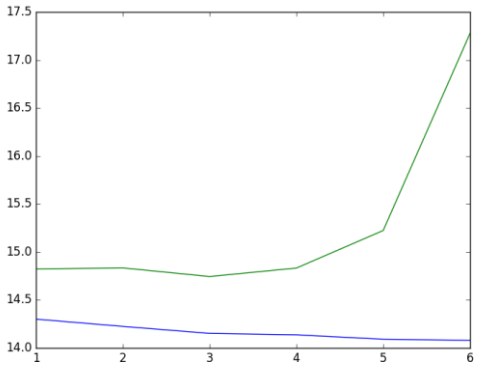
Homework 1

By Zhuo Chen,31449044
Group Partner: Yiran Xie & Lemin Tian

1.

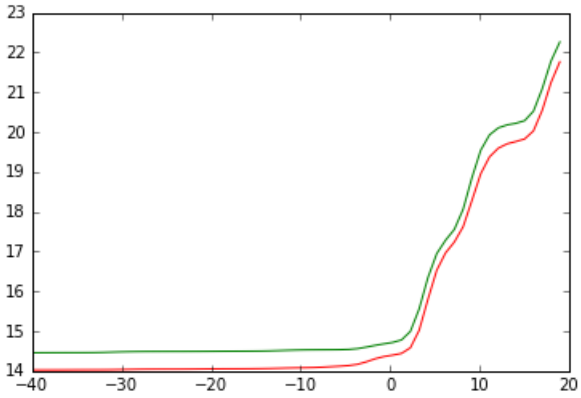
1a.1

Plot1



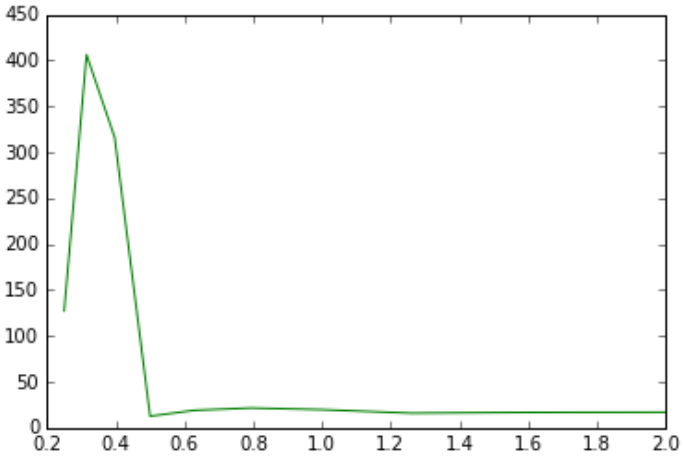
1a.2

Plot2

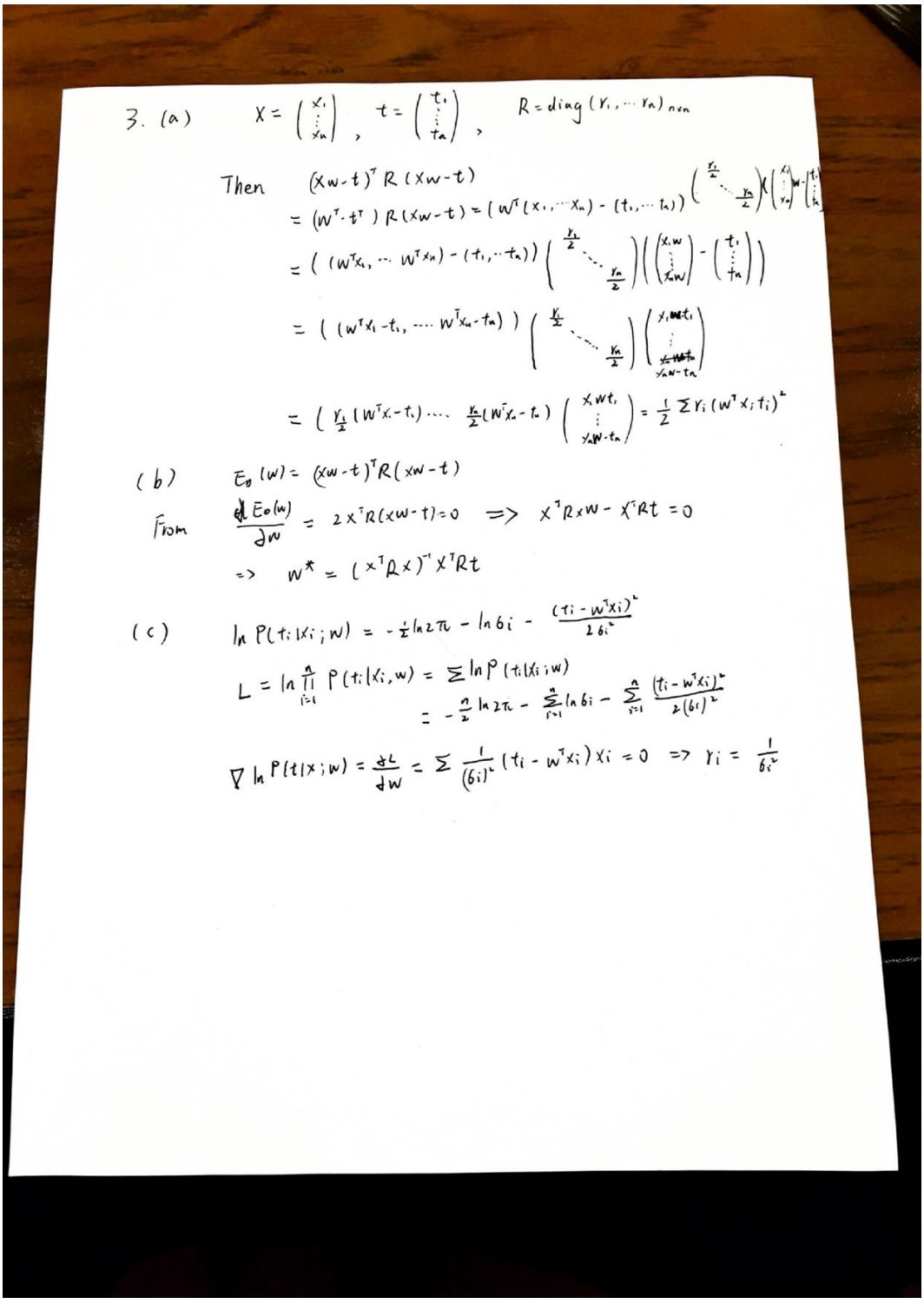


1b

Plot3



3.



3. (a) $X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, $t = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$, $R = \text{diag}(r_1, \cdots r_n)_{n \times n}$

Then $(Xw-t)^T R (Xw-t)$

$= (w^T \cdot t^T) R (Xw-t) = (w^T(x_1, \cdots x_n) - (t_1, \cdots t_n)) \begin{pmatrix} \frac{r_1}{2} & & \\ & \ddots & \\ & & \frac{r_n}{2} \end{pmatrix} \left(\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} w - \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} \right)$

$= ((w^T x_1, \cdots w^T x_n) - (t_1, \cdots t_n)) \begin{pmatrix} \frac{r_1}{2} & & \\ & \ddots & \\ & & \frac{r_n}{2} \end{pmatrix} \left(\begin{pmatrix} x_1 w \\ \vdots \\ x_n w \end{pmatrix} - \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} \right)$

$= ((w^T x_1 - t_1, \cdots w^T x_n - t_n)) \begin{pmatrix} \frac{r_1}{2} & & \\ & \ddots & \\ & & \frac{r_n}{2} \end{pmatrix} \begin{pmatrix} x_1 w - t_1 \\ \vdots \\ x_n w - t_n \end{pmatrix}$

$= \left(\frac{r_1}{2}(w^T x_1 - t_1) \cdots \frac{r_n}{2}(w^T x_n - t_n) \right) \begin{pmatrix} x_1 w - t_1 \\ \vdots \\ x_n w - t_n \end{pmatrix} = \frac{1}{2} \sum r_i (w^T x_i - t_i)^2$

(b) $E_D(w) = (Xw-t)^T R (Xw-t)$

From $\frac{\partial E_D(w)}{\partial w} = 2X^T R(Xw-t) = 0 \Rightarrow X^T R X w - X^T R t = 0$

$\Rightarrow w^* = (X^T R X)^{-1} X^T R t$

(c) $\ln P(t_i | x_i; w) = -\frac{1}{2} \ln 2\pi - \ln \sigma_i - \frac{(t_i - w^T x_i)^2}{2\sigma_i^2}$

$L = \ln \prod_{i=1}^{n} P(t_i | x_i, w) = \sum \ln P(t_i | x_i; w)$

$= -\frac{n}{2} \ln 2\pi - \sum_{i=1}^{n} \ln \sigma_i - \sum_{i=1}^{n} \frac{(t_i - w^T x_i)^2}{2(\sigma_i)^2}$

$\nabla \ln P(t | x; w) = \frac{\partial L}{\partial w} = \sum \frac{1}{(\sigma_i)^2} (t_i - w^T x_i) x_i = 0 \Rightarrow r_i = \frac{1}{\sigma_i^2}$

4.

(1)

The pre-processing is a process where we compute the median firstly and map the data based on the median. While the median of nominal variables cannot be calculated, thus this type is not suitable.

(2)

The test error of Naïve Bayes classifier is 0.250288. If I always predict the same class, for test data,, it is “non-spam”, the relative error will be 0.38562.

(3)

For spambase data, there are all ratio variables. Meanwhile, there is no nominal data in features, all features are suitable for the pre-processing.

5.

(a) $E(w) = -\ln p(t|w)$

$$= -\ln \left[\prod_{n=1}^{N} \prod_{k=0}^{K-1} p(C_k|\phi(x_n))^{1_{\{t_n=k\}}} \right]$$

$$= -\sum_{n=1}^{N} \sum_{k=0}^{K-1} 1_{\{t_n=k\}} \log \frac{\exp\{w_k^T \phi(x_n)\}}{\sum_{0}^{K-1} \exp\{w_k^T \phi(x_n)\}}$$

Thus, $\nabla_j E(w) = \frac{\partial E(w)}{\partial w_j} = \partial [- \sum_{n=1}^{N} \sum_{0}^{K-1} 1_{\{t_n=k\}} \log \frac{\exp(w_k^T \phi(x_n))}{\sum_{0}^{K-1} \exp\{w_k^T \phi(x_n)\}}] / \partial w_j$

$$= \partial L - \sum_{1}^{N} \sum_{0}^{K-1} 1_{\{t_n=k\}} [w_k^T \phi(x_n) - \log \sum_{k=0}^{K-1} \exp\{w_k^T \phi(x_n)\}] / \partial w_j$$

$$= -\sum_{1}^{N} [\phi\{x_n\} (1_{\{t_n=j\}} - \frac{\exp\{w_j^T \phi(x_n)\}}{\sum_{0}^{K-1} \exp\{w_j^T \phi(x_n)\}}$$

(b) $E^{\lambda}(w) = E(w) + \frac{\lambda}{2} \sum_{k=0}^{K-1} w_k^T w_k$

$$\nabla_{w_j} E^{\lambda}(w) = -\sum_{1}^{N} [\phi\{x_n\} (1_{\{t_n=j\}} - \frac{\exp\{w_j^T \phi(x_n)\}}{\sum_{0}^{K-1} \exp\{w_k^T \phi(x_n)\}})] + \lambda w_j$$

Codes:

```
#1a.1
import numpy as np
train = np.loadtxt("train_graphs_f16_autopilot_cruise.csv", delimiter=",", skiprows=1, usecols=(1,2,3,4,5,6,7))
temp1=train[0:3427,0:6]
phi=np.zeros((3426,37))
for i in range(0,3426):
    for j in range(0,36):
        m=j%6
        phi[i,j+1]=np.power(temp1[i,m],(j-m)/6+1)
phi[:,0]=1
ytemp1=train[:,6]
ytemp1=ytemp1.astype(float)
test = np.loadtxt("test_graphs_f16_autopilot_cruise.csv", delimiter=",", skiprows=1, usecols=(1,2,3,4,5,6,7))
temp2=test[0:2284,0:6]
phi2=np.zeros((2283,37))
for i in range(0,2283):
    for j in range(0,36):
        m=j%6
        phi2[i,j+1]=np.power(temp2[i,m],(j-m)/6+1)
phi2[:,0]=1
ytemp2=test[0:2284,6]
ytemp2=ytemp2.astype(float)
egtrain=np.zeros((6,1))
egtest=np.zeros((6,1))
for i in range(0,6):
    pt=phi[:,0:(6*(i+1)+1)]
    w=np.linalg.inv(pt.T.dot(pt)).dot(pt.T).dot(ytemp1)
    trainfit=phi[:,0:(6*(i+1)+1)].dot(w)
    egtrain1=((np.subtract(trainfit.T,ytemp1.T)).T)**2
    egtrain[i]=np.mean(egtrain1,axis=0)**0.5
    testfit=phi2[:,0:(6*(i+1)+1)].dot(w)
    egtest1=((np.subtract(testfit.T,ytemp2.T)).T)**2
    egtest[i]=np.mean(egtest1,axis=0)**0.5
from matplotlib import pyplot as plt
x=np.linspace(1,6,6)
pl.title('plot 2')
plt.plot(x,egtrain,'r')
plt.plot(x,egtest,'g')
plt.show()
#1a.2
I=np.eye(37)
regtrain=np.ones(61)
regtest=np.ones(61)
```

```
for i in range(0,61):
        wtr=np.linalg.inv(phi.T.dot(phi)+np.exp(i-40)*I).dot(phi.T).dot(ytemp1)
        wte=wtr
        trainfit1=phi.dot(wtr)
        testfit1=phi2.dot(wte)
        regtrain1=((np.subtract(trainfit1.T,ytemp1.T)).T)**2
        regtrain[i]=np.mean(regtrain1,axis=0)**0.5
        regtest1=((np.subtract(testfit1.T,ytemp2.T)).T)**2
        regtest[i]=np.mean(regtest1,axis=0)**0.5
x1=np.linspace(-40,19,61)
pl.title('plot 2')
plt.plot(x1,regtrain,'r')
plt.plot(x1,regtest,'g')
plt.show()
#1.b
testn=np.loadtxt("test_locreg_f16_autopilot_cruise.csv", delimiter=",", skiprows=1, usecols=(1,2,3,4,5,6,7))
xb=testn[:,0:6]
yb=testn[:,6]
xa=train[:,0:6]
ya=train[:,6]
T=np.logspace(-2,1,10,'true',2)
r=np.zeros((100,3426))
yfit=np.zeros((10,100))
rmse1=np.zeros((10,100))
import numpy as np
for t in range(10):
    for i in range(100) :
        for j in range(3426):
            r[i,j]=np.exp(-(np.linalg.norm(xb[i,:]-xa[j,:]))**2/(2*T[t]**2))
        R=np.sqrt(np.diag(r[i,:]))
        w=np.linalg.pinv(R.dot(xa)).dot(R).dot(ya)
        yfit[t,i]=xb[i,:].dot(w)
        rmse1[t,i]=(yfit[t,i]-yb[i])**2
        print rmse1[t,i]
rmse=np.mean(rmse1,axis=1)**0.5
from matplotlib import pyplot as plt
xxb=np.linspace(0,10,10)
pl.title('plot 3')
plt.plot(T,rmse,'g')
plt.show()

#2
import numpy as np
train = np.loadtxt("steel_composition_train.csv", delimiter=",", skiprows=1, usecols=(1,2,3,4,5,6,7,8,9))
```

```
test = np.loadtxt("steel_composition_test.csv", delimiter=",", skiprows=1, usecols=(1,2,3,4,5,6,7,8))
x1=train[:,0:8]
y1=train[:,8]
x2=test
phi=np.zeros((618,33))
for i in range(0,618):
    for j in range(0,32):
        m=j%8
        phi[i,j+1]=np.power(x1[i,m],(j-m)/8+1)
phi[:,0]=1
w=np.linalg.inv(phi.T.dot(phi)).dot(phi.T).dot(y1)
phi2=np.zeros((412,33))
for i in range(0,412):
    for j in range(0,32):
        m=j%8
        phi2[i,j+1]=np.power(x2[i,m],(j-m)/8+1)
phi2[:,0]=1
yfit=phi2.dot(w)

#4
import numpy as np
train = np.loadtxt("spambase.train", delimiter=",")
test = np.loadtxt("spambase.test", delimiter=",")
x1=train[:,0:57]
y1=train[:,57]
x2=test[:,0:57]
y2=test[:,57]
x=np.concatenate((x1,x2))
Nspam=np.sum(y1)
Nspam2=np.sum(y2)
catx1=np.zeros((2000,57))
catx2=np.zeros((2601,57))
p=np.zeros((2,57))
p1=np.zeros((2,57))
p0=np.ones((2601,1))
p10=np.ones((2601,1))

for j in range(57):
    count1=0
    count2=0
    for i in range(2000):
        if x1[i,j] < np.median(x[:,j]):
            catx1[i,j]=1
            if y1[i]==1:
```

```
                    count1=count1+1
                else:
                    count2=count2+1
            else:
                catx1[i,j]=2
        p[0,j]=(count1+1)/(Nspam+2)
        p[1,j]=1-p[0,j]
        p1[0,j]=(count2+1)/(2000-Nspam+2)
        p1[1,j]=1-p1[0,j]
for i in range(2601):
        for j in range(57):
            if x2[i,j] < np.median(x[:,j]):
                catx2[i,j]=1
            else:
                catx2[i,j]=2
for i in range(2601):
        for j in range(57):
            if catx2[i,j]==1:
                p0[i]=p0[i]*p[0,j]
                p10[i]=p10[i]*p1[0,j]
            else:
                p0[i]=p0[i]*p[0,j]
                p10[i]=p10[i]*p1[0,j]
p0=p0*(Nspam/2000)
p10=p10*(1-Nspam/2000)
check=(p0>p10)
sm=0
for i in range(2601):
        if check[i]==1:
            sm=sm
        else:
            sm=sm+1
from __future__ import division
error=sm/2601
```