点集
$D_+ = (x_1, y_1) + a, \ldots, (x_N, y_N) + a, D_- = (x_1, y_1) - a, \ldots, (x_N, y_N) - a$，我们对于这两个点集作分类问题，利用分类问题得到的参数来做回归。

(a)由题设知$x_n \in R^d$，所以$(x_n, y_n) = (x_n^1 \ldots x_n^d, y_n) \in R^{d+1}$，注意学习的时候还要加一个1分量，数据变为$(1, x_n, y_n) = (1, x_n^1 \ldots x_n^d, y_n) \in R^{d+1}$,从而对于分类问题我们需要学习$d+2$个权重$w = (w_0, \ldots w_{d+1}, w_{d+2})$。

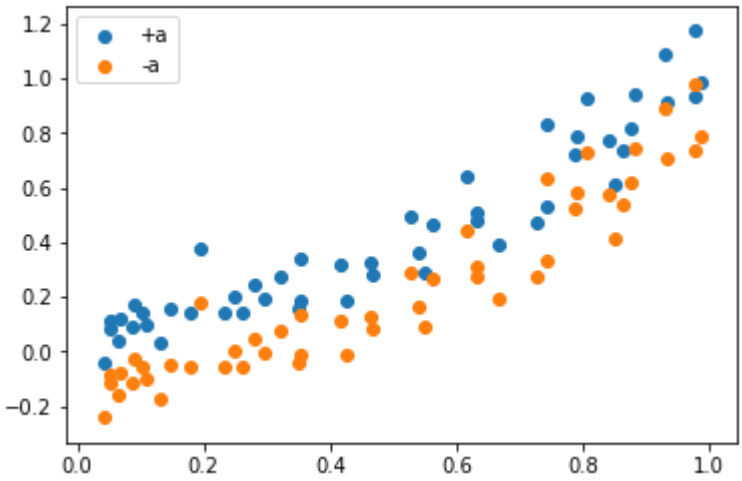
计算完w之后，我们要回到原来的回归问题，这时候只需要$d+1$个即可(不取w的最后一个分量w_{d+2})，
$w^{'} = (w_0, \ldots w_{d+1})$

(b)由(a)我们知道

$$w = (w_{class}^1, w_{class}^2 \ldots, w_{class}^{d+1})$$

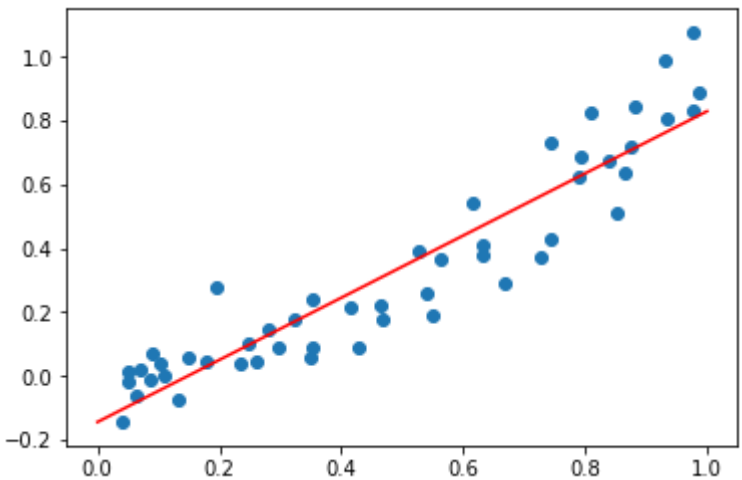
(c)编程处理

```
import numpy as np
import matplotlib.pylab as plt

def generate(n,delta):
    X=[]
    Y=[]
    data=[]
    for i in range(n):
        x=np.random.random()
        t=np.random.normal(0,1)
        y=x*x+delta*t
        data.append([x,y])
        X.append(x)
        Y.append(y)
    return X,Y,data

n=50
delta=0.1

X,Y,data=generate(n,delta)

a=np.array([0,0.1])
D1=np.array(data)+a
D2=np.array(data)-a

X1=D1[:,0]
Y1=D1[:,1]
X2=D2[:,0]
Y2=D2[:,1]

plt.scatter(X1,Y1,label='+a')
plt.scatter(X2,Y2,label='-a')
plt.legend()
plt.show()
```



(c)分别使用线性回归的公式以及这题介绍的方法求解并进行比较。先看下线性回归的效果。

```
from numpy.linalg import inv

X1=np.array([[1,i] for i in X])
Y=np.array(Y)
w1=inv(X1.T.dot(X1)).dot(X1.T).dot(Y)

x=[0,1]
y=[w1[0]+w1[1]*i for i in x]

plt.scatter(X,Y)
plt.plot(x,y,'r')
plt.show()
```



