Homework 2: Extra Credit
CSE 446: Machine Learning

University of Washington

1 Policies [0 points]

Please read these policies. **Please answer the three questions below and include your answers marked in a "problem 0" in your solution set.** Homeworks which do not include these answers will not be graded.

Extra Credit Policy: In order to get extra credit, your HW2 must be submitted by the regular due date, without using late days. Furthermore, extra credit points will only be awarded if there are (honest attempts at) answers to *all* the regular questions in the HW. This is because the extra credit is not designed to be alternative questions to the regular questions.

Gradescope submission: When submitting your HW, please tag your pages correctly as is requested in gradescope. Untagged homeworks will not be graded, until the tagging is fixed.

Submission format: Submit your report as a *single* pdf file. Also, please include all your code in the PDF file in a section at the end of your document, marked "Code"; also specify which problem(s) the code corresponds to. The report (in a single pdf file) must include all the plots and explanations for programming questions (if required). Homework solutions must be organized in order, with all plots arranged in the correct location in your submitted solutions. We highly recommend typesetting your scientific writing using LaTeX(see the website for references for free tools). Writing solutions by hand will be accepted provided they are neat; written solutions need to be scanned and included into a single pdf.

Written work: Please provide succinct answers *along with succinct reasoning for all your answers*. Points may be deducted if long answers demonstrate a lack of clarity. Similarly, when discussing the experimental results, concisely create tables and figures to organize the experimental results. In other words, all your explanations, tables, and figures for any particular part of a question must be grouped together.

Including your Python source code: Updated policy for Jupyter. For the programming assignments, submit your code in the pdf file along with a neatly written README file that instructs us how you ran your code with different settings (if applicable). Please note that we will not accept screenshots of Jupyter notebooks. If you do use Jupyter, you must export your code to a text file and put the text of your code in the submitted pdf file (in the last section) in a manner that can be executed in that order (without any extraneous or missing code).

We assume that you always follow good practice of coding (commenting, structuring); these factors are not central to your grade.

Coding policies: You must write your own code. You are welcome to use any Python libraries for data munging, visualization, and numerical linear algebra. Examples includes Numpy, Pandas, and Matplotlib. You may **not**, however, use any machine learning libraries such as Scikit-Learn, TensorFlow, or PyTorch, unless explicitly specified for that question. If in doubt, post to the message boards.

Collaboration: It is acceptable for you to discuss problems with other students; it is not acceptable for students to look at another students written answers. It is acceptable for you to discuss coding questions with others; it is not acceptable for students to look at another students code. Each student must understand, write, and hand in their own answers. In addition, each student must write and submit their own code in the programming part of the assignment.

Acknowledgments: We expect the students not to refer to or seek out solutions in published material from previous years, on the web, or from other textbooks. Students are certainly encouraged to read extra material for a deeper understanding.

1.1 List of Collaborators

List the names of all people you have collaborated with and for which question(s).

1.2 List of Acknowledgements

If you do inadvertently find an assignment's answer, acknowledge for which question and provide an appropriate citation (there is no penalty, provided you include the acknowledgement). If not, then write "none".

1.3 Certify that you have read the instructions

Please make sure to read and follow these instructions. Write "I have read and understood these policies" to certify this.

2 Making "polynomial features" with PCA! *[35 points]*

In order to get credit on this question, you must do the coding the parts of the question.

This question requires the "mnist_2_vs_9.gz" dataset.

Now let us "blow up" the dimensionality by including all the quadratic interactions along with the linear terms. Let us understand this by example. Suppose x is three dimensional, i.e. $x = (x[1], x[2], x[3])$. Define a new feature vector as follows:

$$\Phi(x) = (1, x[1], x[2], x[3], x[1]^2, x[2]^2, x[3]^2, x[1]x[2], x[1]x[3], x[2]x[3]) .$$

The first term is the bias term, the next three coordinates above are considered the "linear" terms, and the remaining terms are the quadratic terms.

For this question you should use the closed form least squares estimator (with appropriate regularization).

