HW1 CS189

1. (a) No one, Piazza

(b) I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up

Yao Cao

2. (a) ① It takes 1 day to get the first new card;

② after this, the probability to get the second new card is $\frac{39}{40}$ for one trial, until the second new card is obtained,

So the <u>number of trials N_2</u> to get the second new card follows a geometrical distribution with success probability $\underline{P_2 = \frac{39}{40}}$,

$E[N_2] = \frac{1}{P_2} = \frac{40}{39}$

③ similarly, the number of trials N_3 to get the third new card also follows a geometrical distribution with $P_3 = \frac{38}{40}$,

$E[N_3] = \frac{1}{P_3} = \frac{40}{38}$

then total total number of trials N to get all 40 cards is:

$$E[N] = E[N_1 + N_2 + \cdots + N_{40}]$$
$$= E[N_1] + E[N_2] + \cdots + E[N_{40}]$$
$$= 1 + \frac{40}{39} + \frac{40}{38} + \cdots + \frac{40}{1}$$
$$= 40\left(\frac{1}{40} + \frac{1}{39} + \frac{1}{38} + \cdots + \frac{1}{1}\right)$$
$$\approx 171$$

(b) Suppose he has n types of cards, and from one draw, the probability to draw one of the n type from d types is $\frac{n}{d}$

$$\frac{n}{d} \geq 1 - \delta$$

$$n \geq d - d\delta$$

(c) Suppose Willy draws type 1 card, the probability for this is $\frac{1}{d}$

the probabity for Charlie draws zero type 1 card is $\left(\frac{d-1}{d}\right)^n$

the probability to draw at least 1 type 1 card is $1 - \left(\frac{d-1}{d}\right)^n$

Similarly we can apply this to other types of cards, so the probability for Charlie to win is $d \cdot \left(\frac{1}{d} \cdot \left(1 - \left(\frac{d-1}{d}\right)^n\right)\right) = 1 - \left(\frac{d-1}{d}\right)^n$

(d) $1-\left(\frac{d-1}{d}\right)^n = 1-\left(\frac{d-1}{d}\right)^{\alpha d}$

$$\left(\frac{d-1}{d}\right)^{\alpha d} = \left(1-\frac{1}{d}\right)^{\alpha d} = 1 + \alpha d\left(-\frac{1}{d}\right) + \frac{\alpha d(\alpha d-1)}{2}\left(\frac{1}{d^2}\right) + \cdots + \frac{(\alpha d)^k + O(d^{k-1})}{k!}\left(-\frac{1}{d}\right)^k + \cdots$$

$$= 1 - \alpha + \frac{\alpha^2}{2} + \cdots + \frac{(-\alpha)^k}{k!} + \cdots + O\left(\frac{1}{d}\right)$$

$$= e^{-\alpha} + O\left(\frac{1}{d}\right)$$

the probability of winning :

$$P = 1-\left(\frac{d-1}{d}\right)^{\alpha d} \xrightarrow{d\to\infty} 1-e^{-\alpha}$$

(e) similar to part (c), the probability of successfully estimation of a random point is $P = 1-\left(\frac{d-1}{d}\right)^n \geq 1-\delta$

$$\Rightarrow \left(1-\frac{1}{d}\right)^n \leq \delta$$

$$n\ln\left(1-\frac{1}{d}\right) \leq \ln\delta$$

$$n \geq \frac{\ln\delta}{\ln\left(1-\frac{1}{d}\right)}$$

we need to collect at least $\frac{\ln\delta}{\ln\left(1-\frac{1}{d}\right)}$ data points

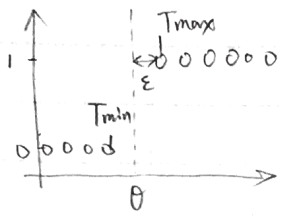
when $d\to\infty$, $\ln\left(1-\frac{1}{d}\right) \to -\frac{1}{d}$

$$\left(e^{-\frac{1}{d}} \xrightarrow{d\to\infty} 1-\frac{1}{d}\right)$$

$$n = \frac{\ln\delta}{\ln\left(1-\frac{1}{d}\right)} \xrightarrow{d\to\infty} -d\ln\delta$$

when d gets large, we need to collect $-d\ln\delta$ data points

3.(a)



$P(T_{max}-\theta > \varepsilon) = P(T_{max} > \theta+\varepsilon)$

$T_{max} > \theta+\varepsilon$ means no data points fall into $(\theta, \theta+\varepsilon)$:

$P(X \in (\theta, \theta+\varepsilon)) = \frac{\varepsilon}{1} = \varepsilon \qquad P(X \notin (\theta, \theta+\varepsilon)) = 1-\varepsilon$

all n points fall out of $(\theta, \theta+\varepsilon)$:

$P(X_1, \cdots X_n \notin (\theta, \theta+\varepsilon)) = P(X_1 \notin (\theta, \theta+\varepsilon))^n = (1-\varepsilon)^n$

so $P(T_{max}-\theta > \varepsilon) = (1-\varepsilon)^n$

similarly $P(\theta - T_{min} > \varepsilon) = P(T_{min} < \theta-\varepsilon) = (1-\varepsilon)^n$

(b) $P(|\hat{\theta}-\theta_0| < \varepsilon) = P(\theta_0-\varepsilon < \hat{\theta} < \theta_0+\varepsilon)$

$= 1 - P(\hat{\theta} > \theta_0+\varepsilon) - P(\hat{\theta} < \theta_0-\varepsilon)$

$= 1 - P(T_{max} > \theta_0+\varepsilon) - P(T_{min} < \theta_0-\varepsilon)$

$= 1 - 2(1-\varepsilon)^n \geq 1-\delta$

$2(1-\varepsilon)^n \leq \delta$

$(1-\varepsilon)^n \leq \frac{\delta}{2}$

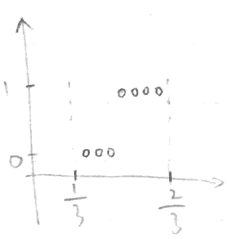
$n\ln(1-\varepsilon) \leq \ln(\frac{\delta}{2})$

$n \geq \frac{\ln(\frac{\delta}{2})}{\ln(1-\varepsilon)}$

(c) We should evenly sample n points between $(\frac{1}{3}, \frac{2}{3})$ (not including $\frac{1}{3}$ and $\frac{2}{3}$)

define $T_{min} = \max\{X_i \mid f_\theta(X_i)=0\}$, $T_{max} = \min\{X_i \mid f_\theta(X_i)=1\}$, let

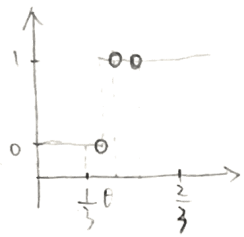
$\hat{\theta} = \frac{1}{2}(T_{min}+T_{max})$, and $T_{max}-T_{min} = \frac{\frac{1}{3}}{n+1} = \frac{1}{3(n+1)}$



$|\hat{\theta}-\theta_0| = |\frac{1}{2}(T_{min}+T_{max})-\theta_0| < \frac{1}{2}(T_{max}-T_{min}) = \frac{1}{6(n+1)} \leq \varepsilon$

$(n+1) \geq \frac{1}{6\varepsilon} \Rightarrow n \geq \frac{1}{6\varepsilon} - 1$

