Homework #4

RELEASE DATE: 10/21/2024

DUE DATE: 11/04/2024, BEFORE 13:00 on GRADESCOPE

QUESTIONS ARE WELCOMED ON DISCORD (INFORMALLY) OR VIA EMAILS (FORMALLY).

You will use Gradescope to upload your scanned/printed solutions. Any programming language/platform is allowed.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

This homework set comes with 200 points and 20 bonus points. In general, every homework set would come with a full credit of 200 points, with some possible bonus points.

1. (10 points, auto-graded) In class, we introduced our version of the cross-entropy error function

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

based on the definition of $y_n \in \{-1, +1\}$. If we transform y_n to $y_n' \in \{0, 1\}$ by $y_n' = \frac{y_n+1}{2}$, which of the following error function is equivalent to E_{in} above?

[a] $\frac{1}{N}\sum_{n=1}^{N}\Big(+y_n' \ln\theta(+\mathbf{w}^T\mathbf{x}_n) + (1-y_n')\ln\theta(-\mathbf{w}^T\mathbf{x}_n)\Big)$

[b] $\frac{1}{N}\sum_{n=1}^{N}\Big(+y_n' \ln\theta(-\mathbf{w}^T\mathbf{x}_n) + (1-y_n')\ln\theta(+\mathbf{w}^T\mathbf{x}_n)\Big)$

[c] $\frac{1}{N}\sum_{n=1}^{N}\Big(-y_n' \ln\theta(+\mathbf{w}^T\mathbf{x}_n) - (1-y_n')\ln\theta(-\mathbf{w}^T\mathbf{x}_n)\Big)$

[d] $\frac{1}{N}\sum_{n=1}^{N}\Big(-y_n' \ln\theta(-\mathbf{w}^T\mathbf{x}_n) - (1-y_n')\ln\theta(+\mathbf{w}^T\mathbf{x}_n)\Big)$

[e] none of the other choices

(Note: In the error functions in the choices, there is a form of $p\log q + (1-p)\log(1-q)$, which is the origin of the name "cross-entropy.")

2. (10 points, auto-graded) In the perceptron learning algorithm, we find one example $(\mathbf{x}_{n(t)}, y_{n(t)})$ that the current weight vector $\mathbf{w}_t$ mis-classifies, and then update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}.$$

A variant of the algorithm finds *all* examples $(\mathbf{x}_n, y_n)$ that the weight vector $\mathbf{w}_t$ mis-classifies (e.g. $y_n \neq \text{sign}(\mathbf{w}_t^T\mathbf{x}_n)$), and then update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \frac{\eta}{N} \sum_{n:\ y_n \neq \text{sign}(\mathbf{w}_t^T\mathbf{x}_n)} y_n\mathbf{x}_n.$$

The variant can be viewed as optimizing some $E_{\text{in}}(\mathbf{w})$ that is composed of one of the following point-wise error functions with a fixed learning rate gradient descent (neglecting any non-differentiable spots of E_{in}). What is the error function?

[a] $\text{err}(\mathbf{w}, \mathbf{x}, y) = -y\mathbf{w}^T\mathbf{x}$

[b] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$

[c] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \max(0, -y\mathbf{w}^T\mathbf{x})$

[d] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \min(0, 1 - y\mathbf{w}^T\mathbf{x})$

[e] $\text{err}(\mathbf{w}, \mathbf{x}, y) = \min(0, -y\mathbf{w}^T\mathbf{x})$

3. (10 points, auto-graded) In regression, we sometimes want to deal with the situation where an over-estimate is worse than an under-estimate. This calls for the following asymmetric squared error

$$\text{err}_\alpha(\mathbf{w}, \mathbf{x}, y) = (1+\alpha)\left[\!\left[\mathbf{w}^T\mathbf{x} \geq y\right]\!\right](\mathbf{w}^T\mathbf{x} - y)^2 + (1-\alpha)\left[\!\left[\mathbf{w}^T\mathbf{x} < y\right]\!\right](\mathbf{w}^T\mathbf{x} - y)^2$$

for some $0 < \alpha < 1$. When using stochastic gradient descent to minimize an $E_{\text{in}}(\mathbf{w})$ that is composed of the asymmetric squared error function, which of the following is the update direction $-\nabla \text{err}_{\text{exp}}(\mathbf{w}, \mathbf{x}_n, y_n)$ for the chosen $(\mathbf{x}_n, y_n)$ with respect to $\mathbf{w}_t$?

