1. (20 points) Consider a binary classification problem, where $\mathcal{Y} = \{-1, +1\}$. Assume a noisy scenario where the data is generated i.i.d. from some $P(\mathbf{x}, y)$. In class, we discussed that when the 0/1 error function (i.e. classification error) is considered, calculating the "ideal mini-target" on each $\mathbf{x}$ reveals the hidden target function of

$$f_{0/1}(\mathbf{x}) = \operatorname{argmax}_{y \in \{-1,+1\}} P(y|\mathbf{x}) = \operatorname{sign}\left(P(+1|\mathbf{x}) - \frac{1}{2}\right).$$

Instead of the 0/1 error, if we consider the CIA error function, where a false positive (classifying a negative example as a positive one) is 1000 times more important than a false negative, the hidden target should be changed to

$$f_{\text{cia}}(\mathbf{x}) = \operatorname{sign}\left(P(+1|\mathbf{x}) - \alpha\right).$$

Prove what the value of α should be.

$f_{0/1}(x) = \operatorname{sign}\left(P(+1|x) - \frac{1}{2}\right)$. $f(x) = 1$ if $P(+1|x) > 0.5$. $f(x) = -1$ if $P(+1|x) < 0.5$ threshold.

y \ f(x):

	+1	-1
+1	0	1
-1	1	0

$f_{0/1}(x_n) \neq y_n$, $y_n = -1$ is 1000 times important than $y_n = +1$

↓

In Error measure, when we enter the point $(y_n, f(x_n)) = (-1, +1)$ once, equals to we enter the point 1000 times.

y \ f(x):

	+1	-1
+1	0	1
-1	1000	0

→ $E_{out} = P(+1|x) \times \mathbb{I}[f(x) = -1] + P(-1|x) \times \mathbb{I}[f(x) = +1] \times 1000$

↳ $P(+1|x) : P(-1|x) = 1000 : 1$

$P(+1|x) = \frac{1000}{1001}$, $\alpha = \frac{1000}{1001}$ #

2. (20 points) Consider a binary classification task, where God gives you some noiseless data i.i.d. from an unknown distribution $P(\mathbf{x})$ and an unknown target function $f(\mathbf{x})$ that maps from $\mathcal{X}$ to $\{-1, +1\}$. After you use the data to obtain some $g(\mathbf{x})$ that suffers

$$\begin{aligned} E_{out}(g) &= \mathcal{E}_{\mathbf{x}\sim P(\mathbf{x})}[\![g(\mathbf{x}) \neq f(\mathbf{x})]\!] \text{ (here } \mathcal{E} \text{ means expectation, as shown in class slides)} \\ &= \mathbb{E}_{\mathbf{x}\sim P(\mathbf{x})}[\![g(\mathbf{x}) \neq f(\mathbf{x})]\!] \text{ (or if you like the more beautiful font E for expectation).} \end{aligned}$$

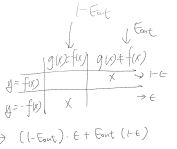
Now, assume that $g(\mathbf{x})$ is put in a noisy test environment where

$$\begin{aligned} P(y = +f(\mathbf{x})|\mathbf{x}) &= 1 - \epsilon \\ P(y = -f(\mathbf{x})|\mathbf{x}) &= \epsilon. \end{aligned}$$

Derive

$$\mathbb{E}_{(\mathbf{x},y)\sim P(\mathbf{x},y)}[\![g(\mathbf{x}) \neq y]\!]$$

as a function of $E_{out}(g)$ and ϵ.



In ideal case $P[g(x) \neq f(x)] = E_{out}$

$P[g(x) = f(x)] = 1 - E_{out}$

In noise $P[y \neq f(x) \mid x] = 1 - \epsilon$

$P[y = -f(x) \mid x] = \epsilon$

	$g(x)=f(x)$	$g(x)\neq f(x)$	
$y=f(x)$	S_1	S_2	$1-\epsilon$
$y=-f(x)$	S_3	S_4	ϵ
P_x	$1-E_{out}$	E_{out}	

$$\begin{aligned} \mathbb{E}_{(x,y)\sim P(x,y)}[g(x) \neq y] &= \mathbb{E}_{x\sim P(x)}[g(x) \neq f(x)] \cdot P(y = +f(x) \mid x) + \\ &\quad \mathbb{E}_{x\sim P(x)}[g(x) = f(x)] \cdot P(y = -f(x) \mid x) \\ &= (1 - E_{out}) \cdot \epsilon + E_{out} \cdot (1 - \epsilon) \end{aligned}$$

