Machine Learning **(Due: 20/11/20)**

Homework #3: Referred Answers

Instructor: Lin, Hsuan Tien *Name:* Hao-Cheng Lo, *Id:* D08227104

Problem 1:

[b] is correct.

By the equation provided, $0.1^2(1-\dfrac{11+1}{N}) \geq 0.006 \rightarrow N \geq 30$.

Problem 2:

[a] is correct.

To prove the normal equation always has at least one solution. We can find that w is in the column space of X^TX denoted $C(X^TX)$ and that similarly y is in the column space of X^T denoted $C(X^T)$. If w and y in the same column space and thus y can be represented as a linear combination of w, then the normal equation always has at least one solution (unique solution or many solutions). Based on linear algebra, $C(X^TX)$ is the orthogonal complement of the left null-space of X^TX denoted as $(N((X^TX)^T))^\perp$, which $= (N(X^TX))^\perp$. Because $\forall$ vector v (1) if $Xv = 0$ then $X^TXv = 0$ and (2) if $X^TXv = 0 \to v^TX^TXv = 0 \to ||Xv|| = 0$ then $Xv = 0$, $(N(X^TX))^\perp = (N(X))^\perp$. Plus, $(N(X))^\perp = C(X^T)$. Taken together, $C(X^TX) = C(X^T)$, thus the normal equation always has at least one solution (unique solution or many solutions).

For other choices, solutions of normal equation are related to $\nabla E_{in} = 0$ which does not guarantee $E_{in} = 0$ or vice versa.

Problem 3:

[c] is correct because H_c is different from H. For each choice, here provides my explanations.

[a] $H_a = (2X)((2X)^T(2X))^{-1}(2X)^T = 2X(4^{-1})(X^TX)^{-1}2X^T = X(X^TX)^{-1}X^T = H.$

[b] Let D be the square diagonal right-matrix that multiplies each of the i-th column of any matrix by i, that is

$$D = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 2 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & n \end{bmatrix}$$

, which is non-singular. Accordingly,

$$H_b = (XD)((XD)^T(XD))^{-1}(XD)^T = XD(D^TX^TXD)^{-1}D^TX^T = XDD^-(X^TX)^{-1}D'^-D'X' = X(X^TX)^{-1}X^T = H.$$

[c] Let D be the square diagonal left-matrix that multiplies each of the n-th row of X by $\frac{1}{n}$, that is

$$D = \begin{bmatrix} 1/1 & 0 & \cdots & 0 \\ 0 & 1/2 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & 1/n \end{bmatrix}$$

, which is non-singular. Accordingly,

$$H_c = (DX)((DX)^T(DX))^{-1}(DX)^T = DX(X^TD^TDX)^{-1}X^TD^T = DX(X^TD^2X)^{-1}X^TD \neq X(X^TX)^{-1}X^T = H.$$

[d] Let E be the square column operation right-matrix that adds three randomly-chosen columns i, j, k to column 1 of X, that is

$$E = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_1 & 1 & 0 & 0 \\ \vdots & 0 & \ddots & \vdots \\ a_{n-1} & 0 & \cdots & 1 \end{bmatrix}$$, where randomly three $a_i = 1,\ \forall i \in 1, ..., n-1$ and others are 0.

, which is non-singular. Accordingly,

$$H_d = (XE)((XE)^T(XE))^{-1}(XE)^T = XE(E^TX^TXE)^{-1}E^TX^T = XEE^-(X^TX)^{-1}E'^-E'X' = X(X^TX)^{-1}X^T = H.$$

[e] This is incorrect cause H_c is different from H.

Problem 4:

[e] is correct.

For the first statement, we can imagine θ is produced by tossing the coin with infinite times and counted by the number of successes/heads. And, ν is calculated from a n-size sample and its corresponding the number of heads. Actually, ν and θ can be mapping to the sample-population relation in regard to that each experiment is i.i.d. to Bernoulli random variable. Thus, the relation of ν and θ is bounded by Hoeffding's inequality. Given Hoeffding's inequality, the first statement is correct.