[a] $2(1 + \alpha^2 \cdot \text{sign}\left(\mathbf{w}_t^T\mathbf{x}_n - y_n\right))\left(y_n - \mathbf{w}_t^T\mathbf{x}_n\right)\mathbf{x}_n$

[b] $2(1 - \alpha^2 \cdot \text{sign}\left(\mathbf{w}_t^T\mathbf{x}_n - y_n\right))\left(y_n - \mathbf{w}_t^T\mathbf{x}_n\right)\mathbf{x}_n$

[c] $2(1 + \alpha \cdot \text{sign}\left(\mathbf{w}_t^T\mathbf{x}_n - y_n\right))\left(y_n - \mathbf{w}_t^T\mathbf{x}_n\right)\mathbf{x}_n$

[d] $2(1 - \alpha \cdot \text{sign}\left(\mathbf{w}_t^T\mathbf{x}_n - y_n\right))\left(y_n - \mathbf{w}_t^T\mathbf{x}_n\right)\mathbf{x}_n$

[e] none of the other choices

4. (10 points, auto-graded) After "visualizing" the data and noticing that all $\mathbf{x}_1$,, $\mathbf{x}_2$, …, $\mathbf{x}_n$ are distinct, Dr. Transformer magically decides the following transform

$$\mathbf{\Phi}(\mathbf{x}) = (\left[\!\left[\mathbf{x} = \mathbf{x}_1\right]\!\right], \left[\!\left[\mathbf{x} = \mathbf{x}_2\right]\!\right], \ldots, \left[\!\left[\mathbf{x} = \mathbf{x}_N\right]\!\right]).$$

That is, $\mathbf{\Phi}(\mathbf{x})$ is a N-dimensional vector whose n-th component is 1 if and only if $\mathbf{x} = \mathbf{x}_n$. If we run linear regression (i.e. squared error) after applying this transform, what is the optimal $\tilde{\mathbf{w}}$? For simplicity, please ignore $z_0 = 1$. That is, we do not pad the $\mathcal{Z}$ space examples with a constant feature.

[a] $\mathbf{1}_N$, the N-dimensional vector of all 1's.

[b] $\mathbf{y}$

[c] $\mathbf{0}_N$, the N-dimensional vector of all 0's.

[d] $-\mathbf{y}$

[e] $\frac{1}{2}(\mathbf{1}_N + \mathbf{y})$

(Note: Be sure to also check what $E_{in}(\tilde{\mathbf{w}})$ is, and think about what $E_{out}(\tilde{\mathbf{w}})$ might be!)

5. (20 points, human-graded) 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! :-)

An iterative optimization algorithm using the "optimal direction" above for updating $\mathbf{w}$ is called **Newton's method**, which can be viewed as "improving" gradient descent by using more information about E. 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

$$h_t(\mathbf{x}) = \theta(\mathbf{w}_t^T \mathbf{x}).$$

Express the Hessian $\mathrm{A}_E(\mathbf{w}_t)$ with $E = E_{\text{in}}$ as $\mathrm{X}^T\mathrm{DX}$, 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.

6. (20 points, human-graded) 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

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

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

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

When minimizing $E_{\text{in}}(\mathrm{W})$ with SGD, we update the $\mathrm{W}^{(t)}$ at the t-th iteration to $\mathrm{W}^{(t+1)}$ by

$$\mathrm{W}^{(t+1)} \leftarrow \mathrm{W}^{(t)} + \eta \cdot \mathrm{V},$$

where V is a $(d+1) \times K$ matrix whose k-th column is an update direction for the k-th weight vector. Assume that an example $(\mathbf{x}_n, y_n)$ is used for the SGD update above. It can be proved that

$$\mathrm{V} = \mathbf{x}_n \cdot \mathbf{u}^T,$$

where $\mathbf{u}$ is a vector in $\mathbb{R}^K$. What is $\mathbf{u}$? List your derivation steps.

