1. b

1.

$$(0.1)^2 \left(1 - \frac{11+1}{N}\right) < 0.006$$

$$1 - \frac{12}{N} \le 0.6$$

$$N \le \frac{12}{0.4} = 30$$

2. a

   Since if $X^T X$ *is invertible*, w has an unique solution, and if $X^T X$ is singular, then w has multiple solutions. Therefore, There is at least one solution for the normal equation.

3. c

3.

(a) $2 \cdot X \cdot \frac{1}{4}(X^T X)^{-1} \cdot 2 \cdot X^T = X(X^T X)^{-1} X^T \rightarrow$ no change.

(b) (d) Column operation → multiplied matrix with RIGHT HAND side matrix

$XC\left((XC)^T XC\right)^{-1}(XC)^T = X \cdot C\,(C^T X^T X\, C)^{-1} C^T X^T$

$= X \cdot C \cdot C^{-1} (X^T X)^{-1} (C^T)^{-1} C^T X^T$, C is invertible, by def

$= X(X^T X)^{-1} X^T \rightarrow$ no change

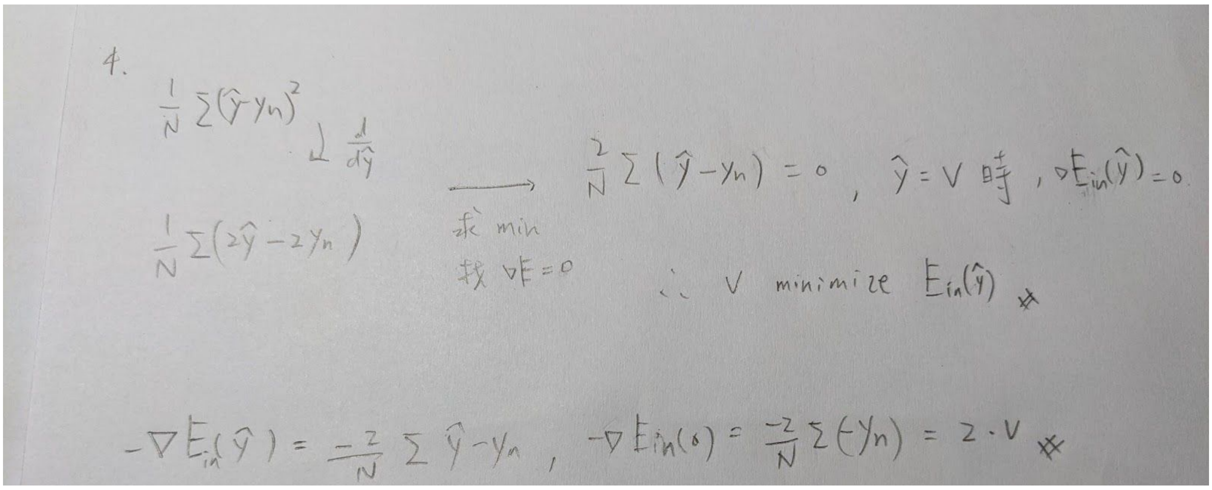
(c) row operation → multiplied matrix with LEFT HAND side matrix

$RX\left((RX)^T RX\right)^{-1}(RX)^T = RX\,(X^T R^T R\, X)^{-1} X^T R^T \rightarrow$ might change

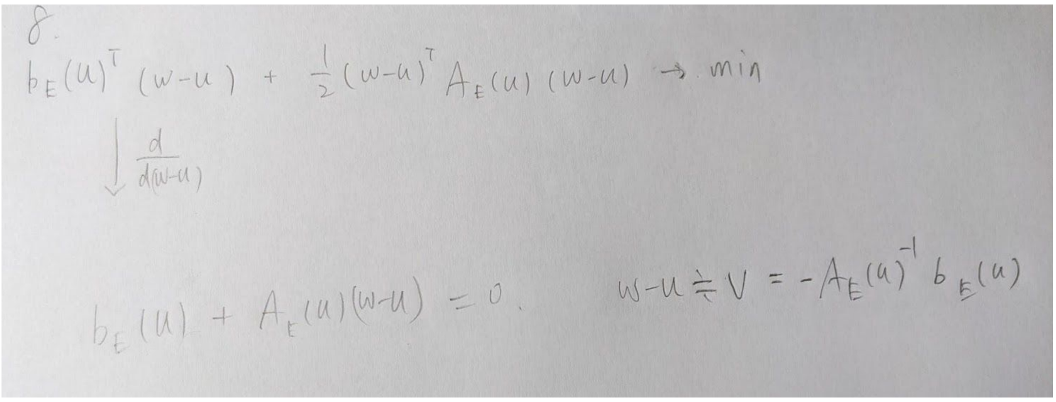
4. e

   For the first option, it is the definition of the slide. For the second option, since \theta_head is the likelihood of the result, and \v is the max observed value, thus \v maximizes the likelihood function.

