EEE598: Statistical Machine Learning: From Theory to Practice

Instructor: Dr. Lalitha Sankar
Homework Assignment #1
Total points: 100

Due Date and Time: 9/22/2022 11:59 PM

Student Name: _____ ASU Id: _____

Instructions:

(1) Every problem should begin on a new page (use the \newpage command).

(2) Page numbers for each problem should be entered in Gradescope when uploading HW

(3) Furthermore, **page numbers for every sub-problem within a problem should also be entered in Gradescope**.

(4) Points are given for steps and work – show all work and steps and clarifications/arguments for the steps

(5) Some rules on Python code files:

(i) Filename convention: For every problem, the corresponding Python filename should follow the following nomenclature: ⟨FirstInitialLastName-HWxx-Problem#Subproblem#⟩. For example, for HW3, problem 7a, my Python file should have the name: LSankar-HW3-7a.xxx

(ii) All subproblems can be in one file but the FILENAME has to be clarifying as in (i) above.

(iii) All Python files generated for an HW can be zipped and uploaded but again, if filenames are misleading then points will be deducted.

(iv) All code will also be checked for complete overlap with other student's code.

(v) **Every file should be independently executable**.

(6) Enjoy!

Some instructions on how to submit solutions on Gradescope and Canvas:

(1) **All plots have to be included in the PDF file uploaded to Gradescope**. It is not the TA's or grader's job to find it on the canvas submission. Plots not in the PDF submission on Gradescope will NOT graded.

(2) **Plots without legend on what you are plotting them for will lead to points being deducted**. Keep in mind you have several plots and it is important for us to know what these plots correspond to.

(3) **Canvas submission should be restricted to only CODE**.

(4) **Please include a copy of your code in your Gradescope submission** – this can make grading easier for us as a first run (before we run your code using the canvas upload). This is particularly helpful if we find that your basic code is correct and your results are correct.

Additional Notes:

a. Please review all homework guidance above before submitting to Gradescope.

b. Please provide succinct answers along with succinct reasoning for all your answers.

c. Similarly, when discussing the experimental results, concisely create tables and/or figures when appropriate to organize the experimental results. **All your explanations, tables, and figures for any particular part of a question must be grouped together**.

d. **Late homework submissions will not be accepted 48 hours after the deadline.** Note that the maximum attainable score decreases by half each day after the due date. (i.e. the maximum score is 50% 24 hours after the deadline and 25% 48 hours after the deadline.)

Maximum Likelihood Estimation (MLE)

Total: *[15 points]*

1. At the 2033 ASU homecoming block party, a local philanthropist hosts an unusual dart game. While the principle that a player wins once their dart hits the bulls eye holds in this game too, the philanthropist wants students to feel encouraged to keep working on their hand-eye coordination and so allows a player to continue attempting (i.e., throwing darts) until they win at which point they are rewarded a dollar. (this is why this is in the future as it seems to be wishful thinking that one could be paid for this but let's proceed ;).

A smart graduate student standing nearby realizes that is a fun ML problem. She begins by letting a random variable K denote the number of attempts it takes to successfully hit the target. Of course, she also recognizes immediately that in each attempt the outcome is either a 1 (successfully hit the bulls eye) or a 0 (failure to do so). She was able to observe only for a short period of time during which time she collected 7 samples (each sample is the number of attempts **to** success, including the success throw, for some player). The following is then her dataset $\mathcal{D}$ of attempts to success for 7 players:

$$\mathcal{D} = [1, 3, 2, 4, 5, 2, 3].$$

Answer the following questions:

(a) *[1 points]* Given the above information, can you model a single attempt as a random variable? If so, write down the distribution for this random variable? [Note: please denote the RV with the appropriate notation, its sample space, and the probability distribution over this sample space.]

(b) *[1 points]* The graduate student decides to assign a probability p to succeed in be the same in each attempt, also assuming that every attempt is independent of any other attempt. Based on the answer to (1.a) above, what is the likelihood of a player succeeding in this target game in their first attempt, i.e., what is $P[K = 1]$?

(c) *[1 points]* If a player is successful at the $k = 5$ attempt, what were the outcomes of the previous 4 attempts?

(d) *[1 points]* Recalling that K denotes the number of attempts it takes to succeed, what is $P_K[5]$? [Hint: use the fact that the outcome in each attempt for a player is independent of any other attempt to write the total probability over 5 attempts.]