For the second statement, the observation is with k heads in n Bernoulli trials. The pdf of n Bernoulli trials (binomial distribution) is

$$f(x) = \left(\frac{n!}{x!(n-x)!}\right) \theta^x (1-\theta)^{n-x}$$

The likelihood function is

$$L(\theta) = \prod_{i=1}^n f(y_i) = \prod_{i=1}^n \left(\frac{n!}{y_i!(n-y_i)!}\right) \theta^{y_i} (1-\theta)^{n-y_i}$$

The log likelihood function can be presented as

$$\ell(\theta) = \log\theta \sum_{i=1}^n y_i + \log(1-\theta) \sum_{i=1}^n (1-y_i)$$

Let the first derivation of log likelihood function be 0

$$\frac{\partial \ell(\theta)}{\partial \theta} = \frac{\sum_{i=1}^n y_i}{\theta} - \frac{\sum_{i=1}^n (1-y_i)}{1-\theta} \overset{\text{set}}{=} 0$$

Thus,

$$\hat{\theta} = \frac{\sum_{i=1}^n y_i}{n} = \nu$$

The second statement is correct.

For the third statement,

$$E_{in}(\hat{y}) = \frac{1}{N}||\hat{y}\mathbf{1}_{n\times 1} - \mathbf{y}||^2 = \frac{1}{N}(\hat{y}^2\mathbf{1}^T\mathbf{1} - 2\hat{y}\mathbf{1}^T\mathbf{y} + \mathbf{y}^T\mathbf{y});\ \nabla E_{in}(\hat{y}) = \frac{1}{N}(2\hat{y}\mathbf{1}^T\mathbf{1} - 2\mathbf{1}^T\mathbf{y}) = \frac{1}{N}(2\hat{y}N - 2\sum y)$$

Let $\nabla E_{in}(\hat{y}) = \dfrac{1}{N}(2\hat{y}N - 2\sum y) = 0$, thus $\hat{y} = \dfrac{1}{N}\sum y = \nu$. Plus the second derivation of E_{in} is positive $= 2$. Thus, ν minimizes $E_{in}(\hat{y})$ and the third statement is correct.

For the forth statement,

$$\text{At } \hat{y} = 0,\ -\nabla E_{in}(\hat{y} = 0) = -\frac{1}{N}(2\times 0 - 2\sum y) = 2\frac{1}{N}\sum y = 2\nu$$

Thus, the forth statement is correct.

Taken together, there are 4 statements being correct.

Problem 5:

[a] is correct.

The pdf of $[0, \theta]$ uniform distribution is

$$f(x) = \frac{1}{\theta - 0} = \frac{1}{\theta}$$

The likelihood function is

$$L(\theta) = \prod_{i=1}^{n} f(y_i) = \prod_{i=1}^{n} \frac{1}{\theta} = \frac{1}{\theta^n} \text{ (Thus, the answer is [a]. The following is MLE of } \theta\text{)}$$

The log likelihood function can be presented as

$$\ell(\theta) = \log \theta^{-n} = -n \log \theta$$

The first derivation of log likelihood function is

$$\frac{\partial \ell(\theta)}{\partial \theta} = \frac{-n}{\theta}$$

Since the first derivation of log likelihood function is monotonously increasing when θ increasing, indicating

$$\hat{\theta}_{MLE} = \max(y_1, ..., y_N)$$

Problem 6:

[b] is correct.

Assume $y = +1$. When $w^T x > 0$, the point-wise error should be 0; when $w^T x < 0$, the point-wise error should increase with $|w^T x|$ raising. Thus, $\text{err}(w, x, y) = \max(0, -yw^T x)$.

Problem 7:

[a] is correct.

The proof: $-\nabla_w \mathrm{err}_{\exp}(w, x, y) = -\nabla_w \exp(-yw^T x) = -(-yx) \exp(-yw^T x) = +yx \exp(-yw^T x)$.

Problem 8:

[b] is correct.

The proof: Given $E(u+v) \approx E(u) + b_E(u)^T v + (1/2)v^T A_E(u)v$, let $\nabla_v E(u+v) \approx \nabla_v(E(u) + b_E(u)^T v + (1/2)v^T A_E(u)v) = b_E(u)^T + A_E(u)v = 0$. Thus, $v = -(A_E(u))^{-1} b_E(u)^T$, noted that $\because$ $A_E(u)$ is positive definite $(A_E(u))^{-1}$ exists.