(d)



sample first x_1 = middle point of $(\frac{1}{3}, \frac{2}{3})$, if $f_\theta(x_1) = 1$, then sample x_2 = middle point of $(\frac{1}{3}, \frac{1}{2})$, else sample $(\frac{1}{2}, \frac{2}{3})$, every step we can cut the search domain by half, after n step, the size of search domain is $\frac{1}{3} \cdot (\frac{1}{2})^n$

the boundary of the search domain is (T_{min}, T_{max}),

let $\hat{\theta} = \frac{1}{2}(T_{min} + T_{max})$

$$|\hat{\theta} - \theta_0| = |\tfrac{1}{2}(T_{min} + T_{max}) - \theta_0| < \tfrac{1}{2}(T_{max} - T_{min}) = \tfrac{1}{3}(\tfrac{1}{2})^{n+1} \le \varepsilon$$

$$2^{n+1} \geq \frac{1}{3\varepsilon}$$

$$(n+1)\ln 2 \geq \ln(\tfrac{1}{3\varepsilon}) = -\ln(3\varepsilon)$$

$$n \geq -\frac{\ln(3\varepsilon)}{\ln 2} - 1$$

(e)

random	deterministic	adaptive
$\frac{\ln(\frac{\delta}{2})}{\ln(1-\varepsilon)}$	$\frac{1}{6\varepsilon} - 1$	$-\frac{\ln(3\varepsilon)}{\ln 2} - 1$
$\xrightarrow{\varepsilon \to 0} \frac{\ln(\frac{\delta}{2})}{-\varepsilon} = \frac{\ln(\frac{2}{\delta})}{\varepsilon}$		$= \frac{-(\ln 3 + \ln \varepsilon)}{\ln 2} - 1$
		$= \frac{\ln \frac{1}{\varepsilon}}{\ln 2} - \frac{\ln 3}{\ln 2} - 1$
n scales as $\frac{1}{\varepsilon}$	n scales as $\frac{1}{\varepsilon}$	n scales as $\ln(\frac{1}{\varepsilon})$
n scales as $\ln(\frac{1}{\delta})$		

(f) ① the more data points, the more accurate our estimation can be

② we need less data points if we are provided with extra information ($\theta \in (\frac{1}{3}, \frac{2}{3})$)

③ we need less data if we can learn adaptively.

4. (a) i) $A = \begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix}$

$$\det(A - \lambda I) = \begin{vmatrix} 2-\lambda & -4 \\ -1 & -1-\lambda \end{vmatrix} = \lambda^2 - \lambda - 2 - 4 = \lambda^2 - \lambda - 6 = (\lambda - 3)(\lambda + 2)$$

$\lambda = 3$ or $\lambda = -2$

$\boxed{\text{right}}$: For $\boxed{\lambda = 3}$

$$\begin{bmatrix} -1 & -4 \\ -1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boxed{\begin{bmatrix} -\frac{4}{\sqrt{17}} \\ \frac{1}{\sqrt{17}} \end{bmatrix}}$$

For $\boxed{\lambda = -2}$

$$\begin{bmatrix} 4 & -4 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boxed{\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}}$$

$\boxed{\text{left}}$: For $\boxed{\lambda = 3}$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} -1 & -4 \\ -1 & -4 \end{bmatrix} = 0$$

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boxed{\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}}$$

For $\boxed{\lambda = -2}$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 4 & -4 \\ -1 & 1 \end{bmatrix} = 0$$

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boxed{\begin{bmatrix} \frac{1}{\sqrt{17}} \\ \frac{4}{\sqrt{17}} \end{bmatrix}}$$

ii) $B = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$

$$\det(B - \lambda I) = \begin{vmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{vmatrix} = (3-\lambda)^2 - 1 = \lambda^2 - 6\lambda + 8 = (\lambda-2)(\lambda-4)$$

$\lambda = 2$ or $\lambda = 4$

right: For $\lambda = 2$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

For $\lambda = 4$

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

left: For $\lambda = 2$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = 0$$

$$x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

For $\lambda = 4$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} = 0$$

$$x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

iii) $A^2 x = A \cdot \lambda x = \lambda \cdot Ax = \lambda \cdot \lambda x = \lambda^2 x$ (same eigenvectors as A)

right eigenvalues: $\lambda^2 = 9$ or $\lambda^2 = 4$

$$x = \begin{bmatrix} -\frac{4}{\sqrt{17}} \\ \frac{1}{\sqrt{17}} \end{bmatrix} \qquad x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$x^T A^2 = x^T A \cdot A = \lambda x^T A = \lambda \cdot \lambda x^T = \lambda^2 x^T$$

left eigenvalues: $\lambda^2 = 9$ or $\lambda^2 = 4$

$$x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \qquad x = \begin{bmatrix} \frac{1}{\sqrt{17}} \\ \frac{4}{\sqrt{17}} \end{bmatrix}$$

iv) B^2

right and left are the same

$\lambda^2 = 4$ or $\lambda^2 = 16$

$$x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \qquad x = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

v) and vi) are calculated using scipy, code uploaded.

v) $AB = \begin{bmatrix} 2 & -10 \\ -4 & -4 \end{bmatrix}$

eigenvalue : 6 $\qquad$ -8

right : $x = \begin{bmatrix} 0.928 \\ -0.371 \end{bmatrix}$ $\qquad$ $x = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}$

left : $x = \begin{bmatrix} 0.707 \\ -0.707 \end{bmatrix}$ $\qquad$ $x = \begin{bmatrix} 0.371 \\ 0.928 \end{bmatrix}$

vi) $BA = \begin{bmatrix} 5 & -13 \\ -1 & -7 \end{bmatrix}$

eigenvalue: 6 $\qquad$ -8

right : $x = \begin{bmatrix} 0.997 \\ -0.077 \end{bmatrix}$ $\qquad$ $x = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}$

left : $x = \begin{bmatrix} 0.707 \\ -0.707 \end{bmatrix}$ $\qquad$ $x = \begin{bmatrix} 0.077 \\ 0.997 \end{bmatrix}$

b) i) $A = \begin{bmatrix} 0.994 & 0.109 \\ 0.109 & -0.994 \end{bmatrix} \begin{bmatrix} 4.497 & \\ & 1.334 \end{bmatrix} \begin{bmatrix} 0.418 & -0.909 \\ 0.909 & 0.418 \end{bmatrix}$

ii) Due to spectral Theorem, symmetric square matrix can be decomposed as $B = UDU^T$, so