(e) *[1 points]* Now generalize (1.d) to any arbitrary $k \in \{1, 2, \ldots\}$, i.e., write down the formula for $P[K = k]$, i.e., the likelihood that it takes k attempts to succeed.

(f) *[5 points]* For the dataset $\mathcal{D}$ given, what is your maximum likelihood estimate of p?

(g) *[5 points]* Now let's generalize the above estimate of p to any set of 7 observations, i.e., $\mathcal{D} = [k_1, \ldots, k_7]$. Write down the MLE estimate of p for $\mathcal{D} = [k_1, \ldots, k_7]$. [Hint: here k_i denotes the number to success for the i^{th} observation made by the graduate student. Your answer will be in terms of the k_i.]

Error from fitting

Total: *[20 points]*

2. Suppose we obtain N labeled samples $\{(x_i, y_i)\}_{i=1}^N$ from an underlying distribution $\mathcal{D}$ (i.e., every example $\{(x_i, y_i)\}_{i=1}^N$ is an independent and identifically distributed instantiation from $\mathcal{D}$). Suppose we break this into N_{train} and N_{test} samples for our training and test set. Recall our definition of the true least squares error

$$\epsilon(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[(f(x) - y)^2]$$

(the subscript $(x, y) \sim \mathcal{D}$ makes clear that our input-output pairs are sampled according to $\mathcal{D}$). Our training and test losses are defined as:

$$\widehat{\epsilon}_{\text{train}}(f) = \frac{1}{N_{\text{train}}} \sum_{(x,y)\in\text{Training Set}} (f(x) - y)^2 \tag{1}$$

$$\widehat{\epsilon}_{\text{test}}(f) = \frac{1}{N_{\text{test}}} \sum_{(x,y)\in\text{Test Set}} (f(x) - y)^2 \tag{2}$$

We then train our algorithm (say linear least squares regression) using the training set to obtain an $\widehat{f}$.

(a) *[10 points]* (bias: the test error) For all fixed f (before we've seen any data) show that

$$\mathbb{E}_{\text{train}}[\widehat{\epsilon}_{\text{train}}(f)] = \mathbb{E}_{\text{test}}[\widehat{\epsilon}_{\text{test}}(f)] = \epsilon(f).$$

Conclude by showing that the test error is an unbiased estimate of our true error. Specifically, show that:

$$\mathbb{E}_{\text{test}}[\widehat{\epsilon}_{\text{test}}(\widehat{f})] = \epsilon(\widehat{f})$$

(b) *[10 points]* (bias: the train/dev error) Is the above equation true (in general) with regards to the training loss? Specifically, does $\mathbb{E}_{\text{train}}[\widehat{\epsilon}_{\text{train}}(\widehat{f})]$ equal $\mathbb{E}_{\text{train}}[\epsilon(\widehat{f})]$? If so, why? If not, give a clear argument as to where your previous argument breaks down.

HINTS: Note that the expectation operator is over the joint distribution of the dataset indicated. Use the i.i.d. property to obtain the joint distribution. Finally, explaining the steps or intuition clearly even if it is hard to do completely in a mathematical fashion can earn points.

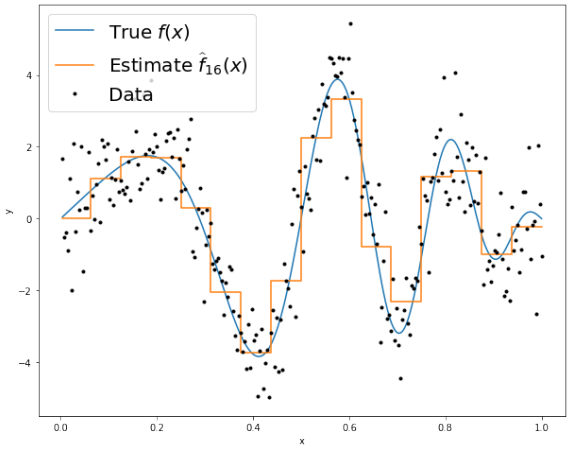

Figure 1: Step function estimator with $n = 256$, $m = 16$, and $\sigma^2 = 1$.

