CSE546 Machine Learning, Autumn 2016: Homework 2

Due: Friday, October 28^{th}, 5pm

Policies

Coding: You must write your own code. You may use any numerical linear algebra package, but you may *not* use machine learning libraries (e.g. sklearn, tensorflow) unless otherwise specified. In addition, each student must write and submit their own code in the programming part of the assignment (we may run your code).

Acknowledgments: We expect you to make an honest effort to solve the problems individually. As we sometimes reuse problem set questions from previous years, covered by papers and webpages, we expect the students not to copy, refer to, or look at the solutions in preparing their answers (referring to unauthorized material is considered a violation of the honor code). Similarly, we expect to not to google directly for answers (though you are free to google for knowledge about the topic). If you do happen to use other material, it must be acknowledged here, with a citation on the submitted solution.

Readings

Read the required material.

Required HW submission format:

The following is the required HW submission format:

- Submit all the answers to the HW as one single *typed* pdf document (not handwritten). This document must contain all plots. It is encouraged you latex all your work, though you may use another comparable typesetting method.
- Additionally, submit your code as a separate archive file (a .tar or .zip file). All code *must* be submitted. The code should be in a runnable file format like .py files or .txt files. Jupyter notebooks are not acceptable.
- List the names of all people you collaborated with and for which question(s). Please mark this as Question 0. Each student must understand, write, and hand in their own answers. Also, each student must understand, write, and hand in their own code.

0 Collaboration and Acknowledgements

List the names of all people you collaborated with and for which question(s). Each student must understand, write, and hand in their own answers. Also, each student must understand, write, and hand in their own code.

1 Programming Question: Multi-Class Classification using Least Squares [20 Points]

Obtain the MNist dataset from `http://yann.lecun.com/exdb/mnist/`.

In HW1, we built a binary classifier. Let us now build a multi-class classifier using regression.

1.1 One vs all classification

Let us do multi-class classification by solving 10 binary classification problems, which is sometimes referred to as "one against all". For the k-th binary classification problem, you will do a linear regression where you label a digit as $Y = 1$ if and only if the label for this digit is k (for $k = 0, 1, 2, \ldots 9$).

For classification, you will then take the largest predicted score among your 10 predictors.

1. *(2 points)* Naively, let us suppose you solve each classification problem separately, by doing k linear regressions (in d dimensions with n points). What is the total computational complexity of computing the solution for *one* of these linear regression problems, in terms of n and d? Take into to account the construction time of computing all matrices and vectors used in computing the (exact) least squares solution. (Assume that matrix inversion of a $d \times d$ matrix takes time d^3, eventhough theoretically faster runtimes are known). Note that if you solve all k problems in this manner, your run time will be a factor of k larger than the time it takes to solve one of these regression problems.
2. *(3 points)* Provide an improved runtime for solving all k problems jointly. You should improve upon the naive algorithm of solving k problems separately (i.e. you should be better than k times the complexity you found in the previous problem). Again, what is the total computational complexity of computing all the solutions, in terms of k, n, and d? Again, take into to account the construction time of computing all matrices and vectors used in computing the (exact) least squares solution.
3. *(7 points)* Run this multiclass algorithm on all 10 classes (with appropriate regularization if needed). On the training set, what is your 0/1 loss and what is your square loss? On the test set, what is your 0/1 loss and what is your square loss?

1.2 Neural Nets with a random first layer: Using more features [5 points]

Now let us try to make "better" features; we are not going to be particularly clever in the way we make these features, though they do provide non-trivial improvements. Let x be an image (as a vector in $\mathbb{R}^d$). Now we will map each x to a k-dimensional feature vector as follows:

$$x \to (h_1(x), h_2(x), \ldots h_k(x))$$

We will use $k = 10000$ here (k is ten thousand). Each h_j will be of the form:

$$h_j(x) = \max\{v_j \cdot x, 0\}$$

where $v_j \in R^d$. To construct all these vectors $v_1, v_2, \dots v_k$, you will independently sample every coordinate for every vector from a standard normal distribution. (We can think of this like a one-layer neural network where our first layer has random weights, and we are not changing them).