3. (20 points) Consider a hypothesis set that contains hypotheses of the form $h(x) = wx$ for $x \in \mathbb{R}$. Combine the hypothesis set with the squared error function to minimize

$$E_{\text{in}}(w) = \frac{1}{N}\sum_{n=1}^{N}(h(x_n) - y_n)^2$$

on a given data set $\{(x_n, y_n)\}_{n=1}^{N}$. Derive the optimal w_{LIN} in terms of (x_n, y_n) and express the result *without* using matrix/vector notations. You can assume all denominators to be non-zero. *(Hint: This is linear regression in $\mathbb{R}$ without the added x_0.)*

$h(x) = wx$

$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}(wx_n - y_n)^2$

w_{LIN} exists when $\min E_{in}(w)$ appears $\rightarrow \nabla E_{in}(w) = 0$

$E_{in}(w) = \frac{1}{N}\left[(wx_1 - y_1)^2 + (wx_2 - y_2)^2 + (wx_3 - y_3)^2 \cdots + (wx_n - y_n)^2\right]$

$= \frac{1}{N}\left(w^2x_1^2 - 2wx_1y_1 + y_1^2 \cdots + w^2x_n^2 - 2wx_ny_n + y_n^2\right)$

$\nabla E_{in}(w) = \frac{1}{N}\left(2wx_1^2 - 2x_1y_1 + 2wx_2^2 + 2x_2y_2 \cdots + 2wx_n^2 + 2x_ny_n\right)$

$= \frac{2}{N}\left(w\sum_{i=1}^{n=N}x_i^2 - \sum_{i=1}^{n=N}x_iy_i\right) = 0$

$\rightarrow w = \dfrac{\sum_{i=1}^{N}x_iy_i}{\sum_{i=1}^{N}x_i^2}$ #

4. (20 points) Consider the target function $f(x) = ax^2 + b$. Sample x uniformly from $[0, 1]$, and use all linear hypotheses $h(x) = w_0 + w_1 \cdot x$ to approximate the target function with respect to the squared error. For any given (a, b), derive the weights (w_0^*, w_1^*) of the optimal hypothesis as a function of (a, b).

$$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}\left(f(x_n) - h(x_n)\right) = \frac{1}{N}\sum_{n=1}^{N}\left(ax_n^2 + b - w_1x_n - w_0\right)^2$$

since x uniformly from $[0,1]$, $E_{in}(w_0, w_1) = \int_0^1 \left(ax^2 - w_1x + b - w_0\right)^2 dx$

$$\min E_{in}(w_0, w_1) \Leftrightarrow \nabla E_{in}(w_0, w_1) = 0$$

$$\begin{cases} \frac{\partial}{\partial w_0} = \int_0^1 \left(ax^2 - w_1x + b - w_0\right)\cdot 2\cdot(-1)\,dx = 0 \\ \frac{\partial}{\partial w_1} = \int_0^1 \left(ax^2 - w_1x + b - w_0\right)\cdot 2\cdot(-x)\,dx = 0 \end{cases}$$

$$\begin{cases} \int_0^1 \left(ax^2 - w_1x + b - w_0\right)dx = 0 \\ \int_0^1 \left(ax^3 - w_1x^2 + bx - w_0x\right)dx = 0 \end{cases}$$

$$\begin{cases} \frac{1}{3}ax^3 - \frac{1}{2}w_1x^2 + (b - w_0)x \Big|_0^1 = 0 \\ \frac{1}{4}ax^4 - \frac{1}{3}w_1x^3 + \frac{1}{2}(b - w_0)x^2 \Big|_0^1 = 0 \end{cases}$$

$$\begin{cases} \frac{1}{3}a - \frac{1}{2}w_1 + b - w_0 = 0 \\ \frac{1}{4}a - \frac{1}{3}w_1 + \frac{1}{2}b - \frac{1}{2}w_0 = 0 \end{cases}$$

$$\begin{cases} w_0 + \frac{1}{2}w_1 = \frac{1}{3}a + b \\ \frac{1}{2}w_0 + \frac{1}{3}w_1 = \frac{1}{4}a + \frac{1}{2}b \end{cases}, \quad w_0 + \frac{2}{3}w_1 = \frac{1}{4}a + b$$