Problem 9:

[b] is correct.

Given $\nabla_w E_{in}(w) = \dfrac{2}{N}(X^T Xw - X^T y)$, the Hessian matrix $A_E(w) = \nabla_w^2 E_{in}(w) = \nabla_w(\dfrac{2}{N}(X^T Xw - X^T y)) = \dfrac{2}{N} X^T X$.

Problem 10:

[b] is correct.

Denote $w_p^T x = a_p$ and $h_p(x) = S(a_p)$ where $p \in 1, ..., K$.

For given $s = t$, $\dfrac{\partial S(a_s)}{\partial a_t} = \dfrac{\partial[\exp(a_s)/\sum \exp(a_i)]}{\partial a_t} = \dfrac{\exp(a_s)\sum\exp(a_i) - \exp(a_s)\exp(a_t)}{\sum\exp(a_i)\sum\exp(a_i)} = S(a_s)(1 - S(a_t))$

For given $s \neq t$, $\dfrac{\partial S(a_s)}{\partial a_t} = \dfrac{\partial[\exp(a_s)/\sum \exp(a_i)]}{\partial a_t} = \dfrac{0 - \exp(a_s)\exp(a_t)}{\sum\exp(a_i)\sum\exp(a_i)} = -S(a_s)S(a_t)$.

Thus, for any s and t, $\dfrac{\partial S(a_s)}{\partial a_t} = S(a_s)([\![s = t]\!] - S(a_t))$.

$\text{err}(W, x, y) = -\ln h_y(x) = -\ln S(a_y)$

$\rightarrow \nabla_{a_k}\text{err}(W, x, y) = -(S(a_y))^{-1}\dfrac{\partial S(a_y)}{\partial a_k} = -(S(a_y))^{-1}S(a_y)([\![y = k]\!] - S(a_k)) = (S(a_k) - [\![y = k]\!])$

$\rightarrow \nabla_{W_{i,k}}\text{err}(W, x, y) = \dfrac{\partial a_k}{\partial W_{ik}}(-(S(a_y))^{-1}\dfrac{\partial S(a_y)}{\partial a_k}) = \dfrac{\partial a_k = w_k^T x}{\partial W_{ik}}(S(a_k) - [\![y = k]\!]) = (S(a_k) - [\![y = k]\!])x_i = (h_k(x) - [\![y = k]\!])x_i$

Problem 11:

[e] is correct.

Given $\theta(y' = 1|x) = \dfrac{\exp(w_3^T x)}{1 + \exp(w_3^T x)}$,

$$\exp(w_3^T x) = \frac{\theta(y' = 1|x)}{1 - \theta(y' = 1|x)} = \frac{\theta(y' = 1|x)}{\theta(y' = -1|x)} = \frac{\Pr(y = 2|x)}{\Pr(y = 1|x)} = \frac{h_2(x)}{h_1(x)} = \frac{\exp(w_2^T x)}{\exp(w_1^T x)} = \exp[(w_2 - w_1)^T x].$$

Thus, $w_3 = w_2 - w_1$.

Problem 12:

[e] is correct.

```
dta = [[0, 1, -1],
       [1, -0.5, -1],
       [-1, 0, -1],
       [-1, 2, 1],
       [2, 0, 1],
       [1, -1.5, 1],
       [0, -2, 1]]

para = [[-9, -1, 0, 2, -2, 3],
        [-5, -1, 2, 3, -7, 2],
        [9, -1, 4, 2, -2, 3],
        [2, 1, -4, -2, 7, -4],
        [-7, 0, 0, 2, -2, 3]]

for i in para:
    for j in dta:
        y = i[0]*1+i[1]*j[0]+i[2]*j[1]+i[3]*(j[0]**2)+i[4]*(j[0]*j[1])+i[5]*(j[1]**2)
        if y*j[2] <= 0:
            print("F")
            break
    else:
        print("T")

'''
##OUT
F
F
F
F
T
'''
```

Problem 13:

[idk]

In [1]:
```
import numpy as np
import numpy.linalg as la
```

In [2]:
```
# data loading
train = np.loadtxt("hw3_train.dat", dtype=np.float, delimiter='\t')
train
```