1. ***[4 points]*** What is the dimensionality of $\Phi(x)$ for a d-dimensional input x?
2. ***[3 points]*** If we did this on our images (of dimensionality $d = 784$), what is the dimensionality of this feature vector? Why might using this with linear regression be a poor idea?
3. ***[2 points]*** Now let us consider building the feature vector $\Phi(\text{Proj}_K(x))$, where $\text{Proj}_K(x)$ is the projection of x onto the top K PCA subspace. Why might this be a better idea?
4. ***[1 point]*** You may use your PCA projections from Q2 or you are welcome to relearn the projections, so that they are specific to the training data when restricted to just these two classes. Please specify what you used. You may get different (possibly better) results by relearning the PCA on just these two classes. It does not matter what method you used, provided that you report what you did.
5. ***[10 points]*** Try out using linear regression with $\Phi(\text{Proj}_K(x))$, with $K = 10$. Report your average squared training error, your dev error, and your test error. Regularize appropriately if need be (a little search on the dev set should be sufficient to find a good regularizer). Also report the corresponding misclassification errors. What is the percent reduction (or increase) in your misclassification error rate in comparison to Q3?
6. ***[15 points]*** Try out using linear regression with $\Phi(\text{Proj}_K(x))$, with $K = 40$. Report your average squared training error, your dev error, and your test error. Regularize appropriately if need be (a little search on the dev set should be sufficient to find a good regularizer). Also report the corresponding misclassification errors. What is the percent reduction in your misclassification error rate in comparison to Q3?

Remark: You can see where this is going.... We could certainly try cubic features, etc. If you start on this path, regularization becomes important as does having fast algorithms.

3 (Baby) Learning Theory: Model Complexity [35 points]

The set of classifiers we choose from is sometimes referred to as the *hypothesis space*. Suppose now that we have a finite set of K classifiers $\{f_1(\cdot), f_2(\cdot), \ldots f_K\}$. . We will now ask the question

of how many samples do we need so that our 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.

In a sense, this question captures the heart of how to think more precisely about overfitting and model complexity.

3.1 A more precise confidence interval *[4 points]*

In HW1, we utilized the central limit to provide a confidence (which we believe to be accurate for "large" N). The Chernoff-Hoeffding bound is a means to just get a correct confidence interval (that holds for all N).

Suppose that $Z_1, Z_2, \ldots Z_N$ are independent, identically distributed (i.i.d.) random variables that are guaranteed to lie in the interval $[0, B]$. Let μ be the true mean, i.e. $\mu = \mathbb{E}[Z_i]$, and let $\widehat{\mu}$ be our empirical average, i.e. $\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} Z_i$. The Chernoff-Hoeffding bound is stated as follows: for any $\epsilon > 0$, we have:

$$\Pr(|\widehat{\mu} - \mu| \geq \epsilon) \leq 2\exp\left(-\frac{2N\epsilon^2}{B^2}\right)$$

1. ***[4 points]*** Show that with probability greater than $1 - \delta$, we have:

$$|\widehat{\mu} - \mu| \leq B\sqrt{\frac{\log(2/\delta)}{2N}}\,.$$

3.2 A confidence interval for a fixed 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:

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

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. ***[4 points]*** Suppose we have a given classifier f and we seek to estimate the loss of f. We then obtain a set of samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$. The empirical loss is:

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

   Provide a valid confidence interval on our estimate of$\widehat{\epsilon}(f)$ of the true loss. In other words, we are seeking a bound on

$$|\epsilon(f) - \widehat{\epsilon}(f)| \leq ??$$

   that holds with probability greater than $1 - \delta$. Justify your steps.

(You may be interested in thinking about how this statement compares to our test set confidence internval in HW1)

3.3 "Empirical risk minimization" *[24 points]*

Let us turn to the standard supervised learning setting. We are provided with an N-sample training set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, 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 our randomly sampled training set.