$B = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 & \\ & 4 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$

iii) $A^2 = \begin{bmatrix} 8 & -4 \\ -1 & 5 \end{bmatrix} = \begin{bmatrix} -0.920 & 0.391 \\ 0.391 & 0.920 \end{bmatrix} \begin{bmatrix} 9.586 & \\ & 3.755 \end{bmatrix} \begin{bmatrix} -0.809 & 0.588 \\ 0.588 & 0.809 \end{bmatrix}$

iv) $B^2 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4 & \\ & 16 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$

v) $AB = \begin{bmatrix} 2 & -10 \\ -4 & -4 \end{bmatrix} = \begin{bmatrix} 0.935 & 0.355 \\ 0.355 & -0.935 \end{bmatrix} \begin{bmatrix} 10.778 & \\ & 4.453 \end{bmatrix} \begin{bmatrix} 0.042 & -0.999 \\ 0.999 & 0.042 \end{bmatrix}$

vi) $BA = \begin{bmatrix} 5 & -13 \\ -1 & -7 \end{bmatrix} = \begin{bmatrix} 0.906 & 0.423 \\ 0.423 & -0.906 \end{bmatrix} \begin{bmatrix} 15.302 & \\ & 3.137 \end{bmatrix} \begin{bmatrix} 0.268 & -0.963 \\ 0.963 & 0.268 \end{bmatrix}$

vii) $C = \begin{bmatrix} 3 & 1 \\ 1 & 3 \\ 2 & -4 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} -0.144 & 0.795 \\ -0.559 & 0.322 \\ 0.798 & 0.431 \\ 0.176 & -0.279 \end{bmatrix} \begin{bmatrix} 5.204 & \\ & 3.862 \end{bmatrix} \begin{bmatrix} 0.082 & -0.997 \\ 0.997 & 0.082 \end{bmatrix}$

(c) if λ is A's eigenvalue associated with x, we have

$$Ax = \lambda x$$

the ith item:

$$\lambda x_i = \sum_j A_{ij} x_j = A_{ii} x_i + \sum_{j \neq i} A_{ij} x_j$$

$$\Rightarrow (\lambda - A_{ii}) x_i = \sum_{j \neq i} A_{ij} x_j$$

On the contrary, if for every i, we have

$$(\lambda - A_{ii}) x_i = \sum_{j \neq i} A_{ij} x_j$$

$$\Rightarrow Ax = \lambda x$$

then λ is A's eigenvalue associated with x

(d) Since x is nontrivial $\Rightarrow |x_i| > 0$

$$\left|(\lambda - A_{ii})x_i\right| = \left|\sum_{j\neq i} A_{ij} x_j\right|$$

$$\Rightarrow |\lambda - A_{ii}|\cdot|x_i| = \left|\sum_{j\neq i} A_{ij} x_j\right| \leq \sum_{j\neq i} |A_{ij}|\cdot|x_j| \leq \sum_{j\neq i} |A_{ij}|\cdot|x_i|$$

$$\Rightarrow \quad |\lambda - A_{ii}| \leq \sum_{j\neq i} |A_{ij}|$$

5. (a) This is supervised learning, since labels y are given

(b) for each t, construct a feature matrix X^t

$$X^t = \begin{bmatrix} | & | & & | \\ x^1 & x^2 & \cdots & x^t \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

then we want to minimize the least-square cost

$$\min_{w^t} \| X^t w^t - y \|_2^2$$

$$\Rightarrow \hat{w}^t = (X^{t^T} X^t)^{-1} X^{t^T} y$$

use $\hat{w}^t$ to obtain $\hat{y}^t$:

$\hat{y}^t = x \hat{w}^t$ $\quad$ x: the feature vector with only the first t measurements for a new car

$x = ((x)_1, (x)_2, (x)_3 \cdots (x)_t)$

(c) $X^{t-1} = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x^{t-1} \\ | & | & & | \end{bmatrix}$ $\quad X^{t-1}\beta^t = x^t$

$$\hat{\beta}^t = ((X^{t-1})^T X^{t-1})^{-1} (X^{t-1})^T x^t$$

$$\hat{x}^t = X^{t-1}\hat{\beta}^t$$

$\tilde{X}^t = \begin{bmatrix} | & | & & | \\ \Delta x^1 & \Delta x^2 & \cdots & \Delta x^t \\ | & | & & | \end{bmatrix}$ $\quad \tilde{X}^t w^t = y$

$$\hat{w}^t = ((\tilde{X}^t)^T \tilde{X}^t)^{-1} (\tilde{X}^t)^T y$$

$$\tilde{y}^t = \tilde{X}^t \hat{w}^t$$

They are the same since to predict $\tilde{y}^t$, the data we use is $\Delta x^1, \cdots \Delta x^t$

Δx^t is just linear combination of $(x^1, \cdots x^t)$

$\tilde{y}^t$ is just linear combination of $(\Delta x^1, \cdots \Delta x^t)$, then it is also linear combination of $(x^1, \cdots x^t)$

$\hat{y}^t$ is also just linear combination of $(x^1, \cdots x^t)$

the OLS problem for both should be the same,

(d) This reminds me of Gram-Schmidt orthogonalization, where

$$\Delta x^1 = x^1$$
$$\Delta x^2 = x^2 - \text{proj}_{x^1}(x^2)$$
$$\Delta x^3 = x^3 - \text{proj}_{x^1}(x^3) - \text{proj}_{x^2}(x^3)$$
$$\vdots$$
$$\Delta x^t = x^t - \text{proj}_{x^1}(x^t) - \cdots - \text{proj}_{x^{t-1}}(x^t)$$

after this process, X is orthogonalized to be $\tilde{X}$

We know that

$\Delta x^t \perp x_1,\ \Delta x^t \perp x_2, \cdots, \Delta^t \perp x_{t-1}$

Δx^{t-1} is just linear combination of $(x_1, \cdots x_{t-1})$

so $\Delta x^t \perp \Delta x^{t-1}$

thus every column vector in $\tilde{X}$ is perpendicular to each other.

since $\tilde{X}$ is orthogonal, we can just use $\hat{w}^{t-1}$ as the first $t-1$ elements of $\hat{w}^t$

By projecting y to $\tilde{X}^{t-1}$ subspace and then substract the projection, then it cannot get any smaller by $\tilde{X}^{t-1}$ subspace, to furthur reduce it, we have to introduce new dimension $\perp$ $\tilde{X}^{t-1}$ subspace, that's why $\hat{w}^t$ should include $\hat{w}^{t-1}$

(e) $\|Y - WX\|_F^2 = \text{trace}((Y^T - X^T W^T)(Y - WX))$

$$= \text{trace}(Y^T Y + X^T W^T W X - X^T W^T Y - Y^T W X)$$

$$= \text{trace}(Y^T Y + X X^T W^T W - 2 Y^T W X)$$

$$\|Y - (W+\Delta)X\|_F^2 = \text{trace}(Y^T Y + X X^T (W+\Delta)^T (W+\Delta) - 2 Y^T (W+\Delta) X)$$

$$= \text{trace}(Y^T Y + X X^T W^T W + X X^T W^T \Delta + X X^T \Delta^T W + X X^T \Delta^T \Delta - 2 Y^T W X - 2 Y^T \Delta X)$$

$$= \|Y - WX\|_F^2 + \text{trace}(X X^T W^T \Delta + X X^T \Delta^T W - 2 Y^T \Delta X) + \text{trace}(X X^T \Delta^T \Delta)$$

$$= \|Y - WX\|_F^2 + \text{trace}(X X^T W^T \Delta + \Delta^T W X X^T - 2 X Y^T \Delta) + \text{trace}(X X^T \Delta^T \Delta)$$

$$= \|Y - WX\|_F^2 + \text{trace}(2 X X^T W^T \Delta - 2 X Y^T \Delta) + \text{trace}(X X^T \Delta^T \Delta)$$

$$= \|Y - WX\|_F^2 + \langle 2 W X X^T - 2 Y X^T, \Delta \rangle_F + \text{trace}(X X^T \Delta^T \Delta)$$