For the other options, follow the below figure.



5. a
   Given N values, calculate the possible \theta for generating this uniform distribution. Since it's uniform distribution, the probability of each value is 1 / \theta_head, then the likelihood of n points is 1\theta_head multiplied over n times.
6. b
   If the term of $y_n \neq sign(x_t^T x_n)$ needed to be preserved, the term $-y_n\ w^T x$, which will be positive, can only be preserved by min max operation in option (b).
7. a
   Just take the derivative of the exponential error function and multiplied by -1.
8. b



9. b

9.

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

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

10. b

10.

$$\frac{\partial err(W, x, y)}{\partial W_{ik}} = \frac{\partial}{\partial W_{ik}}\left(-\sum_{k=1}^{K} \ln h_k(x)\right)$$

$$= \frac{\partial}{\partial W_{ik}}\left[-\sum_{k=1}^{K}\left(\ln \exp(w_k^T x) - \ln\left(\sum \exp(w^T x)\right)\right)\right]$$

$$= -\left(x_i [y=k] - \frac{\partial}{\partial W_{ik}} \ln\left(\sum \exp(w^T x)\right)\right)$$

$$= -\left(x_i [y=k] - \frac{\exp(w_k^T x)}{\sum \exp(w^T x)} \cdot x_i\right)$$

$$= x_i\left(-[y=k] + h_k(x)\right)$$

11. e

11.

$$\min -\sum \ln\left(\frac{\exp(w_{y_n}^T x_n)}{\exp(w_1^T x_n)+\exp(w_2^T x_n)}\right)$$

$y_n=-1$, $$-\ln\left(\frac{\exp(w_1^T x_n)}{\exp(w_1^T x_n)+\exp(w_2^T x_n)}\right) = \ln\left(1+\exp((w_2-w_1)^T x_n)\right)$$

$$= \ln\left(1+\exp(-(-1)(w_2-w_1)^T x_n)\right)$$

$y_n=1$, $$-\ln\left(\frac{\exp(w_2^T x_n)}{\exp(w_1^T x_n)+\exp(w_2^T x_n)}\right) = \ln\left(1+\exp((w_1-w_2)^T x_n)\right)$$

$$= \ln\left(1+\exp(-1(w_2-w_1)^T x_n)\right)$$ #

12. e

B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
0	1		1	0	1	0	0	1		-6	-2.5	16	-6	-4		-1
1	-0.5		1	1	-0.5	1	-0.5	0.25		-6.25	-0.5	9.75	-1.5	-3.25		-1
-1	0		1	-1	0	1	0	0		-6	7	12	-1	-5		-1
-1	2		1	-1	2	1	-2	4		10	17	36	-39	11		1
2	0		1	2	0	4	0	0		-3	9.5	15	-4	1		1
1	-1.5		1	1	-1.5	1	-1.5	2.25		1.75	12.5	13.75	-12.5	4.75		1
0	-2		1	0	-2	0	0	4		3	-9	13	-6	5		1

13. b

13

To get relationship between VC bound and input-size, we should suppose there are N datas.

Consider + − sign, there are 2dN hypothesis sets.

$2^N \le 2dN$

$N \le \log_2 d + \log_2 N + 1$

$\because \frac{N}{2} > \log_2 N, \; N \ge 4$

$\therefore N < \log_2 d + \frac{N}{2} + 1$

$\frac{N}{2} < \log_2 d + 1$

$N < 2[(\log_2 d) + 1]$ #

14. d
15. c
16. c
17. b
18. a
19. b
20. d

```
yiwenlai@YiWens-MacBook-Pro Desktop % python3 hw3.py
Q14  0.6053223804672918
Q15  1811.013
Q16  0.5690528639339274
Q17  0.5027272304016192
Q18  0.3226666666666666
Q19  0.37366666666666665
Q20  0.44666666666666666
```