再看下这题介绍的方法。

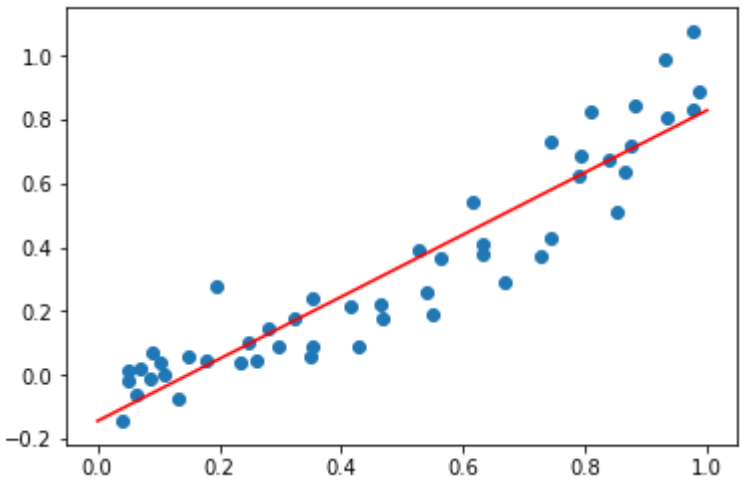
```
#构造数据
a=np.array([0,0.1])
data=np.array(data)
D=[]
for i in data:
    t1=i+a
    t1=np.array([1]+list(t1)+[1])
    t2=i-a
    t2=np.array([1]+list(t2)+[-1])
    D.append(t1)
    D.append(t2)
```

接着利用Problem 3.6介绍的优化方法求解该分类问题。

```
from cvxopt import matrix, solvers

finaldata=[]
for i in D:
    temp=list(i[:-1]*i[-1])
    finaldata.append(temp)

A=np.array(finaldata)*(-1)

n=len(finaldata)
m=len(finaldata[0])

#根据之前所述构造矩阵
a1=np.concatenate((A,(-1)*np.eye(n)),axis=1)
a2=np.concatenate((np.zeros([n,m]),(-1)*np.eye(n)),axis=1)

A1=np.concatenate((a1,a2))
c1=np.array([0.0]*m+[1.0]*n)
b1=np.array([-1.0]*n+[0.0]*n)

#带入算法求解
c1=matrix(c1)
A1=matrix(A1)
b1=matrix(b1)

sol1 = solvers.lp(c1,A1,b1)
```

```
     pcost       dcost       gap    pres   dres   k/t
 0:  2.8069e+01  1.5849e+02  7e+02  3e+00  3e+00  1e+00
 1:  4.5327e+01  6.7513e+01  6e+01  4e-01  5e-01  1e+00
 2:  4.4395e+01  4.9406e+01  1e+01  1e-01  1e-01  3e-01
 3:  4.4309e+01  4.4951e+01  1e+00  1e-02  1e-02  4e-02
 4:  4.4360e+01  4.4547e+01  4e-01  4e-03  4e-03  8e-03
 5:  4.4370e+01  4.4419e+01  1e-01  1e-03  1e-03  1e-03
 6:  4.4377e+01  4.4381e+01  9e-03  8e-05  9e-05  1e-04
 7:  4.4378e+01  4.4378e+01  9e-05  8e-07  9e-07  1e-06
 8:  4.4378e+01  4.4378e+01  9e-07  8e-09  9e-09  1e-08
Optimal solution found.
```

```
w2=list(sol1['x'])

x1=[0,1]
y1=[-(w2[0]+w2[1]*i)/w2[2] for i in x1]

plt.scatter(X,Y)
plt.plot(x,y,'r')
plt.show()
```



Problem 3.14 (Page 114)

In a regression setting, assume the target function is linear, so $f(x) = x^T w_f$, and $y = Xw_f + \epsilon$, where the entries in ϵ are zero mean, iid with variance σ^2 . In this problem derive the bias and variance as follows.

(a) Show that the average function is $\overline{g}(x) = f(x)$, no matter what the size of the data set, as long as $X^T X$ is invertible. What is the bias?

