Homework #5

RELEASE DATE: 11/04/2024

DUE DATE: 11/18/2024, 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.

Experimentally, starting from this homework, we will allow each grading TA to give up to 2 clarity-bonus points for every human-graded problem if the TA believes that the answer is delivered with exceptional clarity, in addition to being correct. We hope that this encourages everyone to think about how to express your ideas clearly.

1. (10 points, auto-graded) Additive smoothing

`https://en.wikipedia.org/wiki/Additive_smoothing`

is a simple yet useful technique in estimating discrete probabilities. Consider the technique for estimating the head probability of a coin. Let $y_1, y_2, \dots, y_N$ denotes the flip results from a coin, with $y_n = 1$ meaning a head and $y_n = 0$ meaning a tail. Additive smoothing adds K "virtual flips", with K_+ of them being head and the other $(K - K_+)$ being tail. Then, the head probability is estimated by

$$\frac{(\sum_{n=1}^{N} y_n) + K_+}{N + K}$$

The estimate can be viewed as the optimal solution of

$$\min_{w_0 \in \mathbb{R}} \frac{1}{N} \sum_{n=1}^{N} (w_0 - y_n)^2 + \frac{K}{N} \Omega(w_0),$$

where $\Omega(w_0)$ is a "regularizer" to this estimation problem. What is $\Omega(w_0)$?

[a] $(w_0 - K_+)^2$

[b] $\left(w_0 - \frac{K - K_+}{K}\right)^2$

[c] $\left(w_0 - \frac{K_+(K - K_+)}{K^2}\right)^2$

[d] $\left(w_0 - \frac{K_+}{K}\right)^2$

[e] $(w_0 - K + K_+)^2$

2. (10 points, auto-graded) Consider the 1D decision stump model and a binary classification dataset of N examples $\{(x_n, y_n)\}_{n=1}^N$ that is linearly separable. That is, there exists some 1D decision stump h that achieves $E_{\text{in}}(h) = 0$. Consider a decision stump model that returns the midpoint between the closest positive and negative points as its threshold θ, and decides the best direction $s \in \{-1, +1\}$ by optimizing E_{in}. Such a model will always achieve $E_{\text{in}} = 0$ when the dataset is linearly separable. If the dataset contains at least two positive examples and at least two negative examples, which of the following is the tightest upper bound on the leave-one-out error of the decision stump model?

[a] 0
[b] $1/(2N)$
[c] $1/N$
[d] $2/N$
[e] $1/2$

3. (10 points, auto-graded) In Lecture 16, we talked about the probability to fit data perfectly when the labels are random. For instance, page 6 of Lecture 16 shows that the probability of fitting the randomly-labeled data perfectly with decision stumps is $(2N)/2^N$. Consider 5 different points in $\mathbb{R}$ as inputs, and a decision stump that minimizes E_{in} in terms of the 0/1 error to the lowest possible value. One way to measure the power of the model is to consider five random labels y_1, y_2, y_3, y_4, y_5, each in ± 1 and generated by i.i.d. fair coin flips, and then compute

$$\mathbb{E}_{y_1,y_2,y_3,y_4,y_5}\left(\min_{s\in\{-1,1\},\theta\in\mathbb{R}} E_{\text{in}}(s,\theta)\right),$$

where $h_{s,\theta} = s \cdot \text{sign}(x - \theta)$ as usual. For a perfect fitting, $\min E_{\text{in}}(s, \theta)$ will be 0; for a less perfect fitting (when the data is not decision-stump separable), the minimum $E_{\text{in}}(s, \theta)$ will be some non-zero value. The expectation above averages over all 32 possible combinations of y_1, y_2, y_3, y_4, y_5. What is the value of the expectation?

[a] 8/80
[b] 9/80
[c] 10/80
[d] 11/80
[e] 12/80

(*Note: It can be shown that* 1 *minus twice the expected value above is the same as the so-called empirical Rademacher complexity of decision stumps. Rademacher complexity, similar to the VC dimension, is another tool to measure the complexity of a hypothesis set. If a hypothesis set shatters some data points, zero E_{in} can always be achieved and thus Rademacher complexity is* 1*; if a hypothesis set cannot shatter some data points, Rademacher complexity provides a soft measure of how "perfect" the hypothesis set is.*)

4. (10 points, auto-graded) Consider a three-example one-dimensional dataset: $\{(x_n, y_n)\}_{n=1}^3 = \{(-4, -1), (0, +1), (4, -1)\}$, and a polynomial transform $\mathbf{\Phi}(x) = [1, x, x^2]^T$. Apply the hard-margin SVM on the transformed examples $\{(\mathbf{\Phi}(x_n), y_n)\}_{n=1}^3$ to get the optimal $(b^*, \mathbf{w}^*)$ in the transformed space. What is the margin achieved by the optimal solution (in the $\mathcal{Z}$ space)?

[a] 1
[b] 2
[c] 4
[d] 8
[e] 16

(*Hint: We encourage you to try to derive this manually to enrich your understanding of the hard-margin SVM. But given that it is a multiple-choice problem, technically nothing prevents you from using a QP solver to verify your answer.*)

5. (20 points, human-graded) Consider a regularization problem of the form

$$(P_R) \min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 + \frac{\lambda}{N}\sum_{i=0}^{d}\alpha_i w_i^2$$

with positive constants α_i. The typical L2 regularizer can be viewed as a special case of this new regularizer with $\boldsymbol{\alpha} = \mathbf{1}$, the vector of 1's.

Now, consider linear regression with virtual examples. That is, we add K virtual examples $(\tilde{\mathbf{x}}_1, \tilde{y}_1), (\tilde{\mathbf{x}}_2, \tilde{y}_2) \dots (\tilde{\mathbf{x}}_K, \tilde{y}_K)$ to the training dataset, and solve

$$(P_V) \min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{N+K}\left(\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 + \sum_{k=1}^{K}(\mathbf{w}^T\tilde{\mathbf{x}}_k - \tilde{y}_k)^2\right).$$

Let $K = d+1$, and for $k = 1, \dots, K$,

$$\begin{aligned} \tilde{\mathbf{x}}_k &= \left[\underbrace{0, \dots, 0}_{(k-1) \text{ dimensions}}, \sqrt{\lambda\alpha_{k-1}}, 0, \dots, 0\right]^T \in \mathbb{R}^{d+1} \\ \tilde{y}_k &= 0 \end{aligned}$$

Prove that the optimal $\mathbf{w}^*$ obtained by solving (P_V) is the same as the optimal solution obtained by solving (P_R).

(*Note: The results show that virtual examples, which were claimed to be a possible way to combat overfitting in Lecture 13, is mathematically related to regularization, another possible way to combat overfitting discussed in Lecture 14.*)

6. (20 points, human-graded) When performing L2-regularization on any twice-differentiable and convex $E_{\text{in}}(\mathbf{w})$ for some linear model of interest, the optimization problem can be written as:

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^{d+1}} \quad & E_{\text{aug}}(\mathbf{w}) \\ \text{subject to} \quad & E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}||\mathbf{w}||_2^2 \end{aligned}$$

Suppose $\mathbf{w}^*$ is the minimizer of $E_{\text{in}}(\mathbf{w})$. That is, $\nabla E_{\text{in}}(\mathbf{w}^*) = \mathbf{0}$. Take the second-order Taylor's expansion of $E_{\text{in}}(\mathbf{w})$ around $\mathbf{w}^*$, we can approximate $E_{\text{in}}(\mathbf{w})$ by

$$\tilde{E}_{\text{in}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}^*) + \underbrace{(\mathbf{w}-\mathbf{w}^*)^T\nabla E_{\text{in}}(\mathbf{w}^*)}_{0} + \frac{1}{2}(\mathbf{w}-\mathbf{w}^*)^T \ \text{H} \ (\mathbf{w}-\mathbf{w}^*)$$

where $\text{H} \in \mathbb{R}^{(d+1)\times(d+1)}$ is some Hessian matrix. The convexity ensures that H is positive semi-definite. Then, $E_{\text{aug}}(\mathbf{w})$ can be approximated by

$$\tilde{E}_{\text{aug}}(\mathbf{w}) = \tilde{E}_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\|\mathbf{w}\|^2.$$

Express the minimizer of $\tilde{E}_{\text{aug}}(\mathbf{w})$ as a function of λ, N, H and $\mathbf{w}^*$. List your derivation steps.

(*Note: The result demonstrates the difference between the minimizers of E_{in} and $\tilde{E}_{\text{aug}}$*)

7. (20 points, human-graded) For N labels $y_1, y_2, \ldots, y_N$ generated i.i.d. from some distribution of mean 0 and variance σ^2. Partition the first $(N-K)$ labels to be the training data, and the last K labels to be the validation data. If we estimate the mean by the "hypothesis" of 0, the expected validation error

$$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-0)^2\right),$$

where the expectation is taken on the process of generating N labels, is simply σ^2 by the independence of the y_n's. Now, assume that we take the average on the first $(N-K)$ examples to estimate the mean instead. That is,

$$\bar{y}=\frac{1}{N-K}\sum_{n=1}^{N-K}y_n$$

What is the expected validation error

$$\mathbb{E}\left(\frac{1}{K}\sum_{n=N-K+1}^{N}(y_n-\bar{y})^2\right)$$

in terms of σ^2, N and K? List your derivation steps.

8. (20 points, human-graded) For N labels $y_1, y_2, \ldots, y_N$ generated i.i.d. from some distribution of a finite mean. If some w_0 is used to estimate the mean, define $E_{\text{in}}(w_0)=\frac{1}{N}\sum_{n=1}^{N}(w_0-y_n)^2$. It can be easily shown that $w_0^*=\frac{1}{N}\sum_{n=1}^{N}y_n$ is the optimal solution—it is just a "constant" regression. Consider a algorithm $\mathcal{A}_{avg}$ that always returns such an optimal solution for any non-empty set of labels. Prove that for $N\geq 2$,

$$E_{\text{loocv}}(\mathcal{A}_{avg})=\left(\frac{N}{N-1}\right)^2 E_{\text{in}}(w_0^*).$$

(*Note: The result demonstrates that E_{in} is a more optimistic estimate than E_{loocv}.*)

9. (20 points, human-graded) Consider a binary classifier g such that

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

When deploying the classifier to a test distribution of $P(y=+1)=P(y=-1)=1/2$, we get $E_{\text{out}}(g)=\frac{1}{2}\epsilon_+ + \frac{1}{2}\epsilon_-$. Now, if we deploy the classifier to another test distribution $P(y=-1)=p$ instead of $1/2$, the $E_{\text{out}}(g)$ under this test distribution will then change to a different value. Note that under this test distribution, a constant classifier g_c that always predicts $+1$ will suffer from $E_{\text{out}}(g_c)=p$ as it errors on all the negative examples. At what $p\in[0,1]$ will our binary classifier g be as good as (or as bad as) the constant classifier g_c in terms of E_{out}? Please express p as a function of ϵ_+ and ϵ_-. List your derivation steps.

10. (20 points, human-graded) Consider L1-regularized logistic regression

$$\mathbf{w}_\lambda = \operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{N}\|\mathbf{w}\|_1 + \frac{1}{N}\sum_{n=1}^{N} \ln(1+\exp(-y_n \mathbf{w}^T \mathbf{x}_n)),$$

where $\mathbf{x}_n$ and $\mathbf{w}$ are $(d+1)$ dimensional vectors, as usual. That is, each $\mathbf{x}$ is padded with $x_0 = 1$. We will use the `mnist.scale` dataset at

`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.bz2`

for training, and

`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/mnist.scale.t.bz2`

for testing (evaluating E_{out}). The datasets are originally for multi-class classification. We will study one of the sub-problems within one-versus-one decomposition: class 2 versus class 6.

We call the algorithm for solving the problem above as $\mathcal{A}_\lambda$. The problem guides you to use LIBLINEAR (`https://www.csie.ntu.edu.tw/~cjlin/liblinear/`), a machine learning package developed at National Taiwan University, to solve this problem. In addition to using the default options, what you need to do when running LIBLINEAR are

- set option `-s 6`, which corresponds to solving L1-regularized logistic regression
- set option `-c C`, with a parameter value of `C` calculated from the λ that you want to use; read `README` of the software package to figure out how `C` and your λ should relate to each other.

LIBLINEAR can be called from the command line or from major programming languages like python. *You may find other options of the package useful for some of the problems below. It is up to you to decide whether to use them or not after checking the README.*

After extracting the data of only 2 and 6 from the training and test sets, we will consider the dataset as a *binary classification problem* and take the "regression for classification" approach with regularized logistic regression (see Page 6 of Lecture 10). So please evaluate all errors below with the 0/1 error.

First select the best λ^* as

$$\operatorname*{argmin}_{\log_{10}\lambda \in \{-2,-1,0,1,2,3\}} E_{\text{in}}(\mathbf{w}_\lambda).$$

Break the tie, if any, by selecting the largest λ. Note that `-s 6` is a randomized solver that may return different results on different random seeds, even with the same data. Repeat the experiments above for 1126 times, each with a different random seed. Plot a histogram of $E_{\text{out}}(g)$, where g is returned by running $\mathcal{A}_\lambda$ with λ^*, over the 1126 experiments. In addition, plot a histogram of the average number of non-zero components in each g.

If you run LIBLINEAR in the command line, please include screenshots of the commands/results; if you run LIBLINEAR from any programming language, please include screenshots of the first page of your code.

11. (20 points, human-graded) Now randomly split the given training examples to two sets: 8000 as the sub-training set and the rest as the validation set. Run $\mathcal{A}_\lambda$ on *only* the sub-training set to get $\mathbf{w}_\lambda^-$ (the weight vector within the g^- returned), and validate $\mathbf{w}_\lambda^-$ with the validation set to get $E_{\text{val}}(\mathbf{w}_\lambda^-)$. Select the best λ^* as

$$\operatorname*{argmin}_{\log_{10}\lambda \in \{-2,-1,0,1,2,3\}} E_{\text{val}}(\mathbf{w}_\lambda^-).$$

Break the tie, if any, by selecting the largest λ. Then, re-run $\mathcal{A}_\lambda$ on the whole training set with λ^* to get g. Repeat the experiments above for 1126 times, each with a different random seed. Plot a histogram of $E_{\text{out}}(g)$ over the 1126 experiments. Compare the E_{out} distribution that you got in Problem 10 to the E_{out} distribution that you get in this problem. Describe your findings.

If you run LIBLINEAR in the command line, please include screenshots of the commands/results; if you run LIBLINEAR from any programming language, please include screenshots of the first page of your code.

12. (20 points, human-graded) Now conduct 3-fold cross validation on the training data. Select the best λ^* as

$$\underset{\log_{10}\lambda\in\{-2,-1,0,1,2,3\}}{\operatorname{argmin}} E_{\text{cv}}(\mathcal{A}_\lambda).$$

Break the tie, if any, by selecting the largest λ. Then, re-run $\mathcal{A}_\lambda$ on the whole training set with λ^* to get g. Repeat the experiments above for 1126 times, each with a different random seed. Plot a histogram of $E_{\text{out}}(g)$ over the 1126 experiments. Compare the E_{out} distribution that you got in Problem 11 to the E_{out} distribution that you get in this problem. Describe your findings.

If you run LIBLINEAR in the command line, please include screenshots of the commands/results; if you run LIBLINEAR from any programming language, please include screenshots of the first page of your code.

13. (Bonus 20 points, human-graded) The elastic net is a famous regularization scheme based on a mixture of L1 and L2 regularization. The model solves

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{N}\|\mathbf{y}-\mathrm{X}\mathbf{w}\|_2^2 + \frac{\lambda_1}{N}\|\mathbf{w}\|_1 + \frac{\lambda_2}{N}\|\mathbf{w}\|_2^2$$

Because $\|\mathbf{w}\|_1$ is not a smooth function, it is harder to solve it by naïve gradient descent. Instead, one can consider solving it by the so-called coordinate descent. Each iteration of coordinate descent updates *one* of the variables (weights), instead of all variables altogether. The update is based on the *optimal* solution on that variable, while keeping other variables fixed. That is, the update is

$$w_i^{(t+1)} \leftarrow \underset{w_i\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \sum_{j\neq i} w_j^{(t)}x_{n,j} - w_i x_{n,i}\right)^2 + \frac{\lambda_1}{N}\left(\underbrace{\sum_{j\neq i}|w_j^{(t)}|}_{\text{constant}} + |w_i|\right) + \frac{\lambda_2}{N}\left(\underbrace{\sum_{j\neq i}(w_j^{(t)})^2}_{\text{constant}} + w_i^2\right).$$

The update actually has a simple closed-form solution of the form

$$w_i^{(t+1)} \leftarrow \alpha\cdot\max(\beta, 0).$$

where $\alpha\in\{-1,+1\}$ is for possible sign flipping and $\beta\in\mathbb{R}$. The update above says if $\beta\le 0$, $w_i^{(t+1)}$ will be set to 0. This operation demonstrates how elestic net (and similarly L1) achieves sparsity in its solution. Derive (α,β) as a function of N, $x_{n,j}$, y_n, λ_1, λ_2, and $w_j^{(t)}$.