In particular, the square loss in this case is:

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \sum_{j=1}^{k} w_j h_j(x^{(i)})\right)^2$$

where the sum is over the N points in our training set, $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(N)}, y^{(N)})\}$, and where $h_j(x)$ is the j-th (random) feature function we have made. Use appropriate regularization.

Keep these vectors $v_1, \dots v_k$ around as you will use them for the next HW problem.

1. *(8 points)* Once you have these k-dimensional features, run our multiclass algorithm on all 10 classes (with appropriate regularization if needed). On the training set, what is your 0/1 loss and what is your square loss? On the test set, what is your 0/1 loss and what is your square loss?

2 Programming Question: Multi-Class Classification using Logistic Regression and Softmax [50 Points]

We now turn to logistic regression and the softmax classifier for MNIST.

Throughout this question use appropriate regularization as needed.

2.1 Binary Logistic Regression [10 Points]

Let us again consider the classification problem of recognizing if a digit is a 2 or not, except now let us use logistic regression. Here, let $Y = 1$ for all the 2's digits in the training set, and use $Y = 0$ for all other digits.

Build a logistic classifier by minimizing the log loss. Use appropriate regularization as needed. Also please use an (unregularized) offset weight w_0. For optimization, use batch gradient descent. Run until you feel you are near to convergence.

1. *(2 points)* What learning rate did you use?
2. *(4 points)* For both the training set and the test, plot the log loss as function of the iteration number (and show both curves on the same plot). Note that you are only optimizing on the training set.
3. *(4 points)* For the 0/1 use a threshold of 0.5 for classifying if the digit is a 2 (based on the training set). On the training set and test set, what are your final log losses and your 0/1 losses?

2.2 Softmax classification: gradient descent [15 Points]

Let us do multi-class classification. Rather than doing "one against all" you will use the soft-max classifier. Here, Y takes values in the set $\{0, \dots k-1\}$. The model is as follows: we have $k-1$, weight vectors $w^{(1)}, w^{(2)}, \dots w^{(k-1)}$. For $Y > 0$

$$\Pr(Y = \ell | x, w) = \frac{\exp(w^{(\ell)} \cdot x)}{1 + \sum_{i=1}^{k-1} \exp(w^{(i)} \cdot x)}$$

and for $Y = 0$

$$\Pr(Y = \ell | x, w) = \frac{1}{1 + \sum_{i=1}^{k-1} \exp(w^{(i)} \cdot x)} .$$

Note that the probabilities sum to 1.

For classification, you will then take the largest predicted score among your 10 predictors. Let us optimize the log loss with batch gradient descent.

The (negative) likelihood function on an N sampling training set is:

$$L(w) = \frac{-1}{N} \sum_{i=1}^{N} \log \Pr(Y = y^{(i)} | x^{(i)}, w)$$

where the sum is over the N points in our training set.

For classification, one can choose the class with the largest predicted probability.

1. *(4 points)* Write out the derivative of the log-likelihood of the soft-max function. Write this in a natural manner only interms of conditional probabilities, the vectors X, and indicator functions. (Do not write this expression in terms of exponentials). You will use this gradient in subsequent parts of this problem.
2. *(1 points)* Implement batch gradient descent. What learning rate did you use?
3. *(4 points)* For both the training set and the test, plot the log loss as function of the iteration number (and show both curves on the same plot). Do this for the 0/1 loss as well.
4. *(1 points)* On the training set and test set, what are your final log losses? your 0/1 losses?

2.3 Softmax classification: stochastic gradient descent [15 Points]

Let us now optimize with stochastic gradient descent (SGD). Let use both SGD with one point at a time, and let use SGD using a mini-batch size of 100. You may turn down the learning rate if you desire (either by hand or according to some decay scheme) or you may choose a single learning rate. You should get a final predictor comparable to that in the previous question.