(b) What is the variance? [Hint: Problem 3.11]

这里$\overline{g}(x)$的定义要参考课本63页，对于这题来说，实际上就是求$E(w_f)$，回顾之前的结论可知$w_f = (X^T X)^{-1} X^T y$。

(a)注意$(X^T X)^{-1}$可逆，ϵ的数学期望为0

$$
\begin{aligned}
E(w_f) &= E((X^TX)^{-1}X^Ty) \\
&= E((X^TX)^{-1}X^T(Xw_f+\epsilon)) \\
&= E((X^TX)^{-1}X^TXw_f) + (X^TX)^{-1}X^TE(\epsilon) \\
&= w_f
\end{aligned}
$$

所以$\bar{g}(x) = x^Tw_f = f(x)$

$$
\begin{aligned}
bias(x) = (\bar{g}(x) - f(x))^2 = 0 \\
bias = 0
\end{aligned}
$$

(b)

$$
\begin{aligned}
var(w_f) &= E_X||x^Tw_f^{'} - x^Tw_f||^2 \\
&= E_X||x^T(X^TX)^{-1}X^Ty - x^Tw_f||^2 \\
&= E_X||x^T(X^TX)^{-1}X^T(Xw_f+\epsilon) - x^Tw_f||^2 \\
&= E_X||x^T(X^TX)^{-1}X^T\epsilon||^2 \\
&= E_X trace(\epsilon^TX(X^TX)^{-1}xx^T(X^TX)^{-1}X^T\epsilon) \\
&= E[trace(xx^T(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1})](trace(AB) = trace(BA)) \\
&= trace(E(xx^T(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1})) \\
&= trace(E(xx^T)(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1}) \\
&= trace(\sum(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1})(\sum = E_x[xx^T])
\end{aligned}
$$

可以看到，和Problem3.11非常类似

Problem 3.15 (Page 114)

In the text we derived that the linear regression solution weights must satisfy $X^TXw = X^Ty$. If X^TX is not invertible, the solution $w_{lin} = (X^TX)^{-1}X^Ty$ won 't work. In this event, there will be many solutions for w that minimize E_{in}. Here, you will derive one such solution. Let ρ be the rank of X. Assume that the singular value decomposition (SVD) of X is $X = U\Gamma V^T$ where $U \in R^{N\times\rho}$ satisfies $U^TU = I_\rho$. $V \in R^{(d+1)\times\rho}$ satisfies $V^TV = I_\rho$. and $\Gamma \in R^{\rho\times\rho}$ is a positive diagonal matrix.

(a) Show that $\rho < d+1$.

(b) Show that $w_{lin} = V\Gamma^{-1}U^Ty$ satisfies $X^TXw_{lin} = X^Ty$, and hence is a solution.

(c) Show that for any other solution that satisfies $X^TXw = X^Ty, ||w_{lin}|| < ||w||$ · That is, the solution we have constructed is the minimum norm set of weights that minimizes E_{in}·

(a)由题设我们知道X^TX不可逆，此外$X^TX \in R^{(d+1)\times(d+1)}$，所以

$$
r(X^TX) < d+1
$$

注意线性代数中的恒等式$r(X) = r(X^TX)$，因此

$$
\rho = r(X) = r(X^TX) < d+1
$$

(b)注意Γ为对角阵，所以$\Gamma^T = \Gamma$

$$
\begin{aligned}
X^TXw_{lin} &= V\Gamma^TU^TU\Gamma V^TV\Gamma^{-1}U^Ty \\
&= V\Gamma U^Ty \\
&= X^Ty
\end{aligned}
$$

(c)先对$X^TXw = X^Ty$进行处理，将$X = U\Gamma V^T$带入

$$X^TXw = V\Gamma^TU^TU\Gamma V^Tw = V\Gamma^2V^Tw$$
$$X^Ty = V\Gamma^TU^Ty = V\Gamma U^Ty$$
从而
$$V\Gamma^2V^Tw = V\Gamma U^Ty$$
两边同乘V^T，$V^TV = I_\rho$，注意Γ为正定阵，从而可逆，所以
$$V^TV\Gamma^2V^Tw = V^TV\Gamma U^Ty$$
$$\Gamma^2V^Tw = \Gamma U^Ty$$
$$V^Tw = \Gamma^{-1}U^Ty$$

接着我们将w，分解为两项

$$w = w - w_{lin} + w_{lin}$$

下面证明$w_{lin}^T(w - w_{lin}) = 0$

$$\begin{aligned} w_{lin}^T(w - w_{lin}) &= (V\Gamma^{-1}U^Ty)^T(w - w_{lin}) \\ &= y^TU\Gamma^{-1}(V^Tw - V^Tw_{lin}) \end{aligned}$$

由我们刚刚推出的等式可知，对于每个满足$X^TXw = X^Ty$的w均满足如下等式

$$V^Tw = \Gamma^{-1}U^Ty$$

所以

$$w_{lin}^T(w - w_{lin}) = y^TU\Gamma^{-1}(V^Tw - V^Tw_{lin}) = y^TU\Gamma^{-1}(\Gamma^{-1}U^Ty - \Gamma^{-1}U^Ty) = 0$$