$$\Rightarrow \nabla_W (\|Y - WX\|_F^2) = 2 W X X^T - 2 Y X^T$$

let it be 0, and solve W

$$2 W X X^T - 2 Y X^T = 0$$

$$W X X^T = Y X^T$$

$$\hat{W} = Y X^T (X X^T)^{-1}$$

(f) from (e) $\hat{W} = YX^T(XX^T)^{-1}$

to be consistent with the lecture we now replace X^T with X so that $X \in \mathbb{R}^{n \times d}$, then

$$\hat{W} = YX(X^TX)^{-1} \qquad (\hat{W} \in \mathbb{R}^{k \times d}, Y \in \mathbb{R}^{k \times n})$$

from a new feature vector $x \in \mathbb{R}^{d \times 1}$, we can get a prediction vector $y \in \mathbb{R}^{k \times 1}$

$$\begin{pmatrix} y_{(1)} \\ y_{(2)} \\ \vdots \\ y_{(k)} \end{pmatrix} = \hat{W}x = YX(X^TX)^{-1}x = \begin{pmatrix} - y_1^T - \\ \vdots \\ - y_k^T - \end{pmatrix} X(X^TX)^{-1}x$$

then $y_{(j)} = y_j^T X(X^TX)^{-1}x$

$$= x^T((X^TX)^{-1})^T X^T y_j$$
$$= x^T(X^TX)^{-1}X^T y_j$$
$$= x^T \cdot \hat{w}_j$$

$$\hat{w}_j = (X^TX)^{-1}X^T y_j$$

$\hat{w}_j$ is the OLS solution that we learn from a single y_j.

thus best linear prediction for vector y is the same as solving the best linear prediction for each of its elements seperately

6. (a) this can be converted to solve the following problem

$$\min_{A,B} \| Xw - y \|^2$$

where $X = \begin{pmatrix} x_1 & u_1 \\ x_2 & u_2 \\ \vdots & \vdots \\ x_{n-1} & u_{n-1} \end{pmatrix}$ $\quad w = \begin{bmatrix} A \\ B \end{bmatrix}$ $\quad y = \begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}$

then $\begin{bmatrix} A \\ B \end{bmatrix} = (X^T X)^{-1} X^T y = \begin{bmatrix} 0.978 \\ -0.088 \end{bmatrix}$

(b) We can prove that $\underset{A,B}{\operatorname{argmin}} \sum_t \|x_{t+1} - Ax_t - Bu_t\|_2^2$ and

$\underset{W}{\operatorname{argmin}} \|Y - WX\|_F^2$ are equivalent, where

$Y \in \mathbb{R}^{3\times(n-1)}$, $W \in \mathbb{R}^{3\times 6}$, $X \in \mathbb{R}^{6\times(n-1)}$

$$Y = \begin{bmatrix} | & | & & | \\ x_2 & x_3 & \cdots & x_n \\ | & | & & | \end{bmatrix}, \quad W = [A \;\; B] \quad X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & & x_{n-1} \\ | & | & \cdots & | \\ u_1 & u_2 & & u_{n-1} \\ | & | & & | \end{bmatrix}$$

$$\left(x_t = [A \;\; B] \begin{bmatrix} | \\ x_{t-1} \\ | \\ u_{t-1} \\ | \end{bmatrix} = Ax_{t-1} + Bu_{t-1} \right)$$

$$\|Y - WX\|_F^2 = \operatorname{trace}\left((Y - WX)^T \cdot (Y - WX)\right)$$
$$= \operatorname{trace}(Y^TY + X^TWWX - 2Y^TWX)$$

$$\operatorname{trace}(Y^TY) = \operatorname{trace}\left(\begin{bmatrix} - & x_2^T & - \\ - & x_3^T & - \\ & \vdots & \\ - & x_n^T & - \end{bmatrix} \begin{bmatrix} | & | & & | \\ x_2 & x_3 & \cdots & x_n \\ | & | & & | \end{bmatrix} \right)$$

$$= \sum_{t=1}^{n-1} x_{t+1}^T x_{t+1}$$

$$WX = [A \;\; B] \begin{bmatrix} | & & | \\ x_1 & \cdots & x_{n-1} \\ | & & | \\ u_1 & & u_{n-1} \\ | & & | \end{bmatrix} = A \begin{bmatrix} | & & | \\ x_1 & \cdots & x_{n-1} \\ | & & | \end{bmatrix} + B \begin{bmatrix} | & & | \\ u_1 & \cdots & u_{n-1} \\ | & & | \end{bmatrix} = AX' + BU$$

$$\operatorname{trace}(X^TWWX) = \operatorname{trace}\left((X'^TA^T + U^TB^T)(AX' + BU)\right)$$
$$= \operatorname{trace}(X'^TA^TAX' + U^TB^TBU + X'^TA^TBU + U^TB^TAX')$$

$$\operatorname{trace}(X'^TA^TAX') = \operatorname{trace}\left(\begin{bmatrix} - & x_1^T & - \\ & \vdots & \\ - & x_{n-1}^T & - \end{bmatrix} A^TA \begin{bmatrix} | & & | \\ x_1 & \cdots & x_{n-1} \\ | & & | \end{bmatrix} \right)$$

$$= \sum_{t=1}^{n-1} x_t^T A^T A x_t$$

similary, $\text{trace}(U^T B^T B U) = \sum_{t=1}^{n-1} u_t^T B^T B u_t$

$$\text{trace}(X'^T A^T B U) = \sum_{t=1}^{n-1} x_t^T A^T B u_t$$

$$\text{trace}(U^T B^T A X') = \sum_{t=1}^{n-1} u_t^T B^T A x_t$$

$$\begin{aligned}\text{trace}(-2Y^T W X) &= -2\,\text{trace}(Y^T W X) \\ &= -2\,\text{trace}(Y^T (AX' + BU)) \\ &= -2\,\text{trace}(Y^T A X' + Y^T B U)\end{aligned}$$

$$\text{trace}(Y^T A X') = \text{trace}\left(\begin{bmatrix} - & x_2^T & - \\ & \vdots & \\ - & x_n^T & - \end{bmatrix} A \begin{bmatrix} | & & | \\ x_1 & \cdots & x_{n-1} \\ | & & | \end{bmatrix}\right)$$

$$= \sum_{t=1}^{n-1} x_{t+1}^T A x_t$$

similarly, $\text{trace}(Y^T B U) = \sum_{t=1}^{n-1} x_{t-1}^T B u_t$

$$\text{thus } \|Y - WX\|_F^2 = \sum_{t=1}^{n-1} \Big(x_{t+1}^T x_{t+1} + x_t^T A^T A x_t + x_t^T A^T B u_t + u_t^T B^T A x_t + u_t^T B^T B u_t - 2 x_{t+1}^T A x_t - 2 x_{t+1}^T B u_t \Big)$$

$$= \sum_{t=1}^{n-1} (x_{t+1}^T - x_t^T A^T - u_t^T B^T) \cdot (x_{t+1} - A x_t - B u_t)$$

$$= \sum_t \| x_{t+1} - A x_t - B u_t \|_2^2$$

then we can construct Y, W, X as shown above and solve

$$\hat{W} = Y X^T (X X^T)^{-1}$$

$$A = \begin{bmatrix} 0.152 & 0.935 & -0.001 \\ 0.039 & 0.310 & 0.874 \\ -0.526 & 0.054 & -0.470 \end{bmatrix} \qquad B = \begin{bmatrix} 0.049 & 0.206 & -0.371 \\ -0.045 & -0.929 & 0.128 \\ 0.911 & -0.471 & -0.842 \end{bmatrix}$$