Bias Variance tradeoff

Total: *[30 points]*

3. For $i = 1, \ldots, n$ let $x_i = i/n$ and $y_i = f(x_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ for some unknown f we wish to approximate at values $\{x_i\}_{i=1}^n$. We will approximate f with a step function estimator which is given as follow. For some $m \leq n$ such that n/m is an integer define the estimator

$$\widehat{f}_m(x) = \sum_{j=1}^{n/m} c_j \mathbf{1}\{x \in \left(\frac{(j-1)m}{n}, \frac{jm}{n}\right]\} \quad \text{where} \quad c_j = \frac{1}{m} \sum_{i=(j-1)m+1}^{jm} y_i. \tag{3}$$

where the indicator function $\mathbf{1}$ with respect to a set A is such that $\mathbf{1}_A(x) = 1$, if $x \in A$, and $\mathbf{1}_A(x) = 0$, otherwise. Note that this estimator just partitions $\{1, \ldots, n\}$ into intervals $\{1, \ldots, m\}, \{m+1, \ldots, 2m\}, \ldots, \{n-m+1, \ldots, n\}$ and predicts the average of the observations within each interval (see Figure 1).

By the bias-variance decomposition at some x_i we have

$$\mathbb{E}\left[(\widehat{f}_m(x_i) - f(x_i))^2\right] = \underbrace{(\mathbb{E}[\widehat{f}_m(x_i)] - f(x_i))^2}_{\text{Bias}^2(x_i)} + \underbrace{\mathbb{E}\left[(\widehat{f}_m(x_i) - \mathbb{E}[\widehat{f}_m(x_i)])^2\right]}_{\text{Variance}(x_i)} \tag{4}$$

(a) *[5 points]* Intuitively, how do you expect the bias and variance to behave for small values of m? What about large values of m?

(b) *[5 points]* If we define $\bar{f}^{(j)} = \frac{1}{m}\sum_{i=(j-1)m+1}^{jm} f(x_i)$ and the *average bias-squared* as $\frac{1}{n}\sum_{i=1}^n (\mathbb{E}[\widehat{f}_m(x_i)] - f(x_i))^2$, show that

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbb{E}[\widehat{f}_m(x_i)] - f(x_i))^2 = \frac{1}{n}\sum_{j=1}^{n/m}\sum_{i=(j-1)m+1}^{jm}(\bar{f}^{(j)} - f(x_i))^2 \tag{5}$$

(c) *[5 points]* If we define the *average variance* as $\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n (\widehat{f}_m(x_i) - \mathbb{E}[\widehat{f}_m(x_i)])^2\right]$, show (both equalities)

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(\widehat{f}_m(x_i) - \mathbb{E}[\widehat{f}_m(x_i)])^2\right] = \frac{1}{n}\sum_{j=1}^{n/m} m\mathbb{E}[(c_j - \bar{f}^{(j)})^2] = \frac{\sigma^2}{m} \tag{6}$$

(d) *[15 points]* Let $n = 256$, $\sigma^2 = 1$, and $f(x) = 4\sin(\pi x)\cos(6\pi x^2)$. For values of $m = 1, 2, 4, 8, 16, 32$ plot the average empirical error $\frac{1}{n}\sum_{i=1}^n (\widehat{f}_m(x_i) - f(x_i))^2$ using randomly drawn data as a function of m on the x-axis. On the same plot, using parts b and c of above, plot the *average bias-squared*, the *average variance*, and their sum (the average error). Thus, there should be 4 lines on your plot, each described in a legend.

This setup of each x_i deterministically placed at i/n is a good approximation for the more natural setting where each x_i is drawn uniformly at random from $[0, 1]$. In fact, one can redo this problem and obtain nearly identical conclusions, but the calculations are messier.

Ridge Regression

Total: *[10 points]*

4. *[10 points]* In this problem we will study the behavior of ridge regression when the number of training examples is about the same number of dimensions. Given training data $\{(x_i, y_i)\}_{i=1}^n$ for $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, recall that the ridge regression solution with parameter λ is equal to

$$\widehat{w} = \arg\min_w \sum_{i=1}^{n} (x_i^T w - y_i)^2 + \lambda \|w\|_2^2. \tag{7}$$

In matrix notation this has the closed form solution $\widehat{w} = (X^T X + \lambda I)^{-1} X^T y$. Usually we don't like to use inverses but for this problem you may use this solution because we will choose the sample sizes relatively small. First, generate some data:

```
import numpy as np
train_n = 100
test_n = 1000
d = 100
X_train = np.random.normal(0,1, size=(train_n,d))
a_true = np.random.normal(0,1, size=(d,1))
y_train = X_train.dot(a_true) + np.random.normal(0,0.5,size=(train_n,1))
X_test = np.random.normal(0,1, size=(test_n,d))
y_test = X_test.dot(a_true) + np.random.normal(0,0.5,size=(test_n,1))
```

For interpretability, consider the normalized training error $\frac{\|X_{\text{train}} w - y_{\text{train}}\|}{\|y_{\text{train}}\|}$ and test error $\frac{\|X_{\text{test}} w - y_{\text{test}}\|}{\|y_{\text{test}}\|}$. Note that for a trivial solution of $w = 0$, the all zeros vector, these normalized errors would equal 1. Using the closed-form solution of above, plot the normalized training error and normalized test error on the y-axis for $\lambda = \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$ on the x-axis. Because each draw of data will give you a different curve, repeat the experiment 30 times and average the curves from the trials to produce a plot. As λ grows, provide an explanation for why the training and test error behaves as it does.

Ridge Regression on MNIST

Total: *[25 points]*

5. In this problem we will implement a least squares classifier for the MNIST data set. The task is to classify handwritten images of numbers between 0 to 9.

You are **NOT** allowed to use any of the prebuilt classifiers in `sklearn`. Feel free to use any method from `numpy` or `scipy`. Remember: if you are inverting a matrix in your code, you are probably doing something wrong (Hint: look at `scipy.linalg.solve`).

Get the data from `https://pypi.python.org/pypi/python-mnist`.
Load the data as follows:

```
from mnist import MNIST

def load_dataset():
    mndata = MNIST('./data/')
    X_train, labels_train = map(np.array, mndata.load_training())
    X_test, labels_test = map(np.array, mndata.load_testing())
    X_train = X_train/255.0
    X_test = X_test/255.0
```

Each example has features $x_i \in \mathbb{R}^d$ (with $d = 28*28 = 784$) and label $z_j \in \{0, \ldots, 9\}$. You can visualize a single example x_i with `imshow` after reshaping it to its original 28×28 image shape (and noting that the label z_j is accurate). We wish to learn a predictor $\widehat{f}$ that takes as input a vector in $\mathbb{R}^d$ and outputs an index in $\{0, \ldots, 9\}$. We define our training and testing classification error on a predictor f as

$$\widehat{\epsilon}_{\text{train}}(f) = \frac{1}{N_{\text{train}}} \sum_{(x,z) \in \text{Training Set}} \mathbf{1}\{f(x) \neq z\} \tag{8}$$

$$\widehat{\epsilon}_{\text{test}}(f) = \frac{1}{N_{\text{test}}} \sum_{(x,z) \in \text{Test Set}} \mathbf{1}\{f(x) \neq z\} \tag{9}$$

We will use one-hot encoding of the labels, i.e. of (x, z) the original label $z \in \{0, \ldots, 9\}$ is mapped to the standard basis vector e_z where e_z is a vector of all zeros except for a 1 in the $(z+1)^{\text{th}}$ position. We adopt the notation where we have n data points in our training objective with features $x_i \in \mathbb{R}^d$ and label one-hot encoded as $y_i \in \{0, 1\}^k$ where in this case $k = 10$ since there are 10 digits.

(a) *[5 points]* In this problem we will choose a linear classifier to minimize the regularized least squares objective:

$$\widehat{W} = \text{argmin}_{W \in \mathbb{R}^{d \times k}} \sum_{i=0}^{n} \|W^T x_i - y_i\|_2^2 + \lambda \|W\|_F^2 \tag{10}$$

Note that $\|W\|_F$ corresponds to the Frobenius norm of W, i.e. $\|W\|_F^2 = \sum_{i=1}^{d} \sum_{j=1}^{k} W_{i,j}^2$. To classify a point x_i we will use the rule $\arg\max_{j=0,\ldots,9} e_{(j+1)}^T \widehat{W}^T x_i$. Note that if $W = \begin{bmatrix} w_1 & \ldots & w_k \end{bmatrix}$ then

$$\sum_{i=0}^{n} \|W^T x_i - y_i\|_2^2 + \lambda \|W\|_F^2 = \sum_{j=1}^{k} \left[\sum_{i=1}^{n} (e_j^T W^T x_i - e_j^T y_i)^2 + \lambda \|W e_j\|^2 \right] \tag{11}$$

$$= \sum_{j=1}^{k} \left[\sum_{i=1}^{n} (w_j^T x_i - e_j^T y_i)^2 + \lambda \|w_j\|^2 \right] \tag{12}$$

$$= \sum_{j=1}^{k} \left[\|X w_j - Y e_j\|^2 + \lambda \|w_j\|^2 \right] \tag{13}$$