因此

$$\begin{aligned} ||w||^2 &= ||w - w_{lin} + w_{lin}||^2 \\ &= ||w - w_{lin}||^2 + ||w_{lin}||^2 + 2w_{lin}^T(w - w_{lin}) \\ &= ||w - w_{lin}||^2 + ||w_{lin}||^2 \\ &\geq ||w_{lin}||^2 \end{aligned}$$
当且仅当$w = w_{lin}$时等号成立

Problem 3.16 (Page 115)

In Example 3.4, it is mentioned that the output of the final hypothesis $g(x)$ learned using logistic regression can be thresholded to get a ' hard ' (± 1) classification. This problem shows how to use the risk matrix introduced in Example 1 . 1 to obtain such a threshold .

Consider fingerprint verification, as in Example 1.1. After learning from the data using logistic regression, you produce the final hypothesis

$$g(x) = P[y = +1|x]$$

which is your estimate of the probability that $y = +1$. Suppose that the cost matrix is given by

		True classification	
		+1 (correct person)	−1 (intruder)
you say	+1	0	c_a
	−1	c_r	0

For a new person with fingerprint x, you compute $g(x)$ and you now need to decide whether to accept or reject the person (i.e., you need a hard classification) . So, you will accept if $g(x) \geq \kappa$, where κ is the threshold.

(a) Define the cost(accept) as your expected cost if you accept the person. Similarly define cost(reject) . Show that

$$cost(accept) = (1 - g(x))c_a$$
$$cost(reject) = g(x)c_r$$

(b) Use part (a) to derive a condition on $g(x)$ for accepting the person and hence show that

$$\kappa = \frac{c_a}{c_a + c_r}$$

(c) Use the cost matrices for the Supermarket and CIA a pplications in Example 1.1 to compute the threshold κ for each of these two cases. Give some intuition for the thresholds you get.

(a)如果accept，那么

$$cost(accept) = P[y = +1|x] \times 0 + P[y = -1|x] \times c_a = (1 - g(x))c_a$$

如果reject，那么

$$cost(reject) = P[y = +1|x] \times c_r + P[y = -1|x] \times 0 = g(x)c_r$$

(b)令只有当$cost(accept) \leq cost(reject)$时才应该接受，解这个不等式可得

$$(1 - g(x))c_a \leq g(x)c_r$$
$$c_a \leq g(x)(c_a + c_r)$$
$$\frac{c_a}{c_a + c_r} \leq g(x)$$

所以当$\frac{c_a}{c_a+c_r} \leq g(x)$时接受，$\frac{c_a}{c_a+c_r} > g(x)$时拒绝，对照题目可知

$$\kappa = \frac{c_a}{c_a + c_r}$$

(c)回顾课本上有关超市和CIA的图

		f	
		$+1$	-1
h	$+1$	0	1
	-1	10	0

Supermarket

		f	
		$+1$	-1
h	$+1$	0	1000
	-1	1	0

CIA

那么对于超市，$\kappa = \frac{1}{1+10} = \frac{1}{11}$，对于CIA，$\kappa = \frac{1000}{1000+1} = \frac{1000}{1001}$，所以只有当$g(x)$很大时才能被CIA接受，符合之前的讨论。

Problem 3.17 (Page 115)

Consider a function

$$E(u,v) = e^u + e^{2v} + e^{uv} + u^2 - 3uv + 4v^2 - 3u - 5v$$

(a) Approximate $E(u+\Delta u, v+\Delta v)$ by $\hat{E}_1(\Delta u, \Delta v)$, where $\hat{E}_1$ is the first-order Taylor's expansion of E around $(u,v) = (0,0)$. Suppose $\hat{E}_1(\Delta u, \Delta v) = a_u \Delta u + a_v \Delta v + a$. What are the values of a_u, a_v, and a?

(b) Minimize $\hat{E}_1$ over all possible $(\Delta u, \Delta v)$ such that $\|(\Delta u, \Delta v)\| = 0.5$. In this chapter, we proved that the optimal column vector $\begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix}$ is parallel to the column vector $-\nabla E(u,v)$, which is called the negative gradient direction. Compute the optimal $(\Delta u, \Delta v)$ and the resulting $E(u+\Delta u, v+\Delta v)$.

(c) Approximate $E(u+\Delta u, v+\Delta v)$ by $\hat{E}_2(\Delta u, \Delta v)$, where E_2 is the second order Taylor's expansion of E around $(u,v) = (0,0)$. Suppose

$$\hat{E}_2(\Delta u, \Delta v) = b_{uu}(\Delta u)^2 + b_{vv}(\Delta v)^2 + b_{uv}(\Delta u)(\Delta v) + b_u \Delta u + b_v \Delta v + b$$

What are the values of $b_{uu}, b_{vv}, b_{uv}, b_u, b_v$, and b?