(c) $\sum_i \left(-\ddot{x}_i + a x_i + b \dot{x}_i + c x_{i-1} + d \dot{x}_{i-1} + e\right)^2$ is equivalent to

$\| Xw - y \|_2^2$ where

$$X = \begin{pmatrix} | & | & | & | & | \\ x_i & \dot{x}_i & x_{i-1} & \dot{x}_{i-1} & 1 \\ | & | & | & | & | \end{pmatrix} \quad w = \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} \quad y = \begin{pmatrix} | \\ \ddot{x}_i \\ | \end{pmatrix}$$

so $w = (X^T X)^{-1} X^T y$

$$w = \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix} = \begin{pmatrix} -0.012 \\ -0.318 \\ 0.011 \\ 0.275 \\ -0.883 \end{pmatrix}$$

(d) $\ddot{x}_i = -0.012 x_i - 0.318 \dot{x}_i + 0.011 x_{i-1} + 0.275 \dot{x}_{i-1} - 0.883$

$$\approx \underbrace{0.011(x_{i-1} - x_i)}_{①} + \underbrace{0.275(\dot{x}_{i-1} - \dot{x}_i)}_{②} - \underbrace{0.043(\dot{x}_i - 65)}_{③} - \underbrace{3.678}_{④}$$

① $x_{i-1} - x_i$: the distance btw this car and the car in front of it, the farther away your front car is, the more likely you are going to step on the gas

② $\dot{x}_{i-1} - \dot{x}_i$: if the car in front of you is faster than you then you may want to be faster to catch on

③ $\dot{x}_i - 65$: the more you are above the speed limit the more likely you are going to slow down

④ consider such a situation: you are very close to the car in front of you, you have similar speed and that speed is around 65 mph, the advice to you is slow down, as the negative last term indicates.

7. (b) residual error is 422.75

w too long, in the code

(c) correct rate in trang : 0.9976

correct rate in test : 0.9981

(d) because w is obtained from training set and maybe too overfitted to training data and cannot be generalized, so we need a new data set to evaluate its performance.

In this case, our model is relatively simple and hard to overfit the data with it; on the other hand we have 11623 training data, which is a lot! So hard to overfit. To sum, the data dimension n (11623) is much larger than the model order d (400)

$$n >> d$$

so it is hard to over fit, and our model can be generalized to describe the true model.

Correct rate

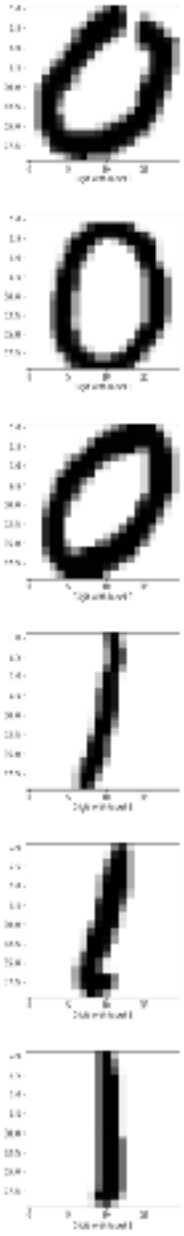
(e)

	-1 & 1	0 & 1	
training	0.9976	0.9897	without bias column
test	0.9981	0.9915	
training	0.9941	0.9941	with bias column
test	0.9962	0.9962	

Without bias column, the results if we use different classification rules are different.
With bias column, they are the same!

If we don't have the bias term, then the hyperplane seperates the two classes has to cross the origin of (x, y) space. if we use $y = -1$ and 1, which is balanced around origin, we can get a better model than $y = 0$ and 1

If we have the bias term, the hyperplane has more freedom and don't have to cross origin, so no matter what's the values of y we are training against, we can always get the best results we want, the last coefficient in w will deal with the shift in y



In [1]:
```
import numpy as np
from scipy.linalg import eig
from scipy.linalg import svd

def compute_svd(a):
    U, s, Vh = svd(a, full_matrices=False)
    print(a)
    print(U)
    print(s)
    print(Vh)

A = np.array([[2, -4], [-1, -1]])
B = np.array([[3, 1], [1, 3]])
C = np.dot(A, B)
D = np.dot(B, A)
c = np.array([[3,1],[1, 3],[2,-4],[-1,-1]])
```

In [2]:
```
w, vl, vr = eig(C,  left=True)
print(C)
print(w)
print(vr)
print(vl)
```

```
[[  2 -10]
 [ -4  -4]]
[ 6.+0.j -8.+0.j]
[[ 0.92847669  0.70710678]
 [-0.37139068  0.70710678]]
[[ 0.70710678  0.37139068]
 [-0.70710678  0.92847669]]
```

In [3]:
```
w, vl, vr = eig(D,  left=True)
print(D)
print(w)
print(vr)
print(vl)
```

```
[[  5 -13]
 [ -1  -7]]
[ 6.+0.j -8.+0.j]
[[ 0.99705449  0.70710678]
 [-0.0766965   0.70710678]]
[[ 0.70710678  0.0766965 ]
 [-0.70710678  0.99705449]]
```

In [4]:
```
compute_svd(A)
```

```
[[ 2 -4]
 [-1 -1]]
[[ 0.99402894  0.10911677]
 [ 0.10911677 -0.99402894]]
[ 4.49661478  1.33433712]
[[ 0.41785681 -0.9085129 ]
 [ 0.9085129   0.41785681]]
```

In [5]:
```
compute_svd(np.dot(A,A))
```

```
[[ 8 -4]
 [-1  5]]
[[-0.92036286  0.39106549]
 [ 0.39106549  0.92036286]]
[ 9.58630798  3.75535608]
[[-0.80885867  0.5880031 ]
 [ 0.5880031   0.80885867]]
```

In [6]:
```
compute_svd(np.dot(A,B))
```

```
[[  2 -10]
 [ -4  -4]]
[[ 0.9347217   0.35538056]
 [ 0.35538056 -0.9347217 ]]
[ 10.77805077   4.45349545]
[[ 0.04155864 -0.99913607]
 [ 0.99913607  0.04155864]]
```

In [7]:
```
compute_svd(np.dot(B,A))
```

```
[[  5 -13]
 [ -1  -7]]
[[ 0.90607303  0.42312134]
 [ 0.42312134 -0.90607303]]
[ 15.30230699   3.13678193]
[[ 0.26840683 -0.96330565]
 [ 0.96330565  0.26840683]]
```

In [8]:
```
compute_svd(c)
```

```
[[ 3  1]
 [ 1  3]
 [ 2 -4]
 [-1 -1]]
[[-0.14395373  0.79545124]
 [-0.5586541   0.32209809]
 [ 0.79770252  0.4306424 ]
 [ 0.17565196 -0.27938733]]
[ 5.20411016  3.86228397]
[[ 0.08248053 -0.99659268]
 [ 0.99659268  0.08248053]]
```

In [ ]:

