Homework #1

CSE 446: Machine Learning
Professor Kevin Jamieson and Professor Anna Karlin
Due: 4/22/19 11:59 PM
100 points

Maximum Likelihood Estimation (MLE)

1. You're a Seahawks fan, and the team is six weeks into its season. The number touchdowns scored in each game so far are given below:

$$[1, 3, 3, 0, 1, 5].$$

Let's call these scores $x_1, \dots, x_6$. Based on your (assumed iid) data, you'd like to build a model to understand how many touchdowns the Seahaws are likely to score in their next game. You decide to model the number of touchdowns scored per game using a *Poisson distribution*. The Poisson distribution with parameter λ assigns every non-negative integer $x = 0, 1, 2, \dots$ a probability given by

$$\mathrm{Poi}(x|\lambda) = e^{-\lambda}\frac{\lambda^x}{x!}.$$

So, for example, if $\lambda = 1.5$, then the probability that the Seahawks score 2 touchdowns in their next game is $e^{-1.5} \times \frac{1.5^2}{2!} \approx 0.25$. To check your understanding of the Poisson, make sure you have a sense of whether raising λ will mean more touchdowns in general, or fewer.

a. *[5 points]* Derive an expression for the maximum-likelihood estimate of the parameter λ governing the Poisson distribution, in terms of your touchdown counts $x_1, \dots, x_6$. (Hint: remember that the log of the likelihood has the same maximum as the likelihood function itself.)

b. *[5 points]* Given the touchdown counts, what is your numerical estimate of λ?

2. *[10 points]* In World War 2 the Allies attempted to estimate the total number of tanks the Germans had manufacturered by looking at the serial numbers of the German tanks they had destroyed. The idea was that if there were n total tanks with serial numbers $\{1, \dots, n\}$ then its resonable to expect the observed serial numbers of the destroyed tanks consitututed a uniform random sample (without replacement) from this set. The exact maximum likelihood estimator for this so-called *German tank problem* is non-trivial and quite challenging to work out (try it!). For our homework, we will consider a much easier problem with a similar flavor.
Let $x_1, \dots, x_n$ be independent, uniformly distributed on the continuous domain $[0, \theta]$ for some θ. What is the Maximum likelihood estimate for θ?

Overfitting

3. Suppose we obtain N labeled samples $\{(x_i, y_i)\}_{i=1}^N$ from our underlying distribution $\mathcal{D}$. 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:

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

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

a. *[3 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. *[4 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 $\epsilon(\widehat{f})$? If so, why? If not, give a clear argument as to where your previous argument breaks down.

c. *[8 points]* Let $\mathcal{F} = (f_1, f_2, \dots)$ be a collection of functions and let $\widehat{f}_{\text{train}}$ minimize the training error such that $\widehat{\epsilon}_{\text{train}}(\widehat{f}_{\text{train}}) \le \widehat{\epsilon}_{\text{train}}(f)$ for all $f \in \mathcal{F}$. Show that

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

(Hint: note that

$$\mathbb{E}_{\text{train,test}}[\widehat{\epsilon}_{\text{test}}(\widehat{f}_{\text{train}})] = \sum_{f\in\mathcal{F}} \mathbb{E}_{\text{train,test}}[\widehat{\epsilon}_{\text{test}}(f)\mathbf{1}\{\widehat{f}_{\text{train}} = f\}]$$
$$= \sum_{f\in\mathcal{F}} \mathbb{E}_{\text{test}}[\widehat{\epsilon}_{\text{test}}(f)]\mathbb{E}_{\text{train}}[\mathbf{1}\{\widehat{f}_{\text{train}} = f\}] = \sum_{f\in\mathcal{F}} \mathbb{E}_{\text{test}}[\widehat{\epsilon}_{\text{test}}(f)]\mathbb{P}_{\text{train}}(\widehat{f}_{\text{train}} = f)$$

where the second equality follows from the independence between the train and test set.)