Out[2]:
```
array([[ 2.965153,  2.447427,  1.958754, ..., -4.510862, -0.006392,
        -1.      ],
       [-4.303194, -0.032933,  2.568076, ..., -0.949456, -0.744622,
        -1.      ],
       [-0.261568,  0.974854, -1.132005, ...,  2.728657,  4.001672,
        -1.      ],
       ...,
       [-3.106073, -0.425107, -4.141368, ..., -1.741274,  1.479998,
        -1.      ],
       [ 3.665836,  3.401544, -0.779725, ...,  2.058829,  1.474205,
         1.      ],
       [ 1.74612 ,  2.365236, -2.501292, ...,  3.269879, -0.259019,
         1.      ]])
```

In [3]:
```
np.shape(train)
```

Out[3]:
```
(1000, 11)
```

In [4]:
```
test = np.loadtxt("hw3_test.dat", dtype=np.float, delimiter='\t')
test
```

Out[4]:
```
array([[ 1.406809,  1.629765,  0.137603, ..., -0.019709,  0.200252,
         1.      ],
       [-4.507169, -0.944514, -1.634057, ...,  3.074277,  1.056912,
         1.      ],
       [-1.559574, -4.985006,  0.946881, ...,  2.908434,  0.312697,
        -1.      ],
       ...,
       [-1.019819,  0.524448,  0.617346, ...,  1.389058,  1.085457,
         1.      ],
       [ 1.42676 , -1.123184,  1.323768, ..., -3.147399, -1.061905,
        -1.      ],
       [-1.098224,  3.060828,  0.935526, ...,  0.476239, -4.390546,
         1.      ]])
```

Q 14

The answer is 0.60. [d]

In [5]:
```
# data
x = np.concatenate((np.ones((1000,1)), train[:, 0:10]), axis=1)
x
```