In [1]:
```
##  6 System Identifcation by ordinary least squares regression, part(a)
from scipy.io import loadmat
import numpy as np
from numpy.linalg import inv
```

In [2]:
```
a = loadmat('hw01-data/system-identification/a.mat')
x = np.transpose(a['x'])
u = np.transpose(a['u'])

X = np.hstack((x[:29, :], u[:29, :]))
y =  x[1:30, :]

XT = np.transpose(X)
w = np.dot(inv(XT @ X) @ XT, y)

print(w)
```

```
[[ 0.97755214]
 [-0.08775322]]
```

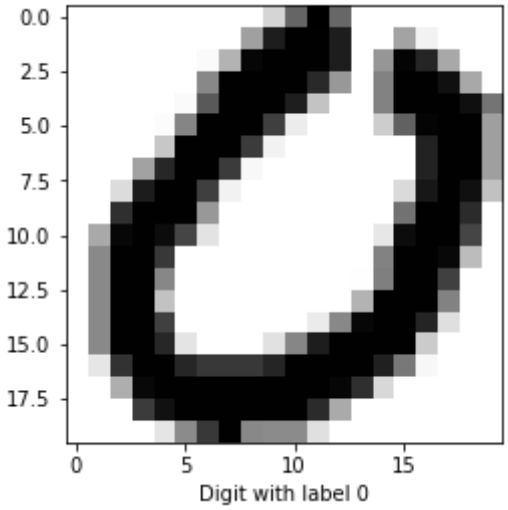
In [3]:
```
##  6 System Identifcation by ordinary least squares regression, part(b)

b = loadmat('hw01-data/system-identification/b.mat')
u = b['u']
x = b['x']
u = u.reshape(u.shape[0:2])
x = x.reshape(x.shape[0:2])
n = u.shape[0]

Y = x[1:n, :].transpose()
X = np.vstack((x[0:n-1, :].transpose(), u[0:n-1, :].transpose()))
XT = X.transpose()
W = Y @ XT @ inv(X @ XT)
A = W[:, 0:3]
B = W[:, 3:6]
print(A)
print('\n')
print(B)
```

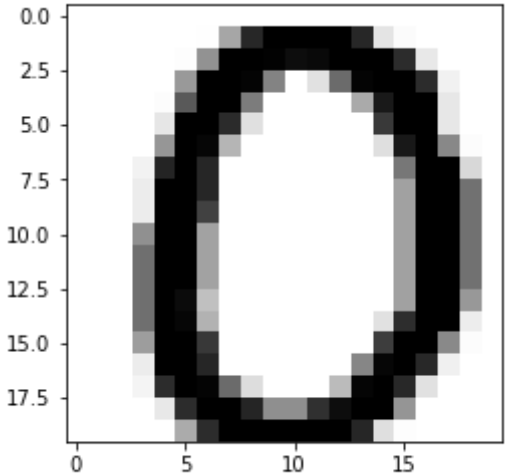
```
[[ 0.15207406  0.93480864 -0.00110243]
 [ 0.03893567  0.30958727  0.87436511]
 [-0.52552959  0.0540906  -0.47026217]]


[[ 0.04894161  0.20568264 -0.37090438]
 [-0.04524735 -0.92861546  0.12756569]
 [ 0.91096923 -0.47124981 -0.84222314]]
```

In [4]:
```
##  6 System Identifcation by ordinary least squares regression, part(c)

train = loadmat('hw01-data/system-identification/train.mat')
x = train['x'].transpose()
xd = train['xd'].transpose()
xdd = train['xdd'].transpose()
xp = train['xp'].transpose()
xdp = train['xdp'].transpose()
n = x.shape[0]

X = np.hstack((x, xd, xp, xdp, np.ones((n, 1))))
y = xdd
XT = X.transpose()
w = np.dot(inv(XT @ X) @ XT, y)
print(w)
```

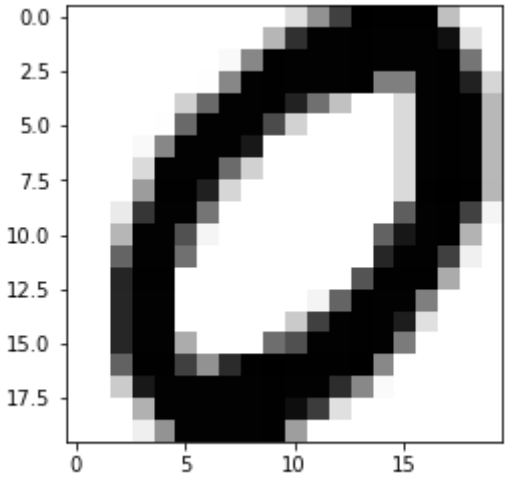
```
[[-0.01152121]
 [-0.31779341]
 [ 0.01128933]
 [ 0.27535824]
 [-0.88293502]]
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

# Load the training dataset
train_features = np.load("train_features.npy")
train_labels = np.load("train_labels.npy").astype("int8")

n_train = train_labels.shape[0]

def visualize_digit(features, label):
    # Digits are stored as a vector of 400 pixel values. Here we
    # reshape it to a 20x20 image so we can display it.
    plt.imshow(features.reshape(20, 20), cmap="binary")
    plt.xlabel("Digit with label " + str(label))
    plt.show()

# Visualize a digit
# visualize_digit(train_features[0,:], train_labels[0])

# TODO: Plot three images with label 0 and three images with label 1

def images(a):
    counter = 0
    for i in range(20):
        if train_labels[i] == a:
            counter += 1
            visualize_digit(train_features[i,:], train_labels[i])
        if  counter==3:
            break
images(0)
images(1)
```