(d) Minimize $\hat{E}_2$ over all possible $(\Delta u, \Delta v)$ (regardless of length) . Use the fact that $\nabla^2 E(u,v)|_{(0,0)}$ (the Hessian matrix at $(0,0)$) is positive definite to prove that the optimal column vector

$$\begin{bmatrix} \Delta u^* \\ \Delta v^* \end{bmatrix} = -(\nabla^2 E(u,v))^{-1} \nabla E(u,v)$$

which is called the Newton direction.

(e) Numerically compute the following values:

(i) the vector $(\Delta u, \Delta v)$ of length 0.5 along the Newton direction, and the resulting $E(u+\Delta u, v+\Delta v)$.

(ii) the vector $(\Delta u, \Delta v)$ of length 0.5 that minimizes $E(u+\Delta u, v+\Delta v)$, and the resulting $E(u+\Delta u, v+\Delta v)$. (Hint: Let $\Delta u = 0.5 sin\theta$.)

Compare the val ues of $E(u+\Delta u, v+\Delta v)$ in (b) , (e i) , and (e ii) . Briefly state your findings.

The negative gradient direction and the Newton direction are quite fundamental for designing optimization algorithms. It is important to understand these directions and put them in your toolbox for designing learning algorithms.

(a)进行一阶泰勒展开，由公式可知

$$a_u = \frac{\partial E(u,v)}{\partial u}|_{(u,v)=(0,0)} = e^u + ve^{uv} + 2u - 3v - 3|_{(u,v)=(0,0)} = -2$$
$$a_v = \frac{\partial E(u,v)}{\partial v}|_{(u,v)=(0,0)} = 2e^v + ue^{uv} - 3u + 8v - 5|_{(u,v)=(0,0)} = -3$$
$$a = E(v,v)|_{(u,v)=(0,0)} = 3$$

(b)和负梯度同向时$\hat{E}_1$最小，那么

$$\begin{aligned}\begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix} &= -k\nabla E(u,v)|_{(u,v)=(0,0)} \\ &= -k\begin{bmatrix}\frac{\partial E(u,v)}{\partial u}|_{(u,v)=(0,0)} \\ \frac{\partial E(u,v)}{\partial v}|_{(u,v)=(0,0)}\end{bmatrix} \\ &= -k\begin{bmatrix}-2 \\ -3\end{bmatrix} \\ &= k\begin{bmatrix}2 \\ 3\end{bmatrix}(k>0)\end{aligned}$$

因为$\|(\Delta u, \Delta v)\| = 0.5$，所以$\Delta u = \frac{2}{\sqrt{13}} \times 0.5, \Delta v = \frac{3}{\sqrt{13}} \times 0.5$，
$E(u+\Delta u, v+\Delta v) = E(\frac{1}{\sqrt{13}}, \frac{3}{2\sqrt{13}}) = 2.25085973499$

```
import numpy as np
def E(u,v):
    return np.exp(u)+np.exp(2*v)+np.exp(u*v)+u*u-3*u*v+4*v*v-3*u-5*v

u=1/np.sqrt(13)
v=3/(2*np.sqrt(13))
print(E(u,v))
```

```
2.25085973499
```

(c)二阶泰勒公式

$$b_{uu} = \frac{1}{2}\frac{\partial^2 E(u,v)}{\partial u^2}|_{(u,v)=(0,0)} = \frac{1}{2}\frac{\partial}{\partial u}\frac{\partial E(u,v)}{\partial u}|_{(u,v)=(0,0)} = \frac{3}{2}$$
$$b_{vv} = \frac{1}{2}\frac{\partial^2 E(u,v)}{\partial v^2}|_{(u,v)=(0,0)} = \frac{1}{2}\frac{\partial}{\partial v}\frac{\partial E(u,v)}{\partial v}|_{(u,v)=(0,0)} = 5$$
$$b_{uv} = \frac{1}{2}\frac{\partial^2 E(u,v)}{\partial u \partial v}|_{(u,v)=(0,0)} = \frac{1}{2}\frac{\partial}{\partial v}\frac{\partial E(u,v)}{\partial u}|_{(u,v)=(0,0)} = -1$$
$$b_u = a_u = -2$$
$$b_v = a_v = -3$$
$$b = a = 3$$