Out[5]:
```
array([[ 1.      ,  2.965153,  2.447427, ...,  1.621895, -4.510862,
        -0.006392],
       [ 1.      , -4.303194, -0.032933, ..., -4.560341, -0.949456,
        -0.744622],
       [ 1.      , -0.261568,  0.974854, ..., -4.031803,  2.728657,
         4.001672],
       ...,
       [ 1.      , -3.106073, -0.425107, ..., -2.093555, -1.741274,
         1.479998],
       [ 1.      ,  3.665836,  3.401544, ...,  0.78021 ,  2.058829,
         1.474205],
       [ 1.      ,  1.74612 ,  2.365236, ..., -0.833364,  3.269879,
        -0.259019]])
```

```
In [6]: y = train[:, 10]
        y
```

```
Out[6]: array([-1., -1., -1.,  1., -1.,  1., -1.,  1., -1., -1.,  1.,  1., -1.,
        1., -1., -1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,
       -1.,  1., -1., -1., -1., -1.,  1., -1.,  1.,  1.,  1., -1., -1.,
       -1.,  1., -1.,  1.,  1.,  1.,  1., -1., -1., -1.,  1.,  1.,  1.,
        1.,  1., -1., -1.,  1., -1., -1., -1.,  1., -1.,  1., -1., -1.,
       -1.,  1.,  1., -1., -1.,  1.,  1.,  1., -1., -1.,  1.,  1., -1.,
        1., -1.,  1., -1.,  1., -1.,  1.,  1., -1.,  1., -1.,  1.,  1.,
        1., -1., -1., -1., -1.,  1., -1.,  1., -1.,  1.,  1.,  1., -1.,
       -1.,  1., -1.,  1., -1., -1.,  1.,  1., -1., -1.,  1.,  1., -1.,
        1.,  1., -1.,  1.,  1., -1., -1.,  1.,  1., -1.,  1.,  1.,  1.,
        1.,  1., -1., -1., -1.,  1., -1., -1., -1.,  1.,  1.,  1., -1.,
        1., -1.,  1., -1.,  1., -1.,  1., -1., -1., -1., -1., -1., -1.,
       -1., -1., -1.,  1., -1., -1., -1.,  1.,  1., -1., -1., -1.,  1.,
       -1., -1.,  1., -1.,  1., -1.,  1., -1.,  1., -1., -1., -1., -1.,
       -1.,  1., -1.,  1., -1.,  1.,  1., -1., -1., -1.,  1., -1., -1.,
       -1., -1., -1., -1., -1., -1., -1.,  1., -1.,  1., -1.,  1., -1.,
       -1., -1.,  1., -1., -1.,  1., -1.,  1.,  1.,  1.,  1.,  1.,  1.,
       -1., -1.,  1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,
        1.,  1.,  1.,  1., -1., -1., -1., -1., -1., -1.,  1.,  1.,  1.,
       -1., -1.,  1.,  1., -1.,  1., -1., -1.,  1., -1.,  1.,  1.,  1.,
       -1., -1.,  1., -1., -1., -1., -1.,  1.,  1.,  1.,  1., -1.,  1.,
        1.,  1., -1., -1., -1.,  1., -1., -1.,  1., -1., -1., -1.,  1.,
        1., -1., -1.,  1., -1.,  1.,  1., -1., -1.,  1.,  1.,  1.,  1.,
        1.,  1., -1.,  1.,  1., -1.,  1.,  1.,  1., -1., -1., -1., -1.,
        1.,  1.,  1.,  1.,  1.,  1., -1., -1.,  1., -1.,  1.,  1., -1.,
       -1., -1.,  1., -1.,  1.,  1.,  1., -1.,  1., -1.,  1., -1.,  1.,
        1.,  1., -1.,  1., -1.,  1.,  1.,  1.,  1.,  1., -1., -1.,  1.,
       -1.,  1.,  1.,  1., -1., -1.,  1.,  1.,  1., -1., -1.,  1.,  1.,
        1., -1., -1.,  1.,  1., -1.,  1.,  1., -1., -1.,  1., -1., -1.,
        1., -1., -1., -1., -1.,  1.,  1.,  1.,  1.,  1.,  1.,  1., -1.,
        1.,  1.,  1.,  1.,  1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,
       -1.,  1., -1.,  1.,  1., -1., -1., -1., -1.,  1.,  1., -1., -1.,
        1.,  1.,  1., -1., -1., -1., -1.,  1.,  1., -1.,  1.,  1.,  1.,
        1., -1., -1.,  1.,  1.,  1., -1.,  1., -1.,  1., -1., -1., -1.,
       -1.,  1., -1., -1., -1.,  1.,  1.,  1., -1., -1., -1., -1.,  1.,
        1., -1., -1.,  1., -1.,  1., -1.,  1., -1., -1., -1., -1., -1.,
        1.,  1.,  1.,  1.,  1., -1., -1.,  1.,  1., -1., -1.,  1.,  1.,
       -1.,  1., -1., -1.,  1.,  1.,  1.,  1., -1., -1., -1., -1.,  1.,
        1., -1.,  1., -1., -1.,  1., -1., -1.,  1., -1.,  1.,  1.,  1.,
       -1., -1.,  1., -1.,  1., -1.,  1., -1.,  1.,  1., -1., -1.,  1.,
       -1.,  1.,  1., -1.,  1.,  1., -1.,  1., -1.,  1.,  1., -1.,  1.,
       -1.,  1., -1.,  1., -1., -1., -1., -1., -1., -1.,  1., -1., -1.,
        1., -1., -1., -1., -1.,  1.,  1.,  1.,  1., -1.,  1.,  1.,  1.,
        1., -1.,  1., -1.,  1., -1.,  1., -1.,  1.,  1.,  1.,  1., -1.,
        1., -1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,  1.,
        1.,  1.,  1., -1., -1., -1., -1.,  1., -1.,  1.,  1., -1.,  1.,
        1., -1.,  1., -1., -1., -1., -1., -1., -1.,  1., -1., -1., -1.,
       -1., -1.,  1., -1.,  1.,  1.,  1., -1., -1., -1., -1.,  1.,  1.,
        1., -1.,  1., -1.,  1.,  1.,  1.,  1.,  1.,  1.,  1., -1.,  1.,
       -1., -1.,  1.,  1., -1., -1.,  1., -1.,  1.,  1.,  1., -1., -1.,
       -1., -1.,  1., -1.,  1.,  1.,  1.,  1., -1.,  1., -1.,  1.,  1.,
        1., -1., -1.,  1., -1., -1., -1.,  1., -1., -1., -1.,  1., -1.,
        1., -1., -1., -1.,  1.,  1.,  1., -1., -1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1., -1., -1., -1.,  1., -1.,  1.,  1.,  1.,  1., -1.,
        1., -1.,  1., -1.,  1.,  1.,  1., -1., -1.,  1., -1., -1., -1.,
        1., -1.,  1.,  1., -1., -1., -1., -1.,  1.,  1.,  1., -1., -1.,
        1.,  1.,  1., -1., -1.,  1.,  1., -1.,  1.,  1.,  1.,  1., -1.,
        1., -1., -1.,  1., -1., -1.,  1., -1., -1.,  1.,  1., -1.,  1.,
        1., -1.,  1.,  1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,  1.,
       -1.,  1., -1., -1., -1., -1., -1.,  1., -1., -1., -1., -1.,  1.,
        1.,  1.,  1.,  1.,  1., -1.,  1.,  1., -1.,  1., -1.,  1., -1.,
        1.,  1.,  1., -1., -1.,  1.,  1., -1., -1.,  1., -1.,  1.,  1.,
       -1.,  1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,  1., -1.,  1.,
        1.,  1., -1., -1.,  1., -1., -1.,  1.,  1., -1., -1., -1.,  1.,
       -1., -1.,  1.,  1., -1.,  1.,  1.,  1., -1., -1.,  1., -1., -1.,
       -1.,  1.,  1., -1.,  1., -1., -1., -1.,  1., -1., -1.,  1.,  1.,
       -1.,  1., -1., -1.,  1.,  1., -1.,  1.,  1., -1., -1., -1., -1.,
       -1.,  1., -1.,  1., -1., -1.,  1., -1.,  1.,  1.,  1., -1., -1.,
        1., -1., -1., -1., -1.,  1.,  1., -1.,  1.,  1.,  1.,  1.,  1.,
        1.,  1.,  1., -1.,  1.,  1., -1., -1.,  1.,  1., -1., -1., -1.,
        1.,  1.,  1., -1.,  1., -1.,  1., -1., -1., -1., -1., -1.,  1.,
        1., -1., -1., -1., -1.,  1., -1., -1.,  1.,  1., -1., -1., -1.,
       -1.,  1., -1.,  1.,  1.,  1.,  1.,  1., -1.,  1., -1.,  1.,  1.,
       -1.,  1., -1., -1.,  1., -1.,  1., -1., -1.,  1.,  1.,  1., -1.,
       -1.,  1.,  1., -1.,  1., -1.,  1.,  1., -1., -1., -1., -1.,  1.,
        1., -1., -1.,  1., -1.,  1., -1.,  1.,  1.,  1., -1., -1., -1.,
       -1., -1., -1.,  1.,  1., -1.,  1.,  1., -1., -1.,  1.,  1.])
```