7. (20 points, human-graded) Following the previous problem, consider a data set with $K = 2$ and obtain the optimal solution from MLR as $(\mathbf{w}_1^*, \mathbf{w}_2^*)$. Now, relabel the same data set by replacing y_n with $y_n' = 2y_n - 3$ to form a binary classification data set, and run logistic regression (what we have learned in class) to get an optimal solution $\mathbf{w}_{\text{lr}}$. Express $\mathbf{w}_{\text{lr}}$ as a function of $(\mathbf{w}_1^*, \mathbf{w}_2^*)$. List your derivation steps.

8. (20 points, human-graded) Consider the target function $f(x) = 1 - 2x^2$. 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. Then, sample two examples x_1 and x_2 uniformly from $[0, 1]$ to form the training set $\mathcal{D} = \{(x_1, f(x_1)), (x_2, f(x_2))\}$, and use linear regression to get g for approximating the target function with respect to the squared error. You can neglect the degenerate cases where x_1 and x_2 are the same. What is $\mathbb{E}_{\mathcal{D}}\left(|E_{\text{in}}(g) - E_{\text{out}}(g)|\right)$? List your derivation steps.

9. (20 points, human-graded) In Lecture 13, we discussed about adding "virtual examples" (hints) to help combat overfitting. One way of generating virtual examples is to add a small noise to the input vector $\mathbf{x} \in \mathbb{R}^{d+1}$ (including the 0-th component x_0) For each $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ in our training data set, assume that we generate virtual examples $(\tilde{\mathbf{x}}_1, y_1), (\tilde{\mathbf{x}}_2, y_2), \ldots, (\tilde{\mathbf{x}}_N, y_N)$ where $\tilde{\mathbf{x}}_n$ is simply $\mathbf{x}_n + \boldsymbol{\epsilon}$ and the noise vector $\boldsymbol{\epsilon} \in \mathbb{R}^{d+1}$ is generated i.i.d. from a multivariate normal distribution $\mathcal{N}(\mathbf{0}_{d+1}, \sigma^2 \cdot \mathrm{I}_{d+1})$. The vector $\boldsymbol{\epsilon}$ is a random vector that varies for each virtual example. Here $\mathbf{0}_{d+1} \in \mathbb{R}^{d+1}$ denotes the all-zero vector and I_{d+1} is an identity matrix of size $d+1$.

Recall that when training the linear regression model, we need to calculate $\mathrm{X}^T\mathrm{X}$ first. Define the hinted input matrix

$$\mathrm{X}_h = \begin{bmatrix} | & \cdots & | & | & \cdots & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_N & \tilde{\mathbf{x}}_1 & \cdots & \tilde{\mathbf{x}}_N \\ | & \cdots & | & | & \cdots & | \end{bmatrix}^T.$$

The expected value $\mathbb{E}(\mathrm{X}_h^T\mathrm{X}_h)$, where the expectation is taken over the (Gaussian)-noise generating process above, is of the form

$$\alpha \mathrm{X}^T\mathrm{X} + \beta\sigma^2\mathrm{I}_{d+1}.$$

Derive the correct (α, β).

(*Note: The form here "hints" you that the expected value is related to the matrix being inverted for regularized linear regression—see Lecture 14. That is, data hinting "by noise" is closely related to regularization. If* $\mathbf{x}$ *contains the pixels of an image, the virtual example is a Gaussian-noise-contaminated image with the same label, e.g.* `https://en.wikipedia.org/wiki/Gaussian_noise`. *Adding such noise is a very common technique to generate virtual examples for images.*)

10. (20 points, code needed, human-graded) Next, we use a real-world data set to study linear and polynomial regression. We will reuse the `cpusmall_scale` data set at

`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/cpusmall_scale`

to save you some efforts.

First, execute the following. The first four steps are similar to what you have done in Homework 3. The last step is new. We will take $N = 64$ (instead of 32).

The data set contains 8192 examples. In each experiment, you are asked to

(1) randomly sample N out of 8192 examples as your training data.

(2) add $x_0 = 1$ to each example, as always.

(3) run linear regression on the N examples, using any reasonable implementations of $\mathrm{X}^\dagger$ on the input matrix X, to get $\mathbf{w}_{lin}$

(4) evaluate $E_{\text{in}}(\mathbf{w}_{lin})$ by averaging the squared error over the N examples; estimate $E_{\text{out}}(\mathbf{w}_{lin})$ by averaging the squared error over the rest $(8192 - N)$ examples

(5) implement the SGD algorithm for linear regression using the results on pages 10 and 12 of Lecture 11. Pick one example uniformly at random in each iteration, take $\eta = 0.01$ and initialize $\mathbf{w}$ with $\mathbf{w}_0 = \mathbf{0}$. Run the algorithm for 100000 iterations, and record $E_{\text{in}}(\mathbf{w}_t)$ and $E_{\text{out}}(\mathbf{w}_t)$ whenever t is a multiple of 200.

Repeat the steps above 1126 times, each with a different random seed. Calculate the average $(E_{\text{in}}(\mathbf{w}_{lin}), E_{\text{out}}(\mathbf{w}_{lin}))$ over the 1126 experiments. Also, for every $t = 200, 400, \ldots$, calculate the average $(E_{\text{in}}(\mathbf{w}_t), E_{\text{out}}(\mathbf{w}_t))$ over the 1126 experiments. Plot the average $E_{\text{in}}(\mathbf{w}_t)$ as a function of t. On the same figure, plot the average $E_{\text{out}}(\mathbf{w}_t)$ as a function of t. Then, show the average $E_{\text{in}}(\mathbf{w}_{lin})$ as a horizontal line, and the average $E_{\text{out}}(\mathbf{w}_{lin})$ as another horizontal line. Describe your findings.

Finally, provide the first page of the snapshot of your code as a proof that you have written the code.

11. (20 points, code needed, human-graded) Next, consider the following *homogeneous* order-Q polynomial transform

$$\mathbf{\Phi}(\mathbf{x}) = (1, x_1, x_2, ..., x_{12}, x_1^2, x_2^2, ..., x_{10}^2, ..., x_1^Q, x_2^Q, ..., x_{12}^Q).$$

Transform the training and testing data according to $\mathbf{\Phi}(\mathbf{x})$ with $Q = 3$, and again implement steps 1-4 in the previous problem on $\mathbf{z}_n = \mathbf{\Phi}(\mathbf{x}_n)$ instead of $\mathbf{x}_n$ to get $\mathbf{w}_{poly}$. Repeat the four steps above 1126 times, each with a different random seed. Plot a histogram of $E_{\text{in}}^{sqr}(\mathbf{w}_{lin}) - E_{\text{in}}^{sqr}(\mathbf{w}_{poly})$. What is the average $E_{\text{in}}^{sqr}(\mathbf{w}_{lin}) - E_{\text{in}}^{sqr}(\mathbf{w}_{poly})$? This is the E_{in} gain for using (homogeneous) polynomial transform. Describe your findings.

Finally, provide the first page of the snapshot of your code as a proof that you have written the code.

12. (20 points, code needed, human-graded) Following from the 1126 experiments in the previous problem, plot a histogram of $E_{\text{out}}^{sqr}(\mathbf{w}_{lin}) - E_{\text{out}}^{sqr}(\mathbf{w}_{poly})$. What is the average $E_{\text{out}}^{sqr}(\mathbf{w}_{lin}) - E_{\text{out}}^{sqr}(\mathbf{w}_{poly})$? This is the E_{out} change for using (homogeneous) polynomial transform. Describe your findings.

Finally, provide the first page of the snapshot of your code as a proof that you have written the code.

13. (Bonus 20 points, human-graded) Define the multiplicative hypotheses as

$$\mathcal{H}_2 = \left\{ \tilde{h}_{\mathbf{u}}(\mathbf{x}) \colon \tilde{h}_{\mathbf{u}}(\mathbf{x}) = \text{sign}\left(u_0 + \prod_{i=1}^{d} (|x_i| + 1)^{u_i} \right) \right\}.$$

Compared with the linear hypotheses

$$\mathcal{H}_1 = \left\{ h_{\mathbf{w}}(\mathbf{x}) \colon h_{\mathbf{w}}(\mathbf{x}) = \text{sign}\left(w_0 + \sum_{i=1}^{d} w_i x_i \right) \right\},$$

the multiplicative hypotheses appear to be implementing much more complicated boundaries. Prove or disprove that $d_{vc}(\mathcal{H}_2) > d_{vc}(\mathcal{H}_1)$.