(d)先判断正定性

$$\nabla^2 E(u,v) = \begin{bmatrix}\frac{\partial^2 E(u,v)}{\partial u^2} & \frac{\partial^2 E(u,v)}{\partial u \partial v} \\ \frac{\partial^2 E(u,v)}{\partial u \partial v} & \frac{\partial^2 E(u,v)}{\partial v^2}\end{bmatrix} = \begin{bmatrix}3 & -2 \\ -2 & 10\end{bmatrix}$$

显然$\nabla^2 E(u,v)$正定，接着带入牛顿公式求解

$$\begin{bmatrix}\Delta u^* \\ \Delta v^*\end{bmatrix} = -(\nabla^2 E(u,v))^{-1}\nabla E(u,v) = -\begin{bmatrix}b_{uu} & b_{uv} \\ b_{uv} & b_{vv}\end{bmatrix}^{-1}\begin{bmatrix}\frac{\partial E(u,v)}{\partial u} \\ \frac{\partial E(u,v)}{\partial v}\end{bmatrix}|_{(u,v)=(0,0)} = -\begin{bmatrix}3 & -2 \\ -2 & 10\end{bmatrix}^{-1}\begin{bmatrix}-2 \\ -3\end{bmatrix}$$

带入公式求解，$\Delta u = 0.4472136, \Delta v = 0.2236068, E(u+\Delta u, v+\Delta v) = 1.87339277$

```
import numpy as np
from numpy.linalg import inv

m1=np.array([[3,-2],[-2,10]])
m2=np.array([[2],[3]])
d=inv(m1).dot(m2)

l=np.sqrt((d*d).sum())
d1=0.5*d/l

u=d1[0]
v=d1[1]

print(E(u,v))
print(d1)
```

```
[ 1.87339277]
[[ 0.4472136]
 [ 0.2236068]]
```

(e)将之前计算出来的系数带入$\hat{E}_2(\Delta u, \Delta v) = b_{uu}(\Delta u)^2 + b_{vv}(\Delta v)^2 + b_{uv}(\Delta u)(\Delta v) + b_u \Delta u + b_v \Delta v + b$，令$\Delta u = t, \Delta v = s$

$$\begin{aligned}\hat{E}_2(\Delta u, \Delta v) &= b_{uu}(\Delta u)^2 + b_{vv}(\Delta v)^2 + b_{uv}(\Delta u)(\Delta v) + b_u \Delta u + b_v \Delta v + b \\ &= 1.5(t^2) + 5(s^2) - 2st - 2t - 3s - 2 \\ &= 1.5(t-a)^2 + 5(s-b)^2 - 2(t-a)(s-b) + C\end{aligned}$$

其中a, b, C均为常数，后续会求解出来，令$t_1 = t - a, s_1 = s - b$

$$\begin{aligned}\hat{E}_2(\Delta u, \Delta v) &= 1.5(t-a)^2 + 5(s-b)^2 - 2(t-a)(s-b) + C \\ &= 1.5t_1^2 + 5s_1^2 - 2t_1 s_1 + C \\ &= \frac{3}{2}(t_1 - \frac{2}{3}s_1)^2 + (5 - \frac{2}{3})s_1^2 + C\end{aligned}$$

题目中要使得$E_2(u,v) = E_2(0,0) + \hat{E}_2(\Delta u, \Delta v)$最小，所以求$\hat{E}_2(\Delta u, \Delta v)$的最小值即可。

由上式，当$s_1 = 0, t_1 - \frac{2}{3}s_1 = 0$时，即$s_1 = t_1 = 0$时$\hat{E}_2(\Delta u, \Delta v)$最小，注意$t_1, s_1$的定义可得此时

$$t = a, s = b$$

而$\Delta u = t, \Delta v = s$，所以等号成立的条件为

$$\Delta u = a, \Delta v = b$$

接下来求解a,b

$$\begin{aligned}1.5(t-a)^2+5(s-b)^2-2(t-a)(s-b)+C&=1.5(t^2-2at+a^2)+5(s^2-2sb+b^2)-2(ts-as-bt+ab)+C\\&=1.5t^2+5s^2-2st-(3a-2b)t-(10b-2a)s+1.5a^2+5b^2-2ab+C\\&=1.5t^2+5s^2-2st-2t-3s-2\end{aligned}$$

那么

$$\begin{cases}3a-2b=2\\-2a+10b=3\end{cases}$$

$$\begin{bmatrix}3&-2\\-2&10\end{bmatrix}\begin{bmatrix}a\\b\end{bmatrix}=\begin{bmatrix}2\\3\end{bmatrix}$$

$$\begin{bmatrix}\Delta u\\\Delta v\end{bmatrix}=\begin{bmatrix}a\\b\end{bmatrix}=\begin{bmatrix}3&-2\\-2&10\end{bmatrix}^{-1}\begin{bmatrix}2\\3\end{bmatrix}$$

所以这种方法可以得出牛顿方法同样的解，这也验证了牛顿方法的正确性，计算结果同(d)。

Problem 3.18 (Page 116)

Take the feature transfrm ϕ_2 in Equation (3.13) as ϕ.