$$\rightarrow \frac{1}{6}w_1 = \frac{1}{6}a, \quad w_1^* = a \#$$

$$w_0^* = -\frac{1}{6}a + b \#$$

5. (20 points) Consider running linear regression on $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where $\mathbf{x}_n$ includes the constant dimension $x_0 = 1$ as usual. For simplicity, you can assume that $\mathrm{X}^T\mathrm{X}$ is invertible. Assume that the unique (why :-)) solution $\mathbf{w}_{\mathrm{LIN}}$ is obtained after running linear regression on the data above. Now, consider an output transformation

$$y'_n = ay_n + b.$$

for some given constants (a, b). Run linear regression on $\{(\mathbf{x}_n, y'_n)\}_{n=1}^N$ to obtain the unique solution $\mathbf{w}'_{\mathrm{LIN}}$. Derive $\mathbf{w}'_{\mathrm{LIN}}$ as a function of $\mathbf{w}_{\mathrm{LIN}}$ and (a, b).

$W_{LIN} = (X^TX)^{-1}X^Ty$ ——— ①

$E_{in}(w) = \frac{1}{N}\|Xw' - (ay+b)\|^2 = \frac{1}{N}\left(w'^TX^TXw' - 2w^TX^T(ay+b) + (ay+b)^T(ay+b)\right)$

$\nabla E_{in}(w') = \frac{1}{N}(2X^TXw' - 2X^T(ay+b) + 0)$

W_{LIN}' happens when $\nabla E_{in} = 0$: $2X^TXw' = 2X^T(ay+b)$, $W_{LIN}' = (X^TX)^{-1}X^T(ay+b) = (X^TX)^{-1}X^T \cdot ay + (X^TX)^{-1}X^T \cdot b$

$\left[\begin{array}{l} a(X^TX)^{-1}X^Ty = a \cdot W_{LIN} \\ (X^TX)^{-1}X^T \cdot b \end{array}\right.$

since $X \cdot aW_{LIN} = ay$ $\begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} \cdots & x_{1n} \\ & & & & x_{2n} \\ & & & & x_{3n} \\ & & & & \vdots \\ 1 & & & & x_{nn} \end{bmatrix} \times a\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = a\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

$X \cdot W_{LIN}' = ay+b$ $\begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} \cdots & x_{1n} \\ \vdots & & & & \vdots \\ 1 & x_{n1} & \cdots & & x_{nn} \end{bmatrix} \times \begin{bmatrix} aw_1+b \\ aw_2 \\ aw_3 \\ \vdots \\ aw_n \end{bmatrix} = \begin{bmatrix} ay_1+b \\ ay_2+b \\ \vdots \\ ay_n+b \end{bmatrix}$ → all b's comes from $(W_{LIN}')_1$

$\downarrow$

$a\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} + b\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

→ $W_{LIN}' = aW_{LIN} + b\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$, where $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ is a $n \times 1$ matrix. #

6. (20 points) Let $E(\mathbf{w})\colon \mathbb{R}^d \to \mathbb{R}$ be a function. Denote the gradient $\mathbf{b}_E(\mathbf{w})$ and the Hessian $\mathrm{A}_E(\mathbf{w})$ by

$$\mathbf{b}_E(\mathbf{w}) = \nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\partial E}{\partial w_0}(\mathbf{w}) \\ \frac{\partial E}{\partial w_1}(\mathbf{w}) \\ \vdots \\ \frac{\partial E}{\partial w_d}(\mathbf{w}) \end{bmatrix}_{d\times 1} \quad \text{and} \quad \mathrm{A}_E(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_0^2}(\mathbf{w}) & \frac{\partial^2 E}{\partial w_0 \partial w_1}(\mathbf{w}) & \cdots & \frac{\partial^2 E}{\partial w_0 \partial w_d}(\mathbf{w}) \\ \frac{\partial^2 E}{\partial w_1 \partial w_0}(\mathbf{w}) & \frac{\partial^2 E}{\partial w_1^2}(\mathbf{w}) & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_d}(\mathbf{w}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_d \partial w_0}(\mathbf{w}) & \frac{\partial^2 E}{\partial w_d \partial w_1}(\mathbf{w}) & \cdots & \frac{\partial^2 E}{\partial w_d^2}(\mathbf{w}) \end{bmatrix}_{d\times d}.$$