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). Precisely, our classifier is defined as follows:

$$\widehat{f} = \operatorname*{argmin}_{f \in \mathcal{F}} \widehat{\epsilon}(f) \,.$$

The "best in class" function f^* on the true loss is:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \epsilon(f) \,.$$

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

1. ***[3 points]*** Is it the case our previous confidence interval (from the last question, in Section 3.2), when applied to the estimated loss of f^*, holds with probability greater than $1 - \delta$. Why or why not?
2. ***[1 point]*** Is it the case that our previous confidence interval (from the last question, in Section 3.2), when applied to the estimated loss of $\widehat{f}$, holds with probability greater than $1 - \delta$. Why or why not?
3. ***[8 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 Δ so that:

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

   Make your argument precise. Note the subtlety, we are simultaneously seeking accuracy on all *all* $f \in \mathcal{F}$ as opposed to just one of them.
   (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. ***[4 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

   $$|\epsilon(\widehat{f}) - \widehat{\epsilon}(\widehat{f})| \leq ??$$

   that holds with probability greater than $1 - \delta$. Justify your steps.

5. ***[8 points]*** Provide a bound on how close your loss, using the ERM, is to the best possible loss (in the hypothesis space). Specifically, show that

$$\epsilon(\widehat{f}) - \epsilon(f^*) \leq \sqrt{\frac{2\log(2K/\delta)}{N}}$$

that holds with probability greater than $1-\delta$. Make your argument precise.
The quantity $\epsilon(\widehat{f}) - \epsilon(f^*)$ is often referred to as the *regret*, as it is how close our true loss is to the minimal possible true loss (among our class of hypotheses). This bound is sometimes referred to as the "*Occam's razor bound*".

3.4 Model Complexity and the "big data" regime *[5 points]*

As we get more data, we typically try more complicated learning algorithms and richer hypothesis classes. In this setting, we think about choosing K as a function of N. We might hope that even if we choose larger K with larger N, then we could still do well.

1. Let us say that learning occurs if our regret goes to 0 for sufficiently large N (say with some constant probability), i.e. $\epsilon(\widehat{f}) - \epsilon(f^*) \to 0$ as $N \to \infty$. Let us determine if learning is possible in each of the following cases. (A brief explanation is fine)
   (a) ***[1 points]*** Suppose K is a constant (i.e. we do not make K larger with N). Are we able to learn? Why or why not?
   (b) ***[2 points]*** Let's consider an even larger hypothesis class. Let us suppose we choose $K = N^p$ for some constant $p \geq 1$. Note that some thing like N^5 could be quite a large hypothesis space. Are we able to learn? Why or why not?
   (c) ***[2 points]*** Let's be even more greedy in our choice of a hypothesis class. Suppose $K = 2^N$, are we able to learn? Why or why not?

   Your answers to the above are one interpretation as to why learning complex models is possible.

3.5 Model complexity of linear classifiers and neural nets. *[0 points]*

We have just shown that we can learn with a finite hypothesis class. What about the case when $K = \infty$ such as is the case of linear classifiers, neural nets, or just about anything we do. It seems our previous bounds will not apply. It turns out that there is a way to use the bounds above to (at least intuitively) understand the case even when $K = \infty$.

First, let us get a little intuition as to why the above bounds could be very loose. Suppose that our hypothesis space $\mathcal{F}$ consists of exactly one function repeated K times, i.e. $f_i = f_1$ for all i. Clearly, using K in the bound is entirely artificial, as we really have just one function. The source of 'slop' in the argument is that our use of the union bound introduced an artificial factor of K.

This suggests the solution. Basically, we need to make some kind of argument about how many 'effective' classifiers we have. For example, with a linear classifier, two lines very 'near' to each other will behave nearly identically. One aspect of machine learning theory is in making this

intuition precise, and this leads to notions like the "covering number" or the "VC dimension" of our hypothesis space (which are a means to measure the "effective" size).

For linear classifiers in dimension d, we can think of $K_{\text{effective}} \approx 2^{\text{d}}$ and so we can non-trivially learn with about $O(d)$ samples.

4 Bayes Optimal Prediction: Regression *[8 points]*

Suppose now that y is real valued. The square loss of a hypothesis f, where f maps an input to a real value, is defined as:

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

Prove that the Bayes optimal hypothesis for the square loss (i.e. the hypothesis that achieves the minimal possible square loss among all possible hypothesis) is:

$$f^{(\text{BO})}(x) = \mathbb{E}[y|x]$$

(where the conditional expectation is with respect to $\mathcal{D}$).

5 Code

Please include all your code in the PDF file in this section. Specify which problem(s) the code corresponds to. Re Jupyter: refer to the policies section of the HW.