(a) Show that $d_{vc}(\mathcal{H}_\phi)\leq 6$

(b) Show that $d_{vc}(\mathcal{H}_\phi)>4$. [Hint: Exercise 3.12]

(c) Give an u pper bound on $d_{vc}(\mathcal{H}_{\phi_k})$ for $\mathcal{X}=R^d$.

(d) Define

$$\tilde{\phi_2}:x\rightarrow(1,x_1,x_2,x_1+x_2,x_1-x_2,x_1^2,x_1x_2,x_2x_1,x_2^2)\ for\ x\in R^2$$

Argue that $d_{vc}(\mathcal{H}_{\tilde{\phi_2}})=d_{vc}(\mathcal{H}_{\phi_2})$. In other words, while $\tilde{\phi_2}(\mathcal{X})\in R^9$, $d_{vc}(\mathcal{H}_{\tilde{\phi_2}})\leq 6<9$. Thus, the dimension of $\phi(\mathcal{X})$ only gives an upper bound of $d_{vc}(\mathcal{H}_\phi)$, and the exact value of $d_{vc}(\mathcal{H}_\phi)$ can depend on the components of the transform.

回顾103页的ϕ_2

$$\phi_2(x)=(1,x_1,x_2,x_1^2,x_1x_2,x_2^2)$$

(a)可以将$\phi_2(x)\in R^5$看成R^5上的感知机，所以

$$d_{vc}(\mathcal{H}_{\phi_2})\leq 6$$

(b)对于每种分类，实际上我们是在求

$$sign(w^Tx^i)=y^i(i=1,2...N)$$

其中$(x^1,\dots,x^N)$为输入数据，$(y^1,\dots,y^N)$为对应的分类($y^i\in\{1,-1\}$)，对于此题，我们取$N=5$，且求解一个更强的条件

$$w^Tx^i=y^i(i=1,2...5)$$

结合$\phi_2(x)=(1,x_1,x_2,x_1^2,x_1x_2,x_2^2)$可得

$$w_0 + w_1 x_1^i + w_2 x_2^i + w_3 (x_1^i)^2 + w_4 x_1^i x_2^i + w_5 (x_2^i)^2 = y^i (i = 1, 2...5)$$

这是关于$w_j (j = 0, 1, \ldots 5)$的六元一次方程组，且方程组有五个，所以必然有解。从而对于5个点，任意一种分类均可以表示出来，所以

$$d_{vc}(\mathcal{H}_\phi) \geq 5 > 4$$

(c)可以将$d_{vc} \in R^d$看成R^d上的感知机，所以

$$d_{vc}(\mathcal{H}_{\phi_k}) \leq d + 1$$

(d)我们每一种分类实际上对应了一个w，使得

$$sign(w^T x^i) = y^i (i = 1, 2...N)$$

其中$(x^1, \ldots, x^N)$为输入数据，$(y^1, \ldots, y^N)$为对应的分类。所以如果我们能证明ϕ_2对应的w与$\tilde{\phi_2}$对应的$\tilde{w}$可以形成一一对应关系，那么即可证明结论，因为两种特征转化下的分类可以一一对应。

先证明ϕ_2对应的w与$\tilde{\phi_2}$对应的$\tilde{w}$可以形成一一对应关系

$$\begin{aligned} w^T \phi_2 &= w_0 + w_1 x_1 + w_2 x_2 + w_3 (x_1 + x_2) + w_4 (x_1 - x_2) + w_5 (x_1^2) + w_6 (x_1 x_2) + w_7 (x_2 x_1) + w_8 (x_2^2) \\ &= w_0 + (w_1 + w_3 + w_4) x_1 + (w_2 + w_3 - w_4) x_2 + w_5 (x_1^2) + (w_6 + w_7)(x_1 x_2) + w_8 (x_2^2) \\ &= 0 \end{aligned}$$

那么w可以对应为$\tilde{w} = (w_0, w_1 + w_3 + w_4, w_2 + w_3 - w_4, w_5, w_6 + w_7, w_8)$