Then, the second-order Taylor's expansion of $E(\mathbf{w})$ around $\mathbf{u}$ is:

$$E(\mathbf{w}) \approx E(\mathbf{u}) + \mathbf{b}_E(\mathbf{u})^T(\mathbf{w}-\mathbf{u}) + \frac{1}{2}(\mathbf{w}-\mathbf{u})^T \mathrm{A}_E(\mathbf{u})(\mathbf{w}-\mathbf{u}).$$

Suppose $\mathrm{A}_E(\mathbf{u})$ is positive definite. The optimal direction $\mathbf{v}$ such that $\mathbf{w} \leftarrow \mathbf{u}+\mathbf{v}$ minimizes the right-hand-side of the Taylor's expansion above is simply $-(\mathrm{A}_E(\mathbf{u}))^{-1}\mathbf{b}_E(\mathbf{u})$.

Hint: Homework 0! :-)

Now, consider minimizing $E_{\text{in}}(\mathbf{w})$ in logistic regression problem with Newton's method on a data set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with the cross-entropy error function for E_{in}:

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln(1+\exp(-y_n \mathbf{w}^T \mathbf{x}_n)).$$

For any given $\mathbf{w}_t$, let

$$h_t(\mathbf{x}) = \frac{1}{1+\exp(\mathbf{w}_t^T \mathbf{x})}.$$

Express the Hessian $\mathrm{A}_E(\mathbf{w}_t)$ with $E = E_{\text{in}}$ as $X^T D X$, where D is an N by N diagonal matrix. Derive what D should be in terms of h_t, $\mathbf{w}_t$, $\mathbf{x}_n$, and y_n.

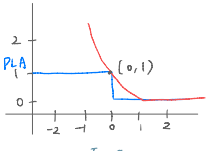
7. (20 points) The truncated squared loss

$$\text{err}(s, y) = (\max(0, 1 - ys))^2$$

can be easily shown to be an upper bound on the 0/1 error. Assume that s is generated from a linear scoring function $s = \mathbf{w}^T\mathbf{x}$ like Page 3/25 of Lecture 11. Derive a "perceptron learning algorithm" by applying SGD on the truncated squared loss. Compare the resulting algorithm with the original PLA. Discuss the similarities and differences using 5 to 10 sentences.

PLA: $\text{error}_{0/1}(s, y) = [\![\ \text{sign}(s) \neq y\ ]\!]$

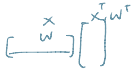
$w_{t+1} = w_t + 1 \cdot [\![\ y_n \neq \text{sign}(s)\ ]\!]\ (y_n \cdot x_n)$



SGD: $\text{error}(s, y) = (\max(0, 1-ys))^2$

$w_{t+1} = w_t + \eta \cdot (-\nabla \text{error}(y, w^T x))$

$= w_t + \eta \cdot \left(-\nabla \left(\max(0, 1-ys)\right)^2\right)$



- If $ys > 0$, $\text{error}(s, y) = 0$

  $0 < ys < 1$, $\text{error}(s, y) > \text{error}_{0/1}(s, y)$

  $ys = -1$, $\text{error}(s, y) = \text{error}_{0/1}(s, y)$

$$\left(\nabla \max(0, 1-ys)\right)^2 = \nabla\left(1 - w^T x y_n\right)^2 = \nabla\left(1 - w x^T y_n\right)^2$$

$$= \nabla\left(x \cdot w^T w \cdot x^T - 2 x w^T y_n + 1\right)^2$$

$$= 2\left(x^T x \cdot w - y_n x\right) = 2\left(s - y_n\right) x = (1 - ys) x y_n$$

$$w_{t+1} = w_t + 2\eta\ (1 - ys)\ x_n y_n \quad \#$$

PLA always updates with $\eta = 1$ when $[\![\ y_n \neq \text{sign}(s)\ ]\!]$; while SGD updates with altered η when $\max(0, 1-ys) > 0$. When $ys > 1$, both PLA and SGD won't update (error = 0), when $0 < ys < 1$, SGD updates (PLA doesn't), when $ys < 0$, $w_{t+1}^{PLA} = w^{PLA} + x_n y_n$, $w_{t+1}^{SGD} = w_t^{SGD} + 2\eta\,(1-ys)\,x_n y_n$ (updates depend on $x_n y_n$, alternable).