Bias Variance tradeoff

4. For $i = 1, \dots, n$ 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. For some $m \le n$ such that n/m is an integer define the estimator

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

Note that this estimator just partitions $\{1, \dots, n\}$ into intervals $\{1, \dots, m\}, \{m+1, \dots, 2m\}, \dots, \{n-m+1, \dots, n\}$ and predicts the average of the observations within each interval (see Figure 1).
By the bias-variance decomposition at some x_i we have

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

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?

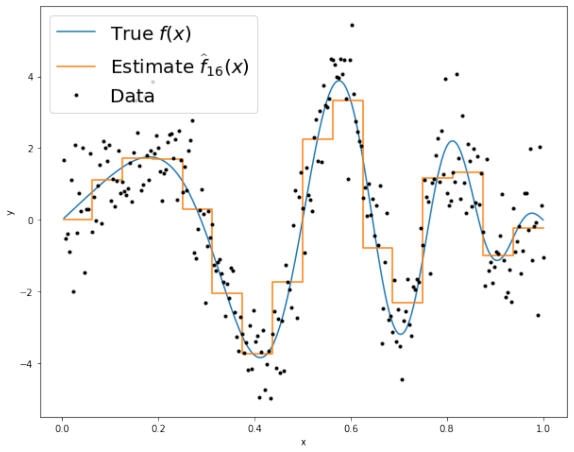

Figure 1: Step function estimator with $n = 256$, $m = 16$, and $\sigma^2 = 1$.

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

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

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)

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

d. *[15 points]* Let $n = 256$, $\sigma^2 = 1$, and $f(x) = 4\sin(\pi x)\cos(6\pi x^2)$. For values of $m = 1, 2, 4, 6, 8, 16, 32$ 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.

e. *(Extra credit* [5 points] *)* By the Mean-Value theorem we have that $\min_{i=(j-1)m+1,\dots,jm} f(x_i) \le \bar{f}^{(j)} \le \max_{i=(j-1)m+1,\dots,jm} f(x_i)$. Suppose f is L-Lipschitz so that $|f(x_i) - f(x_j)| \le \frac{L}{n}|i-j|$ for all $i, j \in \{1, \dots, n\}$ for some $L < 0$. Show that the average bias-squared is $O(\frac{L^2m^2}{n^2})$. Using the expression for average variance above, the total error behaves like $O(\frac{L^2m^2}{n^2} + \frac{1}{m})$. Minimize this expression with respect to m. Does this value of m, and the total error when you plug this value of m back in, behave in an intuitive way with respect to n, L, σ^2? It turns out that this simple estimator (with the optimized choice of m) obtains the best achievable error rate up to a universal constant in this setup for this class of L-Lipschitz functions (see Tsybakov's *Introduction to Nonparametric Estimation* for details).

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

5. *[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

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

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

6. 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

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

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

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 zth 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:

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

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 $\arg\max_{j=0,\ldots,9} e_j^T \widehat{W}^T x_i$. Note that if $W = \begin{bmatrix} w_1 & \ldots & w_k \end{bmatrix}$ then

$$\begin{aligned}\sum_{i=0}^{n} \|W^T x_i - y_i\|_2^2 + \lambda\|W\|_F^2 &= \sum_{j=0}^{k} \left[\sum_{i=1}^{n} (e_j^T W^T x_i - e_j^T y_i)^2 + \lambda \|W e_j\|^2 \right] \\ &= \sum_{j=0}^{k} \left[\sum_{i=1}^{n} (w_j^T x_i - e_j^T y_i)^2 + \lambda \|w_j\|^2 \right] \\ &= \sum_{j=0}^{k} \left[\|X w_j - Y e_j\|^2 + \lambda \|w_j\|^2 \right]\end{aligned}$$

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

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

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 $\arg\max_{j=0,\ldots,9} e_j^T W^T x_i'$ 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 the 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*

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

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.