where $X = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^\top \in \mathbb{R}^{n \times d}$ and $Y = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}^\top \in \mathbb{R}^{n \times k}$. Show that

$$\widehat{W} = (X^T X + \lambda I)^{-1} X^T Y \tag{14}$$

(b) *[8 points]* Code up a function called `train` that takes as input $X \in \mathbb{R}^{n \times d}$, $Y \in \{0,1\}^{n \times k}$, $\lambda > 0$, and returns $\widehat{W}$. Code up a function called `predict` that takes as input $W \in \mathbb{R}^{d \times k}$, $X' \in \mathbb{R}^{m \times d}$ and returns an m-length vector with the ith entry equal to $\arg\max_{j=0,\dots,9} e_j^T W^T x_i'$ where x_i' is a column vector representing the ith example from X'.

Train $\widehat{W}$ on the MNIST training data with $\lambda = 10^{-4}$ and make label predictions on the test data. What is the training and testing error (they should both be about 15%)?

(c) *[10 points]* We just fit a classifier that was linear in the pixel intensities to the MNIST data. For classification of digits the raw pixel values are very, very bad features: it's pretty hard to separate digits with linear functions in pixel space. The standard solution to this is to come up with some transform $h : \mathbb{R}^d \to \mathbb{R}^p$ of the original pixel values such that the transformed points are (more easily) linearly separable. In this problem, you'll use the feature transform:

$$h(x) = \cos(Gx + b).$$

where $G \in \mathbb{R}^{p \times d}$, $b \in \mathbb{R}^p$, and **the cosine function is applied elementwise**. We'll choose G to be a *random* matrix, with each entry sampled i.i.d. from a Gaussian with mean $\mu = 0$ and variance $\sigma^2 = 0.1$, and b to be a random vector sampled i.i.d. from the uniform distribution on $[0, 2\pi]$. The big question is: *how do we choose p?* Cross-validation, of course!

Randomly partition your training set into proportions 80/20 to use as a new training set and validation set, respectively. Using the `train` function you wrote above, train a $\widehat{W}^p$ for different values of p and plot the classification training error and validation error on a single plot with p on the x-axis. Be careful, your computer may run out of memory and slow to a crawl if p is too large ($p \leq 6000$ should fit into 4 GB of memory that is a minimum for most computers, but if you're having trouble you can set p in the several hundreds). You can use the same value of λ as above but feel free to study the effect of using different values of λ and σ^2 for fun.

(d) *[2 points]* Instead of reporting just the test error, which is an unbiased estimate of the *true* error, we would like to report a *confidence interval* around the test error that contains the true error.

Lemma 1. *(Hoeffding's inequality) Fix $\delta \in (0,1)$. If for all $i = 1, \dots, m$ we have that X_i are i.i.d. random variables with $X_i \in [a,b]$ and $\mathbb{E}[X_i] = \mu$ then*

$$\mathbb{P}\left(\left| \left(\frac{1}{m} \sum_{i=1}^{m} X_i \right) - \mu \right| \geq \sqrt{\frac{(b-a)^2 \log(2/\delta)}{2m}} \right) \leq \delta \tag{15}$$

We will use the above equation to construct a confidence interval around the true classification error $\epsilon(\widehat{f}) = \mathbb{E}_{\text{test}}[\widehat{\epsilon}_{\text{test}}(\widehat{f})]$ since the test error $\widehat{\epsilon}_{\text{test}}(\widehat{f})$ is just the average of indicator variables taking values in $\{0,1\}$ corresponding to the ith test example being classified correctly or not, respectively, where an error happens with probability $\mu = \epsilon(\widehat{f}) = \mathbb{E}_{\text{test}}[\widehat{\epsilon}_{\text{test}}(\widehat{f})]$, the *true* classification error.

Let $\widehat{p}$ be the value of p that approximately minimizes the validation error on the plot you just made and use $\widehat{f}(x) = \arg\max_j x^T \widehat{W}^{\widehat{p}} e_j$ to compute the classification test error $\widehat{\epsilon}_{\text{test}}(\widehat{f})$. Use Hoeffding's inequality, of above, to compute a confidence interval that contains $\mathbb{E}_{\text{test}}[\widehat{\epsilon}_{\text{test}}(\widehat{f})]$ (i.e., the *true* error) with probability at least 0.95 (i.e., $\delta = 0.05$). Report $\widehat{\epsilon}_{\text{test}}(\widehat{f})$ and the confidence interval.