If $ys \ll 0$, updates larger; $ys < 0$, updates less.

However, eventually how much to update depend on η, which is usually smaller than 0.1

Consequently, SGD usually upates with flexible and smaller changes, PLA offers constant and unstable steps. In error measure, Error(SGD) > Error(PLA), that is, SGD can also be viewed as upper bound of PLA.

Multinomial Logistic Regression

8. In Lecture 11, we solve multiclass classification by OVA or OVO decompositions. One alternative to deal with multiclass classification is to extend the original logistic regression model to Multinomial Logistic Regression (MLR). For a K-class classification problem, we will denote the output space $\mathcal{Y} = \{1, 2, \cdots, K\}$. The hypotheses considered by MLR can be indexed by a matrix

$$\mathrm{W} = \begin{bmatrix} | & | & \cdots & | & \cdots & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_k & \cdots & \mathbf{w}_K \\ | & | & \cdots & | & \cdots & | \end{bmatrix}_{(d+1)\times K},$$

that contains weight vectors $(\mathbf{w}_1, \cdots, \mathbf{w}_K)$, each of length $d+1$. The matrix represents a hypothesis

$$h_y(\mathbf{x}) = \frac{\exp(\mathbf{w}_y^T \mathbf{x})}{\sum_{i=1}^{K} \exp(\mathbf{w}_i^T \mathbf{x})}$$

that can be used to approximate the target distribution $P(y|\mathbf{x})$ for any $(\mathbf{x}, y)$. MLR then seeks for the maximum likelihood solution over all such hypotheses. For a given data set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ generated i.i.d. from some $P(\mathbf{x})$ and target distribution $P(y|\mathbf{x})$, the likelihood of $h_y(\mathbf{x})$ is proportional to $\prod_{n=1}^{N} h_{y_n}(\mathbf{x}_n)$. That is, minimizing the negative log likelihood is equivalent to minimizing an $E_{\text{in}}(\mathrm{W})$ that is composed of the following error function

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

Consider minimizing $E_{\text{in}}(\mathrm{W}) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{err}(\mathrm{W}, \mathbf{x}_n, y_n)$ with gradient descent. Derive $\nabla E_{\text{in}}(\mathrm{W})$. Your result should simply be a matrix with the same size as W. *(Note: the hypothesis that transforms the scores $\{\mathbf{w}_i^T \mathbf{x}\}_{i=1}^{K}$ to $h_y(\mathbf{x})$ is often called a softmax function in (multiclass) deep learning.)*

$$\frac{\partial}{\partial \mathbf{w}_i} \sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)$$

$$= \frac{\partial}{\partial \mathbf{w}_i} \left(\exp\left(\mathbf{w}_1^T \mathbf{x}_n\right) + \exp\left(\mathbf{w}_2^T \mathbf{x}_n\right) \cdots \right)$$

$$= \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) \begin{bmatrix} x_{n,1} \\ x_{n,2} \\ \vdots \\ x_{n,n} \end{bmatrix} = \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) \mathbf{x}_n$$

$$E_{\text{in}}(\mathrm{W}) = \frac{1}{N} \sum_{n=1}^{N} \ln h_{y_n}(\mathbf{x}_n)$$

$$\nabla E_{\text{in}}(\mathrm{W}) = -\frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial \mathbf{w}_i} \ln \frac{\exp\left(\mathbf{w}_{y_n}^T \mathbf{x}_n\right)}{\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)}$$

$$= -\frac{1}{N} \sum_{n=1}^{N} \frac{\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)}{\exp\left(\mathbf{w}_{y_n}^T \mathbf{x}_n\right)} \times \frac{\exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) \mathbf{x}_n \sum_{k=1}^{K} \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) - \exp\left(\mathbf{w}_{y_n}^T \mathbf{x}_n\right) \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) \mathbf{x}_n}{\left(\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)\right)^2}$$

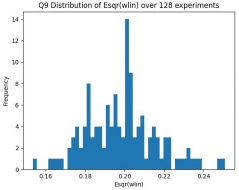
$$= -\frac{1}{N} \sum_{n=1}^{N} \frac{\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)}{\left(\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)\right)^2} \times \frac{\exp\left(\mathbf{w}_{y_n}^T \mathbf{x}_n\right) \mathbf{x}_n \sum_{k=1}^{K} \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) - \exp\left(\mathbf{w}_{y_n}^T \mathbf{x}_n\right) \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) \mathbf{x}_n}{\exp\left(\mathbf{w}_{y_n}^T \mathbf{x}_n\right)}$$

$$= -\frac{1}{N} \sum_{n=1}^{N} \frac{\left[\sum_{k=1}^{K} \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right) - \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right)\right]}{\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)} \cdot \mathbf{x}_n = \frac{1}{N} \sum_{n=1}^{N} \left(h_i(\mathbf{x}) - 1\right) \mathbf{x}_n$$

$$\text{if } j \neq i, \quad -\frac{1}{N} \sum_{n=1}^{N} \frac{\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)}{\exp\left(\mathbf{w}_y^T \mathbf{x}_n\right)} \times \frac{-\exp\left(\mathbf{w}_y^T \mathbf{x}_n\right) \exp\left(\mathbf{w}_i^T \mathbf{x}_n\right)}{\left(\sum_{k=1}^{K} \exp\left(\mathbf{w}_k^T \mathbf{x}_n\right)\right)^2} \mathbf{x}_n = \frac{1}{N} \sum_{n=1}^{N} h_i(\mathbf{x}) \cdot \mathbf{x}_n$$

$$\rightarrow \quad \nabla E_{\text{in}}(\mathrm{W}) = -\frac{1}{N} \begin{bmatrix} | & & | \\ \sum_{n=1}^{N} \left(h_1(\mathbf{x}_n) - [\![y_n = 1]\!]\right) \mathbf{x}_n & \sum_{n=1}^{N} \left(h_2(\mathbf{x}_n) - [\![y_n = 2]\!]\right) \mathbf{x}_n & \cdots \\ | & & | \end{bmatrix}$$

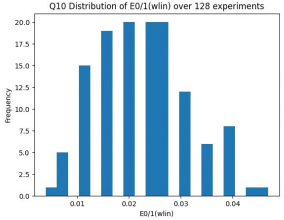
9. (20 points, *) Implement the linear regression algorithm taught in the lecture. Run the algorithm for 128 times, each with a different random seed for generating the two data sets above. Plot a histogram to visualize the distribution of $E_{\text{in}}^{\text{sqr}}(\mathbf{w}_{\text{LIN}})$, where $E_{\text{in}}^{\text{sqr}}$ denotes the *averaged* squared error over N examples. What is the median $E_{\text{in}}^{\text{sqr}}$ over the 128 experiments?



median = 0.199638 #

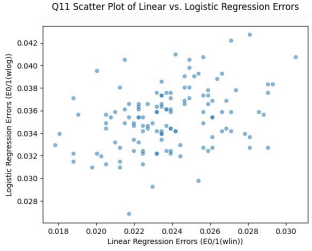
10. (20 points, *) Following the previous problem, plot a histogram to visualize the distribution of $E_{\text{in}}^{0/1}(\mathbf{w}_{\text{LIN}})$, where $E_{\text{in}}^{0/1}$ denotes the averaged 0/1 error over N examples (i.e. using $\mathbf{w}_{\text{LIN}}$ for binary classification). What is the median $E_{\text{in}}^{0/1}$ over the 128 experiments?

(Note: You can choose to run 128 new experiments in this problem, or just re-use the 128 hypotheses $\mathbf{w}_{\text{LIN}}$ and test data sets obtained from the previous problem.)



median = 0.02343 #

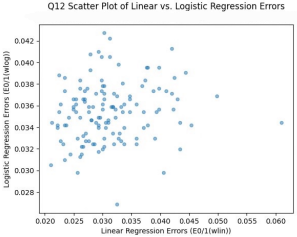
11. (20 points, *) Consider two algorithms. The first one, $\mathcal{A}$, is the linear regression algorithm above. The second one $\mathcal{B}$ is logistic regression, trained with fixed learning rate gradient descent with $\eta = 0.1$ for $T = 500$ iterations, starting from $\mathbf{w}_0 = \mathbf{0}$. Run the algorithms on the same $\mathcal{D}$, and record $[E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D})), E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}))]$. Repeat the process for 128 times, each with a different random seed for generating the training and test data sets above. Plot a scatter plot for $[E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D})), E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}))]$. What is the median of $E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D}))$ and what is the median of $E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}))$?



$E_{out}^{0/1}(A(D))$ median : 0.0235595703125

$E_{out}^{0/1}(B(D))$ median : 0.035400390625

12. (20 points, *) Following the previous problem, in addition to the 256 examples in $\mathcal{D}$, add 16 outlier examples generated from the following process to your training data (but not to your test data). All outlier examples will be labeled $y = +1$ and $\mathbf{x} = [1, x_1, x_2]$ where (x_1, x_2) comes from a normal distribution of mean $[0, 6]$ and covariance $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.3 \end{bmatrix}$. Name the new training data set $\mathcal{D}'$. Run the algorithms on the same $\mathcal{D}'$, and record $[E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D}')), E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}'))]$. Repeat the process for 128 times, each with a different random seed for generating the training and test data sets above. Plot a scatter plot for $[E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D}')), E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}'))]$. What is the median of $E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D}'))$ and what is the median of $E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}'))$? Compare your results to the previous problem. Describe your findings.



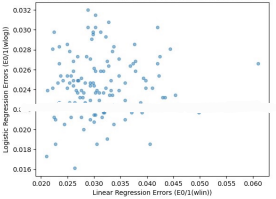
$E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D}'))$: median.lin:0.0301513671875

$E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}'))$: median.log:0.035400390625

Median of linear regression algorithm becomes larger than training without outlier data,
while median of logistic algorithms doesn't change much.

Since linear regression make error ^2, when we use noise in training data, it may sacrifice original best g,
modify itself to reduce the error from noice when training.

The plot also shows that Eout(log) doesn't increase with Eout(Lin), but in most of cases Eout(log) is bigger than Eout(Lin),
so I run another plot to see if iteration=20000(which means it walks more steps to modify), Eout'(log) : 0.0241
the consequence indicates Eout(log) becomes smaller than Eout(Lin) .That is, if we increase iteration, logistic have the
ability to perform better that linear.



13. (Bonus 20 points) When using Newton's method for solving logistic regression, as discussed in Problem 6, each update $\mathbf{v}$ is calculated by

$$\mathbf{v} = -(\mathrm{X}^T\mathrm{D}\mathrm{X})^{-1}\nabla E_{\text{in}}(\mathbf{w}_t)$$

when $(\mathrm{X}^T\mathrm{D}\mathrm{X})$ is invertible. In linear regression, when $\mathrm{X}^T\mathrm{X}$ is invertible, the optimal

$$\mathbf{w}_{\text{LIN}} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathbf{y}.$$

If we can express $-\nabla E_{\text{in}}(\mathbf{w}_t)$ as some $\tilde{\mathrm{X}}^T\tilde{\mathbf{y}}$, and $\mathrm{X}^T\mathrm{D}\mathrm{X}$ as some $\tilde{\mathrm{X}}^T\tilde{\mathrm{X}}$, then each iteration of Newton-solving logistic regression is performing an internal linear regression! State the internal linear regression problem—in particular, what are $\tilde{\mathrm{X}}$ and $\tilde{\mathbf{y}}$?

13. $h_t(x) = \dfrac{1}{1+\exp(w^T x)}$, $D = \begin{bmatrix} h_t(x_1)(1-h_t(x_1)) & & \\ & h_t(x_2)(1-h_t(x_2)) & \\ & & \ddots \\ & & & h_t(x_N)(1-h_t(x_N)) \end{bmatrix}$

① $X^T D X = X^T \sqrt{D} \cdot \sqrt{D}^T X$, $\tilde{X} = \sqrt{D} X$ #

② $-\nabla E_{in}(w_t) = \frac{1}{N}\sum_{n=1}^{N} y_n x_n (1-h_t(x_n)) = X^T [y(1-h_t(w))]$

$\frac{1}{N}\left(X^T\sqrt{D}\right)\left[\sqrt{D}^{-1} y (1-h_t(x))\right] = \tilde{X}^T \tilde{y}$, $\tilde{X}^T \tilde{y} = X^T \sqrt{D}^T \tilde{y} = X^T \sqrt{D}\,\tilde{y}$

$\tilde{y} = \frac{1}{N}\sqrt{D}^{-1} y (1-h_t(x))$ #