In [7]:
```
w_lin = la.solve(x.T@x, x.T@y)
w_lin
```

Out[7]:
```
array([ 0.00754448,  0.14325775,  0.03548979,  0.02775754,  0.02547401,
       -0.08173504,  0.0807915 , -0.10929449,  0.07789522,  0.16407524,
        0.14069183])
```

In [8]:
```
ein_lin = 0.001*(la.norm(x@w_lin-y)**2)
ein_lin
```

Out[8]:
```
0.6053223804672918
```

In [9]:
```
# gradient
nablaein_lin = 0.002*(x.T@x@w_lin-x.T@y)
nablaein_lin
```

Out[9]:
```
array([ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  2.84217094e-17,
        2.84217094e-17,  0.00000000e+00,  0.00000000e+00, -2.27373675e-16,
       -1.13686838e-16,  0.00000000e+00, -2.27373675e-16])
```

Q15

The answer is 1800 [c].

In [10]:
```
def exp(w,x,y):
    count = 0
    while 0.001*(la.norm(x@w-y)**2) > 1.01*ein_lin:
        #randomly pick example
        tempidx = np.random.randint(0, 999)
        xt = x[tempidx]
        yt = y[tempidx]
        #SGD update
        w = w+0.001*2*(yt-w.T@xt)*xt
        #count iterations
        count+=1
    return count
```

In [12]:
```
x = np.concatenate((np.ones((1000,1)), train[:, 0:10]), axis=1)
y = train[:, 10]
w = np.zeros(11)
n_iter = []
for i in range(1000):
    n_iter.append(exp(w,x,y))
np.mean(n_iter)
```

Out[12]:
```
1855.774
```

Q16

The answer is 0.56 [c].