1. *(4 points)* Implement SGD, with mini-batch size of 1. After every 15,000 points you have passed through, for both the training set and the test, plot the log loss as function of the iteration number (and show both curves on the same plot). Do this for the 0/1 loss on the training and test set as well (showing both curves on the same plot). These plots should start at iteration 0. Your curves should eventually drop to the performance comparable to that of batch gradient descent. How many iterations did this take you? Specify your learning rate? (or specify your learning rate decay scheme if you alter the learning rate).
2. *(1 points)* Compare (to batch gradient descent) the total computational complexity it took you to reach a comparable 0/1-loss on your training set. Note that each iteration of batch gradient descent costs a factor of n more (where n is the number of data points).
3. *(4 points)* Implement SGD, with mini-batch size of 100. In other words, rather than estimating the gradient with one random sample, use 100 samples. Make identical plots of the log loss and the 0/1 loss as before. Your curves should eventually drop to the performance comparable to that of batch gradient descent. How many iterations did this take you? Specify your learning rate? (or your learning rate scheme if you adapted it).
4. *(1 points)* Compare the mini-batch algorithm to that of just using one point at a time in terms of total computational time.

5. *(0 points)* Aside: in many practical cases, computing a mini-batch gradient estimate with 100 points can be done efficiently with matrix operations (i.e. it is not 100 slower than computing just one stochastic gradient with batch size 1).

2.4 Neural Nets with a random first layer: Using more features [10 points]

Let us now use the features we made in Section 1.2.

1. *(5 points)* Run your favorite algorithm from the previous experiments (for the softmax classifier) on the our k-dimensional feature vector (with appropriate regularization if needed). Show reasonable plots for how your log-loss and 0/1 loss evolve on both your training and test errors (as you did above). Specify what algorithm you used and your learning rate scheme.
2. *(5 points)* On the training set, what is your 0/1 loss and what is your log loss? On the test set, what is your 0/1 loss and what is your log loss?

3 (Baby) Learning Theory: Why is statistical learning possible? [22 points]

Suppose now that you have a finite set of K classifiers. This set of classifiers is sometimes referred to as the *hypothesis space*. We will now ask the question of how many samples do you need so that your classification error rate is close to that of the best classifier among this set.

More broadly, as we get more data, we would expect that we can utilize a richer hypothesis class. Specifically, we expect that with more data we can fit a more complex model. In this question, we seeks to quantify this relationship.

3.1 A confidence interval for a coin [2 points]

A simplified version of the Chernoff-Hoeffding bound is as follows: Suppose that $Z_1, Z_2, \dots Z_N$ are i.i.d. Bernoulli(θ) binary random variables. In other words, $Pr(Z = 1) = \theta$ and $Pr(Z = 0) = 1 - \theta$. In this setting, it is common to use the terminology that θ is the bias of the coin. Let $\overline{Z}$ be the empirical average, i.e. $\overline{Z} = \frac{1}{N}\sum_{i=1}^{N} Z_i$. Then for any $\epsilon > 0$,

$$\Pr(|\overline{Z} - \theta| \geq \epsilon) \leq 2\exp(-2N\epsilon^2)$$

1. *(2 points)* Provide a confidence interval for $\overline{Z}$ that holds with probability greater than $1 - \delta$.

3.2 Estimating the performance of a given classifier [2 points]

Suppose we have a distribution over pairs (X, Y) where Y is a binary random variable in $\{0, 1\}$. Here, X lives in an arbitrary set $\mathcal{X}$ (X need not be a vector). The 0/1 loss for a classifier $f : \mathcal{X} \to \{0, 1\}$ is defined as:

$$L(f) = \mathbb{E}_{X,Y}[\mathbf{1}(f(X) \neq Y)]$$

where $\mathbf{1}(\mathcal{E})$ is the indicator function for the event $\mathcal{E}$ (the function takes the value 1 if $\mathcal{E}$ occurs and 0 otherwise). The expectation is with respect to the underlying distribution on (X, Y).

1. *(2 points)* Suppose you have a given classifier f and you seek to estimate the loss of f. You obtain a set of samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Define an estimator, $\widehat{L}(f)$ for the loss of this classifier and also provide a confidence interval for this loss, which holds with probability greater than $1-\delta$. In other words, we are seeking a bound on $|L(f) - \widehat{L}(f)|$ that holds with probability greater than $1-\delta$.

