Georgia Institute of Technology

ECE 6254 - Assignment 2

Fall 2020 - v1.1

- There are 2 problems over 3 pages (including the cover page)
- The problems are not necessarily in order of difficulty.
- All problems are assigned the same weight in the overall grade.
- Each question is graded out of two points (0 for no meaningful work, 1 for partial work, 2 if correct)
- Unless otherwise specified, you should concisely indicate your reasoning and show all relevant work.
- The grade on each problem is based on our judgment of your level of understanding as reflected by what you have written. If we can't read it, we can't grade it.
- Please use a pen and not a pencil if you handwrite your solution.
- **You must submit your exam on Gradescope. Make sure you allocate time for the submission**.

Problem 1: Sample complexity for bounded loss function

In class, we proved that the PAC learnability of finite hypothesis classes with empirical risk minimization for the the 0-1 loss function. The goal is to prove that the result also holds for a bounded loss function $\ell(\hat{y}, y) \in [a; b]$ for some $a < b$ with sample complexity

$$\left\lceil \frac{2\ln(2|\mathcal{H}|/\delta)(b-a)^2}{\epsilon^2} \right\rceil.$$

Hint: This requires you to carefully go through the lecture notes, not to invent some completely new proof. However, you need to show that you understand the details of the steps involved and you must properly justify the various steps. Correct equations without justifications will not get full credit.

[Q1] Express concisely but precisely in your own words the meaning of PAC learnability.

[Q2] Use Hoeffding's inequality to develop a bound on $\mathbb{P}\left(\left|\widehat{R}_N(h_j) - R(h_j)\right| \geqslant \epsilon\right)$.

[Q3] Justify how to get a bound on $\mathbb{P}\left(\exists j \in [\![1, |\mathcal{H}|]\!] \text{ s.t. } \left|\widehat{R}_N(h_j) - R(h_j)\right| \geqslant \epsilon\right)$.

[Q4] Show how the above bound can be used to derive a bound on $\mathbb{P}\left(\left|R(h^*) - R(h^\sharp)\right| \geqslant 2\epsilon\right)$.

[Q5] Derive the expression of the sample complexity given above.

Problem 2: Learning tradeoff on MNIST dataset

The objective of this problem is to start using classifiers and visualize the learning tradeoff discussed during the lectures on a real world example. You don't need to fully understand how the classifiers operate to work on this problem, we will revisit them in more depth during lectures. Although there are not explicit points allocated to how cool your plots look, this will influence your grade to a small extent. *Even if not explicitly asked*, make sure that your plots are also supported by a (possibly brief) discussion commenting on what should be observed.

[Q1] Load the MNIST dataset (`fetch_openml('mnist_784',return_X_y=True)`) and select only the classes corresponding to 9 and 8. Show a few representative examples of the objects being classified. Briefly explain why this dataset could be somewhat challenging.

We will run a K nearest neighbor classifier on the dataset consisting only of "9" and "8". This can be obtained using the `neighbors` module in scikit. There is "richness" associated to the class of nearest neighbor classifiers, where the richness is *inversely proportional* to K. This is a bit different from the situation we analyzed, in which the richness is captured by the size of the hypothesis class.

Create a training set consisting of the first 3000 images and a testing set with the following 4000 images. Make sure that your sets are approximately balanced, in the sense that the classes contain roughly the same number of images.

[Q2] Run a nearest neighbor dataset with $K = 4$ and report the empirical risk (with a zero one loss function) as well as the risk on the test dataset. Identify a few improperly classified examples and comment.

[Q3] Repeat this experiment by varying the value of K between 1 and 10. Plot the empirical risk and the testing risk as a function of decreasing values of K. Comment on the learning tradeoff.

To visualize what's happening, we need to reduce the dimension of the input vectors. We will talk about

dimensionality reduction techniques later this semester, for now you'll have to accept that we'll use Isomap to reduce the dimension to two for visualization.

[Q4] Using Isomap and two components (function `manifold.Isomap` in `scikit-learn`), learn the transformation on the training dataset. Plot the transformed training dataset with different colors for each class.

[Q5] Create 4 nearest neighbor classifier to predict the label of the testing data. Plot the elements of the testing dataset that are *correctly* classified using the Isomap transformed learned earlier. Color code using the predicted labels. Explain why the classes are not perfectly separated although the predictions are correct.