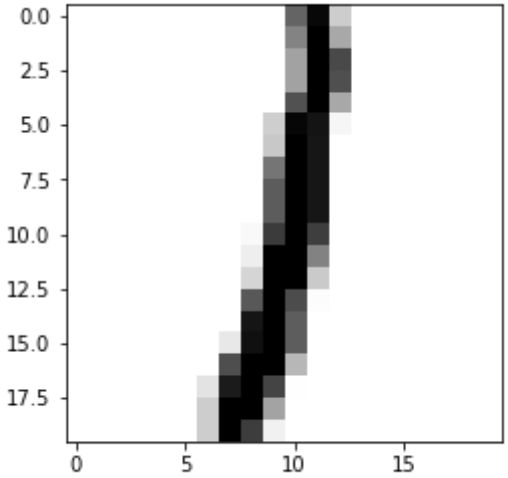




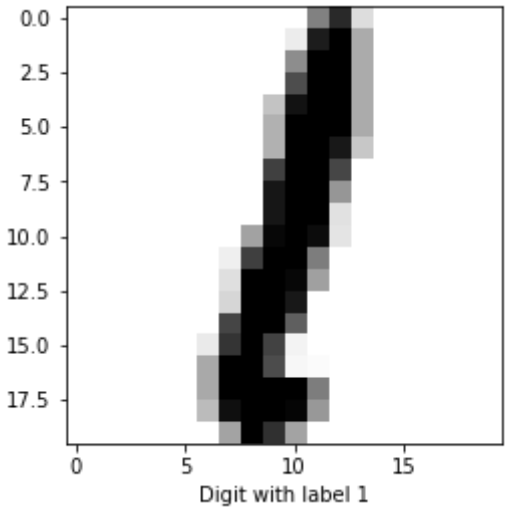
Digit with label 0

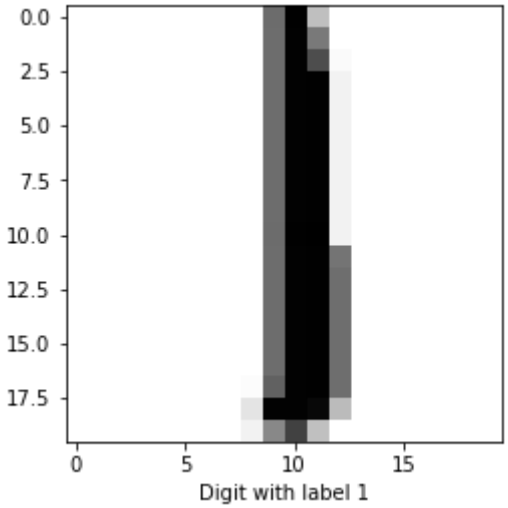


Digit with label 0



Digit with label 1


0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
Digit with label 1


0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
Digit with label 1

In [4]:
```
# Linear regression
from numpy.linalg import inv

# TODO: Solve the linear regression problem, regressing
# X = train_features against y = 2 * train_labels - 1

X = train_features
y = 2 * train_labels - 1

def get_w(X, y):
    XT = X.transpose()
    w = np.dot(inv(XT @ X) @ XT, y)
    return(w)

w = get_w(X, y)
res = np.dot(X, w) - y
res_2 = np.dot(res.transpose(), res)
# TODO: Report the residual error and the weight vector

print('the residual error is {} \n'.format(res_2))
print('w is a 400*1 vector: w = ')
print(w)
```

```
the residual error is 422.75079345703125

w is a 400*1 vector: w =
[ -3.30796123e-01   3.91724765e-01   1.48155391e-01  -1.60603136e-01
   1.03279680e-01  -1.96949169e-02  -1.27705634e-01   9.45962034e-03
  -1.71494633e-02  -5.67550585e-03  -4.69030812e-03  -1.12644807e-02
  -5.71013801e-03   4.59017232e-03   1.78760011e-02  -3.00972313e-02
   1.00369528e-02  -6.45812601e-02  -2.30415799e-02  -2.63008233e-02
  -1.63458556e-01   4.67080057e-01  -2.82919407e-03  -1.05642147e-01
  -1.80704594e-01   1.34412199e-01  -5.09417057e-03  -3.07268575e-02
  -6.59221411e-02   1.76010150e-02  -3.06512788e-02  -6.22963160e-03
   1.54067874e-02  -3.13660577e-02  -2.52744369e-03  -6.17406890e-03
  -1.02588534e-03   5.19412048e-02   3.95637155e-02   6.29596263e-02
  -3.88406903e-01  -2.16920286e-01  -9.80447531e-02   1.68329507e-01
  -5.70877790e-02   1.50778815e-02   2.43577361e-03   2.72967089e-02
   6.30412847e-02  -2.69233901e-02   7.73133337e-03  -1.71149727e-02
  -5.02606295e-02   8.06276128e-03  -6.38876110e-03  -1.26746073e-02
   1.55037120e-02  -1.06071420e-02  -7.69636221e-03  -4.18695137e-02
   5.26521444e-01   2.31917337e-01  -8.41784179e-02   9.90925729e-02
  -3.96229923e-02  -2.31645964e-02   6.78411871e-03  -5.28082922e-02
   2.26982012e-02  -1.84088089e-02   2.88220122e-03  -1.06377620e-02
   2.26956047e-02  -3.03869843e-02  -1.73668489e-02   2.12011971e-02
   3.33772749e-02  -3.48210931e-02  -2.71486416e-02  -6.14579022e-02
  -2.08384514e-01  -7.03627765e-02  -6.45077527e-02  -5.42105623e-02
   2.14532912e-02  -2.83413231e-02  -2.76354402e-02   1.08109191e-02
  -3.84216346e-02   2.86547896e-02  -2.90161967e-02   5.93165029e-03
  -7.47637823e-03  -9.09803435e-03  -1.19278338e-02  -1.55726075e-02
  -2.03939676e-02  -6.55515492e-03   4.85598072e-02  -3.95782813e-02
  -3.30486894e-02   1.84932053e-02   8.31140429e-02   9.48820449e-03
   1.04629993e-02   2.05513388e-02  -2.89785564e-02   1.20915063e-02
   1.13603324e-02  -1.36096030e-02   1.48535892e-02   4.62725386e-03
   3.68354190e-03   7.35396333e-03   2.30066702e-02  -1.49417445e-02
  -5.77552021e-02   2.55905166e-02  -1.23858154e-02  -3.55641991e-02
   5.60277700e-02  -7.86052197e-02  -5.50595187e-02  -3.80747020e-03
```