3.3 ERM revisited [20 points]

Let us turn to the standard supervised learning setting. We are provided with an N-sample training set $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, sampled i.i.d. from this underlying distribution. Using this training set, we are interested in finding a classifier, $\widehat{f}$, which minimizes the 0/1 loss. Importantly, note that the function $\widehat{f}$ depends on $\mathcal{T}$ (where $\mathcal{T}$ is randomly sampled).

Suppose we have a finte set of classifiers $\mathcal{F} = \{f_1, f_2, \ldots f_K\}$ (e.g. the classifiers based on halfspaces, the class of neural network classifiers, the class of decision trees,...). We often refer to $\mathcal{F}$ as the hypothesis space. Also, suppose the classifier we choose minimizes the loss on the training set, i.e. it is the *empirical risk minimizer* (ERM). Specifically, our classifier is defined as follows:

$$\widehat{f} = \arg\min_{f\in\mathcal{F}} \widehat{L}(f)$$

where

$$\widehat{L}(f) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}(f(x_i) \neq y_i)$$

(i.e. $\widehat{L}(f)$ is an empirical estimate of our loss).

The "best in class" function f^* is:

$$f^* = \arg\min_{f\in\mathcal{F}} L(f)\,.$$

Note that $L(f^*)$ is the best loss we could hope to achieve using functions in $\mathcal{F}$.

1. *(1 points)* Is it the case that previous confidence interval (from the last question), when applied to the loss of f^*, holds with probability greater than $1-\delta$. Why or why not? (Note that this question is not asking wether or not you know the quantities in the purported confidence interval, but it is asking if the confidence interval when applied to f^* leads to a valid probabilistic statement).

2. *(1 points)* Is it the case that previous confidence interval (from the last question), when applied to the loss of $\widehat{f}$, holds with probability greater than $1-\delta$. Why or why not?

3. *(4 points)* Provide a confidence interval that simultaneously holds for the losses of all $f \in \mathcal{F}$, with probability of error δ. In other words, provide a value B so that:

$$\Pr(\text{for all } f \in \mathcal{F},\ |\widehat{L}(f) - L(f)| \leq B) \geq 1-\delta$$

   Show your steps. Hint: for events $\mathcal{E}_1, \mathcal{E}_2, \ldots \mathcal{E}_m$, the "union bound" states that $\Pr(\mathcal{E}_1 \text{ or } \mathcal{E}_2 \text{ or } \mathcal{E}_m) \leq \sum_{i=1}^{k} \Pr(\mathcal{E}_k)$. (This bound is also sometimes referred to as the Bonferoni bound).

4. *(3 points)* Provide a confidence interval for the loss of $\widehat{f}$ that holds with probability greater than $1-\delta$. In other words, we are seeking a bound on $|L(\widehat{f}) - \widehat{L}(\widehat{f})|$ that holds with probability greater than $1-\delta$.

5. *(4 points)* Provide a bound on how close your loss, using the ERM, is to the best possible loss (in the hypothesis space). Specifically, provide a value R so that the following holds with probability greater than $1-\delta$,

$$L(\widehat{f}) - L(f^*) \leq R$$

   The quantity $L(\widehat{f}) - L(f^*)$ is often referred to as the *regret*, as it is how close you are to the best possible. Here, R is our regret bound.

6. *(5 points)* Let us understand how large a hypothesis class we can utilize, i.e. how large K can be? Suppose we know we will be provided with a training set of size N, and, before we look at our training data, we choose the a hypothesis class $\mathcal{F}$ as a function of the sample size N. As we get more data, we would expect that we can utilize a larger hypothesis class. Let us examine this more quantitatively. We can think of learning being possible if our regret tends to 0 as N becomes large (with a probability of error less than $1-\delta$). Let us determine if learning is possible in each of the following cases.

   (a) *(1 points)* Suppose K is a constant. Are we able to learn, and, if so, what is our regret as function of N in order notation?

   (b) *(1 points)* Let's consider an even large hypothesis class. Suppose $K = N^p$ for some constant p. Are we able to learn, and, if so, what is our regret as function of N in order notation?