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

```
def theta_func(s):
   return 1/(math.exp(-s)+1)
def square_err(w):
   E_in = LA.norm(X.dot(w)-y)
   return pow(E_in,2) / X.shape[0]
def cross_entropy_err(w):
   en = []
   for i in range(1000):
       en.append(-1 * np.log(theta_func(y[i] *
np.dot(w,X[i]))))
   return sum(en)/len(en)
def sign(s):
   if s > 0:
       return 1
   elif s < 0:
       return -1

train = []
with open('hw3_train.txt') as f:
   lines = f.readlines()
   for line in lines:
       nums = line.strip().split()
       data = [float(num) for num in nums]
       data.insert(0,1)
       train.append(data)
train = np.array(train)
X = train[:,:11]
y = train[:,-1]

test = []
with open('hw3_test.txt') as f1:
   lines = f1.readlines()
   for line in lines:
```

```
        nums = line.strip().split()
        data = [float(num) for num in nums]
        data.insert(0,1)
        # print(data)
        test.append(data)
test = np.array(test)
X_test = test[:,:11]
y_test = test[:,-1]
# print(X)

# Q14
X_T = np.transpose(X)
tmp = X_T.dot(X)
tmp = inv(tmp)
tmp = tmp.dot(X_T)
w_lin = tmp.dot(y)
E_lin = square_err(w_lin)
print("Q14 ",E_lin)
#Q15
iterations = []
for i in range(1000):
   steps = 0
   w = np.zeros(11)
   learning_rate = 0.001
   err = square_err(w)
   # print(err)
   while(True):
       if err <= 1.01*E_lin:
           break
       else:
           steps += 1
           indexs = np.random.randint(0,1000,1)
           index = indexs[0]
```

```
            w += 2 * learning_rate *
(y[index]-np.dot(w,X[index])) * X[index]
            err = square_err(w)
   # print(i)
   iterations.append(steps)
print("Q15 ",sum(iterations)/len(iterations))
# Q16
ERR_list = []
for i in range(1000):
   w = np.zeros(11)
   learning_rate = 0.001
   # print(err)
   for j in range(500):
       index = np.random.randint(0,1000,1)[0]
       w += learning_rate * theta_func(-1 * y[index] *
np.dot(w,X[index]) ) * y[index] * X[index]
   # print(i)
   ERR_list.append(cross_entropy_err(w))
print("Q16 ",sum(ERR_list)/len(ERR_list))

# Q17
ERR_list = []
for i in range(1000):
   w = w_lin.copy() # Q17
   # w = np.zeros(11)
   learning_rate = 0.001
   # print(err)
   for j in range(500):
       index = np.random.randint(0,1000,1)[0]
       w += learning_rate * theta_func(-1 * y[index] *
np.dot(w,X[index]) ) * y[index] * X[index]
   # print(i)
   ERR_list.append(cross_entropy_err(w))
print("Q17 ",sum(ERR_list)/len(ERR_list))
```

```
#Q 18
E_in = 0.0
E_out = 0.0
for i in range(len(y)):
   if sign( np.dot(w_lin.copy(), X[i]) ) != y[i]  :
       E_in += 1
for j in range(len(y_test)):
   if sign( np.dot(w_lin.copy(), X_test[j]) ) != y_test[j] :
       E_out += 1
# print(E_in,E_out)
print("Q18 ", abs( (E_in/len(y)) - (E_out/len(y_test)) ) )

#19 20


for Q in [3,10]:
   X_train = []
   y_train = []
   X_test = []
   y_test = []

   with open('hw3_train.txt') as f:
       lines = f.readlines()
       for line in lines:
           nums = line.strip().split()
           data = []
           for i in range(len(nums)-1):
               for p in range(Q):
                   data.append(pow(float(nums[i]),p+1))
           data.insert(0,1)
           label = float(nums[len(nums)-1])
           X_train.append(data)
           y_train.append(label)
```

```python
X = np.array(X_train)
y = np.array(y_train)

with open('hw3_test.txt') as f:
    lines = f.readlines()
    for line in lines:
        nums = line.strip().split()
        data = []
        for i in range(len(nums)-1):
            for p in range(Q):
                data.append(pow(float(nums[i]),p+1))
        data.insert(0,1)
        label = float(nums[len(nums)-1])
        X_test.append(data)
        y_test.append(label)
X_test = np.array(X_test)
y_test = np.array(y_test)

X_T = np.transpose(X)
tmp = X_T.dot(X)
tmp = inv(tmp)
tmp = tmp.dot(X_T)
w_lin = tmp.dot(y)
E_lin = square_err(w_lin)

E_in = 0.0
E_out = 0.0
for i in range(len(y)):
    if sign( np.dot(w_lin.copy(), X[i]) ) != y[i]  :
        E_in += 1
for j in range(len(y_test)):
    if sign( np.dot(w_lin.copy(), X_test[j]) ) != y_test[j]
:
        E_out += 1
```

```
    # print(E_in,E_out)
    if Q == 3:
        print("Q19 ", abs( (E_in/len(y)) - (E_out/len(y_test))
) )
    else:
        print("Q20 ", abs( (E_in/len(y)) - (E_out/len(y_test))
) )
```