2.05794312e-02	-8.04882869e-03	7.84287229e-04	5.85546717e-03
-3.78696434e-02	2.48432159e-04	-4.85737249e-03	7.12568872e-03
-6.08517416e-03	-1.06557533e-02	-4.51316051e-02	7.35598058e-03
-5.80839813e-04	-3.72899435e-02	2.24066377e-02	-3.55857015e-02
1.25210509e-01	3.23692858e-02	2.73668617e-02	-3.78578529e-02
-2.84900293e-02	-2.55225599e-03	-3.19266394e-02	3.13730165e-02
4.19985503e-02	7.49354810e-03	2.51227245e-02	6.46016747e-03
6.20057620e-03	-2.07473338e-03	5.90182841e-03	-8.57779384e-03
2.90097855e-03	2.03040242e-02	-8.59597176e-02	-8.97464901e-03
-5.60657494e-02	5.42648807e-02	-1.96553599e-02	5.73925935e-02
-2.16799136e-02	7.91462883e-03	-5.28785214e-03	-7.76379257e-02
-5.07036373e-02	-2.74826195e-02	6.07776642e-02	4.88356575e-02
-2.40392610e-02	-4.98383641e-02	-4.93458658e-03	-4.80975434e-02
-4.14481685e-02	-6.00590520e-02	-1.63088292e-02	-5.77998385e-02
-1.04835004e-01	-5.49009591e-02	1.14539508e-02	-4.62886021e-02
9.94826853e-03	-2.29425728e-03	-5.04532829e-02	6.33369684e-02
8.83274525e-03	1.28797181e-02	5.98827526e-02	5.08481637e-02
5.55638745e-02	3.24129872e-02	1.85606256e-03	-6.51602447e-03
2.07820237e-02	4.16170210e-02	5.11917062e-02	8.70737806e-03
3.71086597e-03	-3.43105271e-02	-3.77525166e-02	2.70951204e-02
-5.89128584e-03	-1.95766538e-02	9.65390354e-03	-7.81856477e-02
2.13212445e-02	-1.35685146e-01	4.39425915e-01	-8.18718001e-02
5.43916002e-02	-3.66787612e-02	-5.72940633e-02	4.75731492e-02
-8.32982268e-03	-7.52136633e-02	-2.67496258e-02	-4.92984504e-02
3.49877179e-02	5.02439737e-02	6.05673790e-02	-3.81416082e-02
2.50256062e-03	-1.11072585e-02	-6.17939904e-02	-2.07912438e-02
7.77739659e-03	9.29733291e-02	3.28987777e-01	-5.16273789e-02
-1.47197694e-02	-8.32061023e-02	-7.43252039e-02	-7.29229301e-02
-4.12035249e-02	3.09913158e-02	1.29102431e-02	-2.84672678e-02
-3.92790325e-03	-1.00521073e-01	-3.35057825e-02	1.55609846e-03
-7.21260905e-04	-2.80618668e-02	5.95020130e-03	3.56642902e-02
1.08798593e-02	1.07424021e-01	4.42860126e-02	-1.00261532e-03
1.39009580e-03	-8.00528973e-02	-3.36570404e-02	-7.74169713e-03
-8.94960016e-03	-2.00045556e-02	2.30861567e-02	-7.37616941e-02
8.12910497e-03	4.37760875e-02	-3.59041095e-02	-3.68333161e-02
-6.17185645e-02	-3.72650661e-02	-5.80597967e-02	-1.79026444e-02
5.28743528e-02	7.61481002e-04	4.68757376e-02	3.64771038e-02
3.71705405e-02	3.71804237e-02	3.30805629e-02	-3.52992415e-02
-8.89341757e-02	-2.42443606e-02	-7.89628774e-02	5.99945411e-02
-8.14004987e-02	-3.61125022e-02	-1.25761181e-02	-2.42130980e-02
5.14357537e-03	1.78405661e-02	2.22027320e-02	2.82184780e-03
3.35302949e-02	-4.93126549e-02	2.01935694e-02	3.94619480e-02
-2.18398906e-02	-9.06860158e-02	-4.52998281e-03	-2.45162938e-02
1.84685104e-02	-1.53115913e-02	-1.20573044e-02	-1.18375078e-01
2.88001671e-02	-1.90547481e-02	-5.65437078e-02	1.42882764e-02
-2.23247074e-02	-8.45870189e-03	-2.25760341e-02	2.53214911e-02
-3.29714268e-02	2.33236756e-02	1.05693061e-02	1.18442606e-02
3.81864794e-02	2.81872228e-02	-2.45874375e-03	5.40287420e-03
-4.95716929e-04	1.93904787e-02	5.34845442e-02	-5.50350510e-02
-1.27018005e-01	-1.13497362e-01	7.43942559e-02	-6.41146451e-02
-2.47328319e-02	1.66992620e-02	2.09305733e-02	1.21586993e-02
4.45517376e-02	-1.40924025e-02	1.58408955e-02	2.01890692e-02
-1.64979994e-02	1.85777508e-02	2.12734863e-02	7.97321647e-03
-3.30512077e-02	-1.31743565e-01	-5.20438924e-02	-6.18480146e-03
4.09973860e-02	6.37193769e-02	1.58209503e-02	8.63967761e-02
4.42898162e-02	-2.31197551e-02	-8.82010534e-03	-3.39905694e-02
1.49171129e-02	4.80811819e-02	1.25215128e-02	-2.58592516e-03

```
  3.72425020e-02  -2.24012323e-03   5.62974922e-02   9.65552405e-03
  1.01853073e-01   8.83490518e-02  -4.26429659e-02   2.76069760e-01
  5.16162813e-02   5.58910817e-02   1.50369890e-02  -7.86101259e-03
 -1.79310963e-02   5.78273460e-03  -1.39468256e-02  -2.94634756e-02
 -7.18816891e-02  -6.41089454e-02   2.71457061e-03  -2.76246220e-02
 -3.69601846e-02   7.89900869e-03   7.59387240e-02  -8.44904035e-03
  1.20694041e-02   1.71198979e-01  -3.02480102e-01   2.80249119e-03
 -2.34687328e-03  -3.16523015e-02   3.73742916e-03  -3.21977921e-02
  1.20266862e-02  -4.40902263e-03   1.04535855e-02   3.74992751e-03
  1.62230078e-02  -3.36626545e-03  -4.06874008e-02  -1.61136314e-02
 -6.13355637e-03  -3.09129804e-02  -5.01953661e-02   1.58628821e-02
 -1.27862543e-01   1.68132693e-01  -2.77890831e-01   4.19037044e-02]
```

In [5]:

```
# Load the test dataset
# It is good practice to do this after the training has been
# completed to make sure that no training happens on the test
# set!

from numpy import logical_xor, logical_not
test_features = np.load("test_features.npy")
test_labels = np.load("test_labels.npy").astype("int8")

n_test = test_labels.shape[0]

# TODO: Implement the classification rule and evaluate it
# on the training and test set

def correct_rate(features, labels, n, w, rule):
    predict = (np.dot(features, w) > rule)
    correct = logical_not(logical_xor(predict, labels))
    return float(sum(correct))/n


print('the correct pertentage by this model in the training \n \
        set is {} \n'.format(correct_rate(train_features, train_labels, n_tr

print('the correct pertentage by this model in the test \n \
        set is {} \n'.format(correct_rate(test_features, test_labels, n_test
```

```
the correct pertentage by this model in the training
         set is 0.9975909833949926

the correct pertentage by this model in the test
         set is 0.9981087470449173
```

In [6]:
```
# TODO: Try regressing against a vector with 0 for class 0
# and 1 for class 1

w2 = get_w(X, train_labels)
print('0 for 0, 1 for 1 model: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(train_features, train_labels, n_tr

print('0 for 0, 1 for 1 model: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(test_features, test_labels, n_test
```

```
0 for 0, 1 for 1 model: the correct pertentage by this model in the train
ing
         set is 0.9896756431213972

0 for 0, 1 for 1 model: the correct pertentage by this model in the test
         set is 0.9914893617021276
```

In [7]:
```
# TODO: Form a new feature matrix with a column of ones added
# and do both regressions with that matrix

train_features_1 = np.hstack((train_features, np.ones((n_train, 1))))
test_features_1 = np.hstack((test_features, np.ones((n_test, 1))))

w3 = get_w(train_features_1, 2*train_labels-1)
print('train with bias column: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(train_features_1, train_labels, n_

print('train with bias column: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(test_features_1, test_labels, n_te


w4 = get_w(train_features_1, train_labels)
print('train with bias column & 0 for 0 1 for 1: the correct pertentage by t
        set is {} \n'.format(correct_rate(train_features_1, train_labels, n_

print('train with bias column & 0 for 0 1 for 1: the correct pertentage by t
        set is {} \n'.format(correct_rate(test_features_1, test_labels, n_te
```

```
train with bias column: the correct pertentage by this model in the train
ing
         set is 0.9941495311021251

train with bias column: the correct pertentage by this model in the test
         set is 0.9962174940898345

train with bias column & 0 for 0 1 for 1: the correct pertentage by this
model in the training
         set is 0.9941495311021251

train with bias column & 0 for 0 1 for 1: the correct pertentage by this
model in the test
         set is 0.9962174940898345
```

In [8]:
```
# Logistic Regression

# You can also compare against how well logistic regression is doing.
# We will learn more about logistic regression later in the course.

import sklearn.linear_model

lr = sklearn.linear_model.LogisticRegression()
lr.fit(X, train_labels)

test_error_lr = 1.0 * sum(lr.predict(test_features) != test_labels) / n_test
print(1-test_error_lr)
```

```
0.999527186761
```

In [ ]: