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

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

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

$$f_{\text{CIA}}(\mathbf{x}) = \text{sign}\big(P(+1|\mathbf{x}) - \alpha\big).$$

Prove what the value of α should be.

$f_{0/1}(x) = \text{sign}\left(P(+1|x) - \frac{1}{2}\right)$ the ideal mini target would be +1 if the $P(+1|x) > \frac{1}{2}$

CIA f / g

	+1	−1
+1	0	1
−1	1000	0

(no error)

we don't want a fake positive to happen

⇒ the ratio of positive and negative become 1 : 1000

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

$P(+1|x) : P(-1|x) = 1000 : 1$

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

$\alpha = \frac{1000}{1001}$

#

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

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

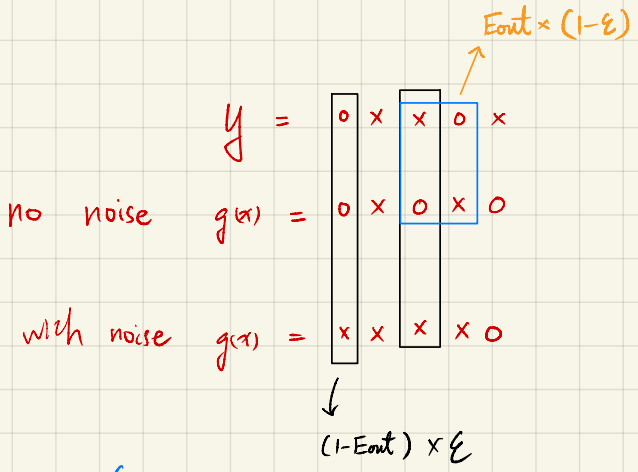
Now, assume that $g(\mathbf{x})$ is put in a noisy test environment where

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

Derive

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

as a function of $E_{\text{out}}(g)$ and ϵ.



Error = (Eout) mistake happens while (1-ε) $P(y=+f(x)|x)$ + g(x) make no mistake (1-Eout)

but noise happens $P(y=-f(x)|x)$ (ε)

$\Rightarrow$ Eout × (1-ε) + (1-Eout) ε

= Eout + ε − 2ε Eout

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.)*

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

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

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

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

$$= \frac{2}{N}\sum_{n=1}^{N}(w\,x_n^2 - x_n y_n) = 0$$

$$\sum_{n=1}^{N} w\,x_n^2 = \sum_{n=1}^{N} x_n y_n$$

$$w = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2} \quad *$$

(Hint: This is linear regression in $\mathbb{R}$ without the added x_0.)

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).

$$E_{in} = \frac{\int_0^1 (w_0 + w_1 x - ax^2 - b)^2 \, dx}{1-0}$$

$$\frac{\partial E_{in}}{\partial w_0} = \int_0^1 2(w_0 + w_1 x - ax^2 - b)\, dx = 0$$

$$\frac{\partial E_{in}}{\partial w_1} = \int_0^1 2(w_0 + w_1 x - ax^2 - b)\, x \, dx = 0$$

$$\left(w_0 x + \frac{1}{2} w_1 x^2 - \frac{1}{3} a x^3 - bx \right) \Big|_0^1 = 0$$

$$\Rightarrow w_0 + \frac{1}{2} w_1 - \frac{1}{3} a - b = 0 \quad \text{— (1)}$$

$$\frac{1}{2} w_0 x^2 + \frac{1}{3} w_1 x^3 - \frac{1}{4} a x^4 - \frac{1}{2} b x^2 \Big|_0^1 = 0$$

$$\Rightarrow w_0 + \frac{2}{3} w_1 - \frac{1}{2} a - b = 0 \quad \text{— (2)}$$

$$(1)(2) \Rightarrow \frac{1}{6} w_1 = \frac{1}{6} a \Rightarrow w_1^* = a \; \#$$

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

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 $\mathbf{w}_{\text{LIN}}$ 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 $\mathbf{w}'_{\text{LIN}}$. Derive $\mathbf{w}'_{\text{LIN}}$ as a function of $\mathbf{w}_{\text{LIN}}$ and (a, b).

$$w_{lin} = (X^TX)^{-1}X^Ty$$

$$E_{in}(w) = \frac{1}{N}\|Xw-(ay+b)\|^2$$

$$= \frac{1}{N}\left(w^TX^TXw - 2w^TX^T(ay+b) + (ay+b)^T(ay+b)\right)$$

$$\nabla E_{in}(w) = \frac{1}{N}\left(2X^TXw - 2X^T(ay+b)\right)$$

$$= \frac{2}{N}\left(X^TXw - X^T(ay+b)\right)$$

$$\Rightarrow \quad X^TXw = X^T(ay+b)$$

$$w'_{lin} = aw_{lin} + (X^TX)^{-1}X^Tb$$

$\Rightarrow$ if $y' = ay$ $\quad w'_{lin} = a\,w_{lin}$

$$X\,a\,w_{lin} = ay$$

$$\begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & & & \\ \vdots & & & & \\ & & & & x_{nn} \end{bmatrix} \begin{bmatrix} a\,w_0 \\ a\,w_1 \\ \vdots \\ a\,w_n \end{bmatrix} = \begin{bmatrix} a\,y_1 \\ a\,y_2 \\ \vdots \\ a\,y_n \end{bmatrix}$$

$$X\,w'_{lin} = ay+b$$

$$\begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & & & \\ \vdots & & & & \end{bmatrix} \begin{bmatrix} a\,w_0+b \\ a\,w_1+0 \\ \vdots \\ a\,w_n+0 \end{bmatrix} = \begin{bmatrix} a\,y_1+b \\ a\,y_2+b \\ \vdots \\ a\,y_n+b \end{bmatrix}$$

$$\Rightarrow \quad w'_{lin} = a\,w_{lin} + b\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}_{n\times 1}$$

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

$$\mathbf{b}_E(\mathbf{w}) = \nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\partial E}{\partial w_1}(\mathbf{w}) \\ \frac{\partial E}{\partial w_2}(\mathbf{w}) \\ \vdots \\ \frac{\partial E}{\partial w_d}(\mathbf{w}) \end{bmatrix}_{d\times 1} \quad \text{and} \quad \mathrm{A}_E(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2}(\mathbf{w}) & \frac{\partial^2 E}{\partial w_1 \partial w_2}(\mathbf{w}) & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_d}(\mathbf{w}) \\ \frac{\partial^2 E}{\partial w_2 \partial w_1}(\mathbf{w}) & \frac{\partial^2 E}{\partial w_2^2}(\mathbf{w}) & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_d}(\mathbf{w}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_d \partial w_1}(\mathbf{w}) & \frac{\partial^2 E}{\partial w_d \partial w_2}(\mathbf{w}) & \cdots & \frac{\partial^2 E}{\partial w_d^2}(\mathbf{w}) \end{bmatrix}_{d\times d}.$$

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 $\mathrm{X}^T\mathrm{D}\mathrm{X}$, 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.

Note: The h_t that we use here is slightly different from that being used in class—this particular h_t predicts $P(-1|\mathbf{x})$ instead of $P(+1|\mathbf{x})$. It was a harmless typo (sorry!!). By default, the TAs will grade by this definition of h_t. But if you really want to choose to use the in-class definition, you are allowed to do so ***as long as you state your choice clearly for the TAs****.*

$h_t(x) = \frac{1}{1+\exp(w_t^T x)}$ $\quad$ $\frac{1}{1+\exp(-w_t^T x)} = h_t(-x) = 1 - h_t(x)$

$E_{in} = \frac{1}{N}\sum_{n=1}^{N} \ln(1+\exp(-y_n w^T x_n))$

$b_E(w) = \nabla E_{in} = \frac{1}{N}\sum_{n=1}^{N} \frac{\exp(-y_n w^T x_n)}{1+\exp(-y_n w^T x_n)}(-y_n x_{n,i}) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{\exp(y_n w^T x_n)+1}(-y_n x_{n,i})$

$= \frac{1}{N}\sum_{n=1}^{N} h_t(y_n x_n)(-y_n x_{n,i})$ assume $y=-1 \Rightarrow \frac{1}{N}\sum_{n=1}^{N}(1-h_t(x_n))(-y_n x_{n,i})$

$A_E(w) = \nabla b_E(w) = \frac{1}{N}\sum_{n=1}^{N} \frac{\exp(-y_n w^T x_n)(-y_n x_{n,j})(1+\exp(-y_n w^T x_n)) - \exp^2(-y_n w^T x_n)(-y_n x_{n,j})}{(1+\exp(-y_n w^T x_n))^2}(-y_n x_{n,i})$

$= \frac{1}{N}\sum_{n=1}^{N} \frac{\exp(-y_n w^T x_n)(-y_n x_{n,j})(-y_n x_{n,i})\left[(1+\exp(-y_n w^T x_n) - \exp(-y_n w^T x_n)\right]}{(1+\exp(-y_n w^T x_n))^2}$

$= \frac{1}{N}\sum_{n=1}^{N} \frac{\exp(-y_n w^T x_n)(-y_n x_{n,j})(-y_n x_{n,i})}{(1+\exp(-y_n w^T x_n))^2}$

$= \frac{1}{N}\sum_{n=1}^{N} \frac{\exp(-y_n w^T x_n)(y_n x_{n,j})(y_n x_{n,i})}{(1+\exp(-y_n w^T x_n))^2} = X^T D X$

$D_{i,j} = \frac{1}{N} h_t(y_n x_n)(1-h_t(y_n x_n)) y_{n,i} \times y_{n,j} = \frac{1}{N}(1-h_t(x_n)) h_t(x_n) y_{n,i} y_{n,j}$

$\because$ D is diagnal matrix $\Rightarrow$ $y_{n,i} \times y_{n,j} = 1$

$\therefore$ $D_{i,i} = \frac{1}{N}(1-h_t(x_n)) h_t(x_n)$

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.

SGD: $w_{t+1} = w_t + \eta \times -\nabla err(w_t, x_n, y_n)$

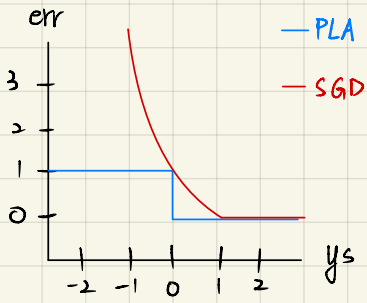
$= w_t + \eta \times (-\nabla(\max(0, 1 - y_n x^T w)^2))$

$w_t - \eta(2 x_n x_n^T w - 2 y_n x_n)$

$= w_t - 2\eta(s - y_n) x_n$

$= w_t - 2\eta(ys - 1) x_n y_n$

$$\begin{cases} w_t + 2\eta(1 - ys) x_n y_n & ys < 1 \\ w_t & ys > 1 \end{cases}$$



PLA: $w_{t+1} = w_t + [y_n \neq \text{sign}(w_t^T x_n)] y_n x_n$

we can see that the origin PLA will update w when $ys < 0$, or we can say $y_n \neq \text{sign}(w_t^T x_n)$, while PLA applying SGD update w when $ys < 1$. so the updating condition (criteria) is different. Second, the origin PLA update w with $1 \times y_n x_n$, while the SGD version of PLA update $2\eta(1-ys) x_n y_n$. which means w will update more when ys is much smaller than 1. With ys become larger, w update less. However, the step size η is usually < 1 (0.1), so update rate is slower than normal PLA. To sum up, the original PLA being faster for immediate respone, but less stable. while the SGD PLA offering more fine-grained control over learning but slower.

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.)*

$$E_{in}(W) = \frac{-1}{N}\sum_{n=1}^{N} \ln h_y(x_n)$$

$$\nabla E_{in}(w) = \left[\frac{\partial E_{in}}{\partial w_1}(w) \quad \frac{\partial E_{in}}{\partial w_2}(w) \cdots \right]$$

$$\nabla E_{in}(w) = -\frac{1}{N}\sum_{n=1}^{N} \frac{\partial}{\partial w} \ln \frac{\exp(w_y^T x_n)}{\sum_{i=1}^{k} \exp(w_i^T x_n)}$$

$$\text{if } y=k: -\frac{1}{N}\sum_{n=1}^{N} \frac{\sum_{i=1}^{k} \exp(w_i^T x_n)}{\exp(w_k^T x_n)} \times \frac{\exp(w_k^T x_n)\, x_n \sum_{i=1}^{k} \exp(w_i^T x_n) - \exp(w_i^T x_n) \exp(w_k^T x_n)\, x_n}{\left(\sum_{i=1}^{k} \exp(w_i^T x_n)\right)^2}$$

$$\frac{\partial}{\partial w_1}\sum_{k=1}^{k} \exp(w_i^T x_n) = \frac{\partial}{\partial w_1}\left(\exp(w_1^T x_n) + \exp(w_2^T x_n) \cdots\right)$$

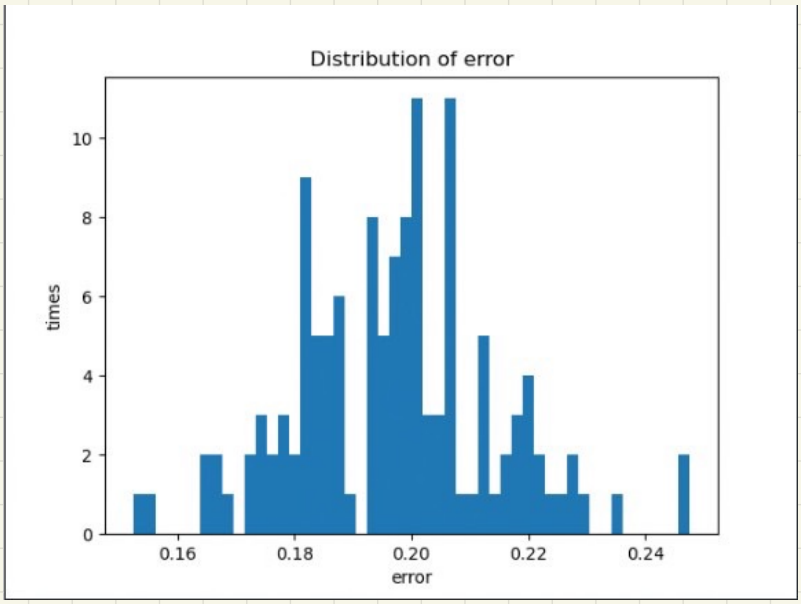
$$= \exp(w_1^T x_n) \begin{bmatrix} x_{n,1} \\ x_{n,2} \\ \vdots \end{bmatrix} = \exp(w_1^T x_n)\, x_n$$

$$= -\frac{1}{N}\sum_{n=1}^{N} \frac{\left(\sum_{i=1}^{k} \exp(w_i^T x_n) - \exp(w_k^T x_n)\right)}{\sum_{i=1}^{k} \exp(w_i^T x_n)} x_n = \frac{1}{N}\sum_{n=1}^{N} (h_k(x) - 1)\, x_n$$

$$\text{if } y \neq k: -\frac{1}{N}\sum_{n=1}^{N} \frac{\sum_{i=1}^{k} \exp(w_i^T x_n)}{\exp(w_y^T x_n)} \times \frac{-\exp(w_y^T x_n)\exp(w_k^T x_n)}{\sum_{i=1}^{k} \exp(w_i^T x_n)^2} x_n = \frac{1}{N}\sum_{n=1}^{N} h(x) \cdot x_n$$

$$\nabla E_{in}(W) = -\frac{1}{N}\begin{bmatrix} \sum_{n=1}^{N} (h_1(x_n) - [\![y=1]\!])\, x_n & \sum_{n=1}^{N} (h_2(x_n) - [\![y=2]\!])\, x_n \cdots \end{bmatrix}$$

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?

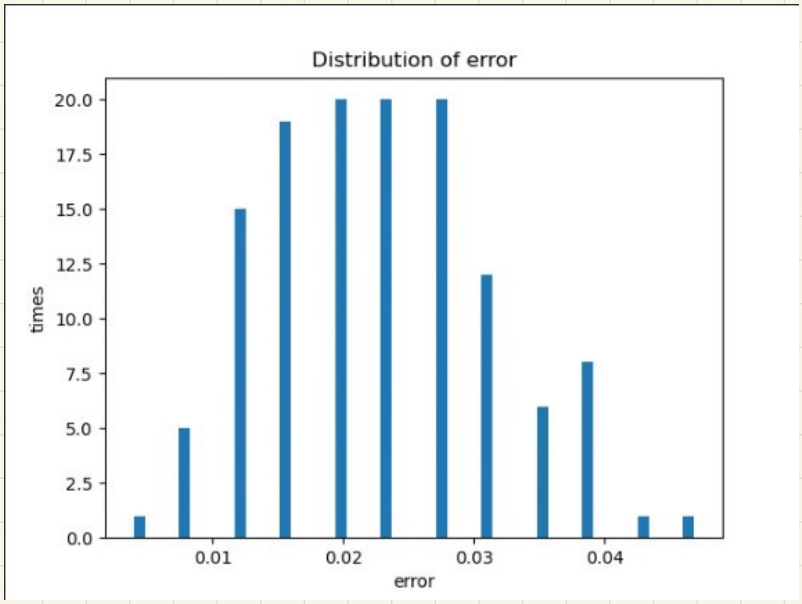


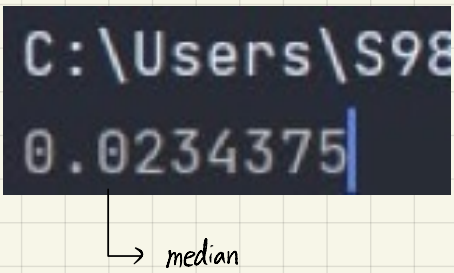

```
C:\Users\S9801\.cond
0.19797113883145015
```

↳ median

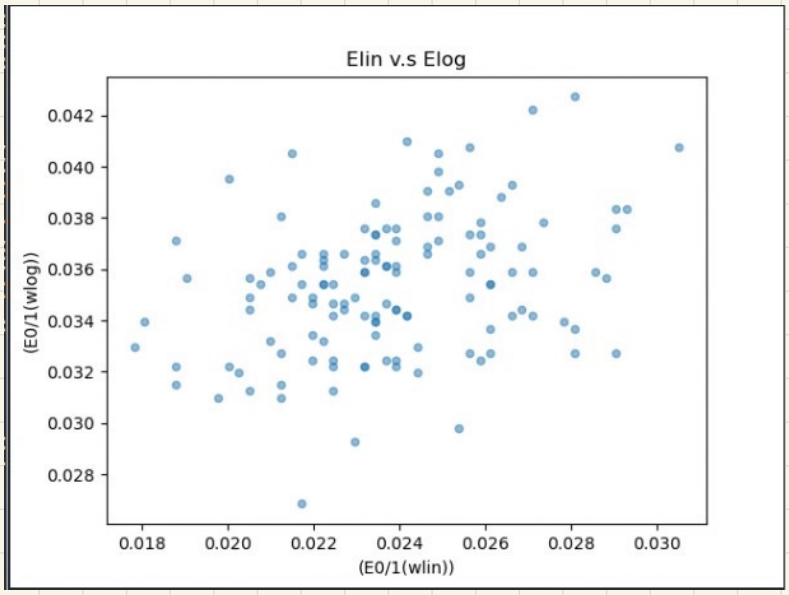
10. (20 points, *) Following the previous problem, plot a histogram to visualize the distribution of $E_{\text{in}}^{0/1}(\mathbf{w}_{\text{LIN}})$, where $E_{\text{in}}^{0/1}$ denotes the averaged 0/1 error over N examples (i.e. using $\mathbf{w}_{\text{LIN}}$ for binary classification). What is the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





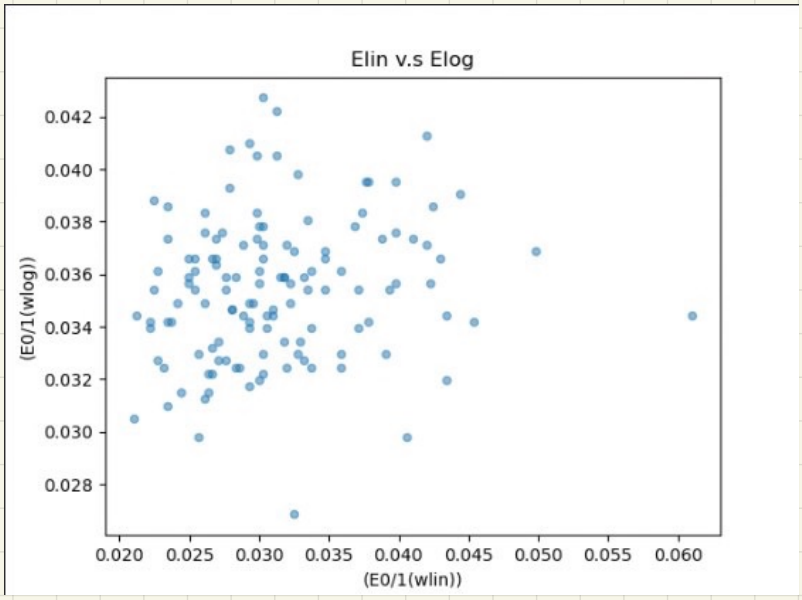
11. (20 points, *) Consider two algorithms. The first one, $\mathcal{A}$, is the linear regression algorithm above. The second one $\mathcal{B}$ is logistic regression, trained with fixed learning rate gradient descent with $\eta = 0.1$ for $T = 500$ iterations, starting from $\mathbf{w}_0 = \mathbf{0}$. 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 $E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}))$?




```
C:\Users\S9801\.conda\envs\ML3\py
0.0235595703125 0.035400390625
```

↳ $E_{\text{out}}^{0/1}(\mathcal{A}(\mathcal{D}))$ ↳ $E_{\text{out}}^{0/1}(\mathcal{B}(\mathcal{D}))$

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.




```
0.0301513671875 0.035400390625
```

the median Eout of linear regression becomes bigger and the distribution of Ein is wider, while the median Eout of logistic regress and the distribution is similar to problem 11.

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 $\tilde{\mathbf{y}}$?

$$h_t(x) = \frac{1}{1+\exp(w_t^T x)}$$

$$D = \begin{bmatrix} h(x_n)(1-h(x_n)) & & 0 \\ & \ddots & \\ 0 & & \end{bmatrix} = \begin{bmatrix} \sqrt{h(x_n)(1-h(x_n))} & & 0 \\ & \ddots & \\ 0 & & \end{bmatrix} \begin{bmatrix} \sqrt{h(x_n)(1-h(x_n))} & & 0 \\ & \ddots & \\ 0 & & \end{bmatrix}$$

$$X^T D X = X^T \sqrt{D}^T \sqrt{D} X = \tilde{X}^T \tilde{X}$$

$$\Rightarrow \tilde{X} = \sqrt{D} X \ \#$$

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

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

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