# ML HW2

B06303126 Lo Yun Chien

November 2020

## Question 1

Correct answer:(b)
The calculation is

$$(0.1)^2(1-\frac{12}{N})<0.006$$

Which means

$$N<30$$

## Question 2

Correct answer:(a)
For any X is a m by n matrix, $X^TX=A$ is a n by n matrix. Whether A is a full rank, $Aw=X^Ty$ exists solution for w. If A is full rank, then the solution is unique. if not, there are many solution.

## Question 3

Correct answer:(c)
Consider H is a projection matrix, projecting y to the column space of X. Then, multiply each row by $\frac{1}{n}$ may create different basis of the space.

## Question 4

Correct answer:(e)
The first argument is right by Hoeffding, notice that $\theta$ is mean and v is sample mean is this Bernoulli trial.
The likelihood function is $f(x|\theta)=(\theta)^x(1-\theta)^{1-x}$ for x is 0 or 1. For N times, the total likelihood is $F(x|\theta)=(\theta)^{Nv}(1-\theta)^{N(1-v)}$, then both take ln and differentiate is w.r.t $\theta$, we get $F'(\theta)=Nv(1-\theta)-(N-Nv)\theta$. To maximize it, we need $F'(\hat{\theta})=0$. Then $v=\hat{\theta}$
Caculate $\nabla E_{in}$, we have $\nabla E_{in}=\frac{2}{N}\sum_{n=1}^{N}(\hat{y}-y_n)$. When $\hat{y}=0$, $\nabla E_{in}|_{\hat{y}=0}=\frac{2}{N}\sum_{n=1}^{N}(-y_n)=-2v$. To minimize it, we want $\nabla E_{in}=0$, hence $\hat{y}=v$.

# Question 5

Correct answer:(c)
Define $F_\theta(y) = \frac{y}{\theta}$ as the cdf of uniform(0,$\theta$), we found the likelihood is

$$\prod_{i=1}^{n} F_\theta(y_n) = \prod_{i=1}^{n} \frac{y_n}{\theta}$$

# Question 6

Correct answer:(b)

$$\nabla E_{in} = -\frac{\eta}{N} \sum_{n:y_n \neq sign(w_t^T x_n)} y_n x_n$$

By integration and suppose $E_{in} = 0$ if every points are correctly classified. we have

$$E_{in} = -\frac{\eta}{N} \sum_{n:y_n \neq sign(w_t^T x_n)} y_n w^T x_n$$

By guessing that err will give correct points 0 error, we have

$$E_{in} = \frac{\eta}{N} \sum_{i=1}^{N} max\{0, -y_n w^T x_n\}$$

So the pointwise error function may be $max\{0, -y_n w^T x_n\}$

# Question 7

Correct answer:(a)
By the err function given as $e^{-yw^T x}$, the corresponding $\nabla e^{-yw^T x}$ is

$$\nabla e^{-yw^T x} = e^{-y_n w^T x_n} - y_n x_n$$

# Question 8

Correct answer: (b)
Simply calc. approximated $\nabla E(w)$

$$\nabla E(w) = b_E(u) + A_E(u)(w - u)$$

We still want $\nabla E(w) = 0$, combining $w = v + u$, so it imply

$$b_E(u) + A_E(u)(w - u) = 0 \Longrightarrow v = (A_E(u))^{-1} - b_E(u)$$

## Question 9

Correct answer: (b)
In linear regression, we have

$$E_{in} = \frac{1}{n}\|Xw - y\|^2$$

Now for $\nabla E_{in}$

$$\nabla E_{in} = \frac{2}{n}(X^T X w - X^T y)$$

Now for $\nabla^2 E_{in}$

$$\nabla^2 E_{in} = \frac{2}{N} X^T X$$

## Question 10

Correct answer: (b)

$$err(W, x, y) = -\ln h_y(x) = -\sum_{k=1}^{K} [y = k](\ln h_k(x))$$

Calc. the partial derivative.

$$\frac{\partial err(W, x, y)}{\partial W_{ik}} = -[y = k](\frac{h_k(x)'}{h_k(x)})$$

Now calculate

$$\frac{\partial h_k(x)}{\partial W_{ik}} = \frac{e^{w_k^T x}}{\sum_{i=1}^{K} e^{w_i^T x}} x_i - \frac{e^{w_k^T x}}{(\sum_{i=1}^{K} e^{w_i^T x})^2} x_i e^{w_k^T x} x_i$$

The conclusion is

$$-[y = k](\frac{h_k(x)'}{h_k(x)}) = -[y = k](x_i - h_k(x)x_i) = x_i[y = k](h_k(x) - 1)$$

## Question 11

I don't know

## Question 12

Correct answer:(e)
Proof by programming

## Question 13

I don't know neither

In [78]:
```
import pandas as pd
import numpy as np
import random
import math
```

## Reading train data

In [79]:
```
train_data = pd.read_table("hw3_train.dat.txt", header = None)
train_data = train_data.rename(columns = {0: "x_{1}", 1: "x_{2}", 2: "x
_{3}", 3: "x_{4}",4: "x_{5}",
                                5: "x_{6}", 6: "x_{7}", 7: "x_{8}", 8: "x_
{9}", 9: "x_{10}", 10: "y"})
train_data["x_{0}"] = 1

cols = train_data.columns.tolist()
cols = cols[-1:] + cols[:-1]

train_data = train_data[cols]
```

## Define E_in function

In [80]:
```
def E_in(w, X, Y):
    return pow(np.linalg.norm(np.add(np.dot(X,w), -Y)), 2) / len(Y)
```

## Conduct linear regression

In [81]:
```
def linear_regression(X, Y):
    Transpose_X = X.T
    X_T_X = Transpose_X.dot(X)
    X_T_X_inv = pd.DataFrame(np.linalg.pinv(X_T_X.values), X_T_X.column
```

```
s, X_T_X.index)
    w_lin = (X_T_X_inv.dot(Transpose_X)).dot(Y)
    return w_lin
```

# Problem 14:

Correct answer: (d)0.60
linear regression with colsed form

In [82]:
```
Y = train_data["y"]
X = train_data.drop(["y"], axis = 1)
```

In [83]:
```
w_lin = linear_regression(X, Y)
sqr_error = E_in(w_lin, X, Y)
sqr_error
```

Out[83]:
```
0.6053223804672917
```

# Problem 15

Correct answer: (c) 1800
Linear regression with SGD

In [84]:
```
N_list = []

for times in range(1000):
    sgd_error = 999
    N = 0
    w = np.zeros(11)
    while (sgd_error >= 1.01 * sqr_error):
        N += 1
        n = random.randint(1, 1000)
        y_n = Y.loc[n-1]
        x_n = X.loc[n-1]
```

```
        w = np.add(w, (y_n - w.T.dot(x_n)) * 2 * 0.001 * x_n)
        sgd_error = E_in(w, X, Y)
    N_list.append(N)
```

In [85]:
```
N_list = pd.DataFrame(N_list)
N_list.mean()
```

Out[85]:
```
0    1949.903
dtype: float64
```

# Problem 16, 17

Correct answer: (c) 0.56

Correct answer: (b) 0.50

logistic regression with SGD

define E_in_log

In [106]:
```
def theta_f(x):
    return 1 / (1 + pow(math.e, -x))

def E_in_log(w, X, Y):
    total_error = 0
    for i in range(len(Y)):
        y_n = Y.loc[i]
        x_n = X.loc[i]
        total_error += math.log(1 + pow(math.e, - (y_n * w.T.dot(x_n
))))
    return total_error / len(Y)
```

In [107]:
```
error_list = []

for i in range(1000):
    sgd_error = 999
    w = np.zeros(11)
    for j in range(500):
        n = random.randint(1, 1000)
```

```
        y_n = Y.loc[n-1]
        x_n = X.loc[n-1]
        w = np.add(w, theta_f(-y_n * w.T.dot(x_n)) * 0.001 * y_n * x_n)
    sgd_error = E_in_log(w, X, Y)
    error_list.append(sgd_error)
```

In [108]:
```
error_df = pd.DataFrame(error_list)
error_df.mean()
```

Out[108]:
```
0    0.568543
dtype: float64
```

In [109]:
```
error_list = []

for i in range(1000):
    sgd_error = 999
    w = w_lin
    for j in range(500):
        n = random.randint(1, 1000)
        y_n = Y.loc[n-1]
        x_n = X.loc[n-1]
        w = np.add(w, theta_f(-y_n * w.T.dot(x_n)) * 0.001 * y_n * x_n)
    sgd_error = E_in_log(w, X, Y)
    error_list.append(sgd_error)

error_df = pd.DataFrame(error_list)
error_df.mean()
```

Out[109]:
```
0    0.502844
dtype: float64
```

# Problem 18:

Correct answer: (a) 0.32

Calculate binary classification error

```

```

In [86]:
```
test_data = pd.read_table("hw3_test.dat.txt", header = None)
test_data = test_data.rename(columns = {0: "x_{1}", 1: "x_{2}", 2: "x_
{3}", 3: "x_{4}",4: "x_{5}",
                                        5: "x_{6}", 6: "x_{7}", 7: "x_{8}", 8: "x_
{9}", 9: "x_{10}", 10: "y"})
test_data["x_{0}"] = 1

cols = test_data.columns.tolist()
cols = cols[-1:] + cols[:-1]

test_data = test_data[cols]
```

In [87]:
```
Y_test = test_data["y"]
X_test = test_data.drop(["y"], axis = 1)
```

In [88]:
```
def binary(x):
    if x > 0:
        return 1
    else:
        return -1

def Error_binary(w, X, Y):
    total_error = 0
    for i in range(len(Y)):
        if binary(w.T.dot(X.loc[i])) == Y.loc[i]:
            pass
        else:
            total_error += 1
    return total_error / len(Y)
```

In [89]:
```
print(abs(Error_binary(w_lin, X, Y) - Error_binary(w_lin, X_test, Y_tes
t)))
```

```
0.3226666666666666
```

## Problem 19, 20:

Correct answer: (b) 0.36
Correct answer: (d) 0.44

Nonlinear transform

In [90]:
```
def phi(x, Q):
    phi_array = np.array([1])
    for i in range(Q):
        q_list = [pow(xi, i+1) for xi in x]
        q_array = np.array(q_list)
        phi_array = np.append(phi_array, q_array)
    return phi_array
```

In [91]:
```
X_3 = X.drop(["x_{0}"], axis=1)
X_3 = X_3.apply(phi, axis=1, args=(3,))
X_3 = pd.DataFrame(X_3.tolist())

X_test_3 = X_test.drop(["x_{0}"], axis=1)
X_test_3 = X_test_3.apply(phi, axis=1, args=(3,))
X_test_3 = pd.DataFrame(X_test_3.tolist())
```

In [92]:
```
w_lin_3 = linear_regression(X_3, Y)
```

In [93]:
```
abs(Error_binary(w_lin_3, X_3, Y) - Error_binary(w_lin_3, X_test_3, Y_t
est))
```

Out[93]:
```
0.37366666666666665
```

In [94]:
```
X_10 = X.drop(["x_{0}"], axis=1)
X_10 = X_10.apply(phi, axis=1, args=(10,))
X_10 = pd.DataFrame(X_10.tolist())

X_test_10 = X_test.drop(["x_{0}"], axis=1)
X_test_10 = X_test_10.apply(phi, axis=1, args=(10,))
X_test_10 = pd.DataFrame(X_test_10.tolist())
```

In [95]:
```
w_lin_10 = linear_regression(X_10, Y)
abs(Error_binary(w_lin_10, X_10, Y) - Error_binary(w_lin_10, X_test_10,
 Y_test))
```

Out[95]:
```
0.446
```