   (c) *(1 points)* Let's consider being more greedy in our choice of a hypothesis class. Suppose $K = \exp(\sqrt{N})$. Are we able to learn, and, if so, what is our regret as function of N in order notation?

   (d) *(2 points)* Let's consider being even more greedy in our choice of a hypothesis class. Suppose $K = \exp(10N)$. Are we able to learn, and, if so, what is our regret as function of N in order notation?

   Your answers to the above are one interpretation as to why learning complex models is possible (from a statistical perspective).

3.4 Aside: passing from the finite to the infinite... [0 points]

We have shown that we can learn among finite hypothesis classes that are very large (as a function of N). In practice, we often use infinite hypothesis classes (e.g. the class of halfspaces, decision trees, neural networks, etc). Why we can learn even when our hypothesis class is infinite? Our previous arguments can help to provide intuition.

First, observe that our above bounds could be extremely loose. Suppose that $\mathcal{F}$ has K functions which are all identical. Our bounds are flagrantly loose in this case. Can you spot which step in our argument is sloppy? We really only have $K = 1$. More generally, even in cases when $\mathcal{F}$ is infinite, it could be the case that many functions in $\mathcal{F}$ are near to being identical, and have nearly identical loss (e.g. similar weight vectors or parameters tend to provide to classifiers with similar losses). When going from finite $\mathcal{F}$ to infinite $\mathcal{F}$, much of learning theory seeks to find a way to precisely characterize the *effective* size of $\mathcal{F}$.

Very crudely, for linear models, we can think of the effective size of $\mathcal{F}$ as being exponential in the dimension d (and so the sample size needed to learn is $O(d)$). If we regularize, this brings down the effective size of $\mathcal{F}$. More generally, we can think of the effective size of $\mathcal{F}$ as being exponential in the VC-dimension of $\mathcal{F}$. One of the most general and powerful techniques is to provide a fine grained "cover" for $\mathcal{F}$, where this cover is: (i) finite and (ii) all $f \in \mathcal{F}$ are near to some function in our cover. We then argue learnability is possible by arguing that if we can learn on our fine grained (finite) cover then we can learn on the entire infinite class $\mathcal{F}$.

4 SVMs: Hinge loss and mistake bounds [8 Points]

Suppose we classify with the hypothesis $h_w(x) = \text{sgn}(w^\top x)$ where $\text{sgn}(z) = 1$ if z is positive and -1 otherwise. Here we are dealing with binary prediction where each label is in $\{-1, 1\}$. Let us say we make a mistake on x, y with w if $y \neq \text{sgn}(w^\top x)$. The number of mistakes with w on a dataset is the classification loss.

The SVM loss function can be viewed as a relaxation to the classification loss. The *hinge* loss on a pair (x, y) is defined as:

$$\ell((x,y), w) = \max\{0, 1 - yw^\top x\}$$

where $x \in \mathrm{R}^d$ and $y \in \{-1, 1\}$. The SVM attempts to minimize:

$$\frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i w^\top x_i\} + \lambda\|w\|^2$$

where λ is a regularizer (there are different conventions as to how we write the regularizer, e.g. sometimes we write $\frac{\lambda}{n}$ instead of just λ).

The hinge loss provides a way of dealing with datasets that are not separable. Let us go through the argument as to why we view this as a relaxation of finding the max margin classifier.

1. *(2 Points)* Argue that the function $\ell((x, y), w) = \max\{0, 1 - yw^\top x\}$ is convex (as a function of w).
2. *(3 Points)* (Margin mistakes) Suppose that for some w we have a correct prediction of y_i with x_i, i.e. $y_i = \mathrm{sgn}(w^\top x_i)$. What range of values can the hinge loss, $\ell((x_i, y_i), w)$, take on this correctly classified example? Points which are classified correctly and which have non-zero hinge loss are referred to as margin mistakes.
3. *(3 Points)* (Mistake bound) Let $M(w)$ be the number of mistakes made by w on our dataset (in terms of the classification loss). Show that:

$$\frac{1}{n}M(w) \le \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i w^\top x_i\}$$

   In other words, the average hinge loss on our dataset is an upper bound on the average number of mistakes we make on our dataset.