接着证明$\tilde{\phi_2}$对应的$\tilde{w}$与ϕ_2对应的w可以形成一一对应关系

$$\begin{aligned} \tilde{w}^T \tilde{\phi}_2 &= \tilde{w}_0 + \tilde{w}_1 x_1 + \tilde{w}_2 x_2 + \tilde{w}_3 (x_1^2) + \tilde{w}_4 (x_1 x_2) + \tilde{w}_5 (x_2^2) \\ &= \tilde{w}_0 + \tilde{w}_1 x_1 + \tilde{w}_2 x_2 + 0(x_1 + x_2) + 0(x_1 - x_2) + \tilde{w}_3 (x_1^2) + \tilde{w}_4 (x_1 x_2) + 0(x_2 x_1) + \tilde{w}_5 (x_2^2) \\ &= 0 \end{aligned}$$

所以$\tilde{w}$可以对应为$w = (\tilde{w}_0, \tilde{w}_1, \tilde{w}_2, 0, 0, \tilde{w}_3, \tilde{w}_4, 0, \tilde{w}_5)$。

因此两种特征变化可以一一对应，那么

$$d_{vc}(\mathcal{H}_{\tilde{\phi_2}}) = d_{vc}(\mathcal{H}_{\phi_2}) \leq 6 < 9$$

Problem 3.19 (Page 117)

A Transformer thinks the following procedures would work well in learning from two-dimensional data sets of any size. Please point out if there are any potential problems in the procedures:

(a) Use the feature transform

$$\phi(x) = \begin{cases} (\underbrace{0, \ldots, 0}_{n-1}, 1, 0, \ldots) & \text{if } x = x_n \\ (0, \ldots, 0) & \text{otherwise} \end{cases}$$

before running PLA.

(b) Use the feature transform ϕ with

$$\phi_n(x) = exp(-\frac{\|x - x_n\|^2}{2\gamma^2})$$

using some very small γ.

(c) Use the feature transform ϕ that consists of all

$$\phi_{i,j}(x) = exp(-\frac{\|x-(i,j)\|^2}{2\gamma^2})$$

before running PLA, with $i \in \{0, \frac{1}{100}, \ldots, 1\}$ and $j \in \{0, \frac{1}{100}, \ldots, 1\}$.

这题有点没有完全理解透，所以仅供参考。

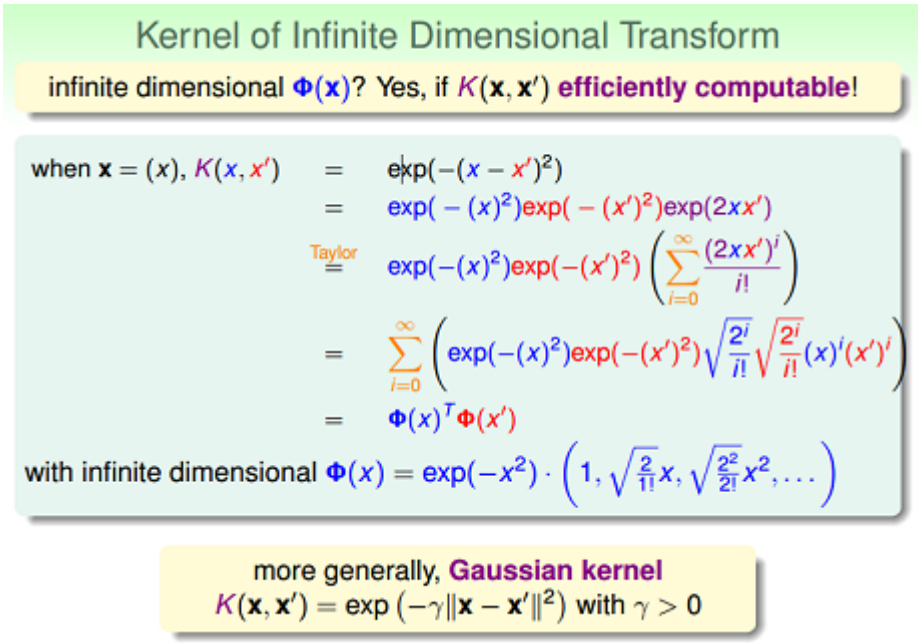
(a)这题和台大的作业3第12题很像，我们来看三个点x_1, x_2, x_3的情形

$$\phi(x_1) = (1,0,0)$$
$$\phi(x_2) = (0,1,0)$$
$$\phi(x_3) = (0,0,1)$$

可以看到这里将3个点映射到了R^3，同理可知N个点可以映射到R^N，R^N的感知机$d_{vc} = N+1$，所以N个点一定能被shatter，从而这种特征转换是好的。

(b)(c)

(c)实际上是(b)的特殊情形，这两种变换的效果都很好，这里可以参考林老师的课件理解



可以简单地理解为(b)(c)是一个无限维的特征转换，所以分类效果会很好。