In [13]:
```
def sigmoid(x):
    s = 1 / (1 + np.exp(-x))
    return s
```

In [14]:
```
def exp16(w,x,y):
    for i in range(500):
        #randomly pick example
        tempidx = np.random.randint(0, 999)
        xt = x[tempidx]
        yt = y[tempidx]
        #SGD update
        w = w+0.001*sigmoid(-yt*(w.T@xt))*yt*xt
        #count iterations
    ein_ce = 0
    for i in range(1000):
        ein_ce += np.log(1+np.exp(-y[i]*(w.T@x[i])))
    ein_ce /= 1000
    return ein_ce
```

In [15]:
```
x = np.concatenate((np.ones((1000,1)), train[:, 0:10]), axis=1)
y = train[:, 10]
w = np.zeros(11)
all_ce = []
for i in range(1000):
    all_ce.append(exp16(w,x,y))
np.mean(all_ce)
```
Out[15]: 0.5689133164970044

Q17

The answer is 0.50 [b].

In [16]:
```
x = np.concatenate((np.ones((1000,1)), train[:, 0:10]), axis=1)
y = train[:, 10]
w = w_lin
all_ce = []
for i in range(1000):
    all_ce.append(exp16(w,x,y))
np.mean(all_ce)
```
Out[16]: 0.5028516300188179

Q18

The answer is 0.32 [a].

In [17]:
```
def sign(x):
    return 1 if x > 0 else -1
```
In [18]:
```
train_x = np.concatenate((np.ones((1000,1)), train[:, 0:10]), axis=1)
train_y = train[:, 10]
test_x = np.concatenate((np.ones((3000,1)), test[:, 0:10]), axis=1)
test_y = test[:, 10]
```
In [19]:
```
ein10 = np.mean([(1-sign(w_lin.T@train_x[i])*train_y[i])/2 for i in range(1000)])
eout10 = np.mean([(1-sign(w_lin.T@test_x[i])*test_y[i])/2 for i in range(3000)])
abs(ein10-eout10)
```
Out[19]: 0.3226666666666666

Q19

The answer is 0.36 [b]

In [20]:
```
train_x_19 = np.ones((1000,1))
test_x_19 = np.ones((3000,1))
for i in range(3):
    train_x_19 = np.concatenate((train_x_19, train[:, 0:10]**(i+1)), axis=1)
    test_x_19 = np.concatenate((test_x_19, test[:, 0:10]**(i+1)), axis=1)
```
In [21]:
```
np.shape(train_x_19)
```
Out[21]: (1000, 31)

In [22]:
```
np.shape(test_x_19)
```
Out[22]: (3000, 31)

In [23]:
```
w_lin19 = la.solve(train_x_19.T@train_x_19, train_x_19.T@train_y)
ein10 = np.mean([(1-sign(w_lin19.T@train_x_19[i])*train_y[i])/2 for i in range(1000)])
eout10 = np.mean([(1-sign(w_lin19.T@test_x_19[i])*test_y[i])/2 for i in range(3000)])
abs(ein10-eout10)
```
Out[23]: 0.37366666666666665

Q20

The answer is 0.44 [d]

In [24]:
```
train_x_20 = np.ones((1000,1))
test_x_20 = np.ones((3000,1))
for i in range(10):
    train_x_20 = np.concatenate((train_x_20, train[:, 0:10]**(i+1)), axis=1)
    test_x_20 = np.concatenate((test_x_20, test[:, 0:10]**(i+1)), axis=1)
w_lin20 = la.solve(train_x_20.T@train_x_20, train_x_20.T@train_y)
ein10 = np.mean([(1-sign(w_lin20.T@train_x_20[i])*train_y[i])/2 for i in range(1000)])
eout10 = np.mean([(1-sign(w_lin20.T@test_x_20[i])*test_y[i])/2 for i in range(3000)])
abs(ein10-eout10)
```

Out[24]: 0.44666666666666666

In [25]:
```
np.shape(test_x_20)
```

Out[25]: (3000, 101)

In [ ]: