6.867 Fall 2021: Homework 1

For the first homework, you will submit a single PDF containing your solutions to the following problems. There are 2 parts to the homework: a theoretical part and a hands-on part. **We impose a strict 8-page limit in total on your write-up**, to ensure that you focus on the important parts of your implementation and analysis. We make a few notes about the collaboration policy, submission procedure, grading policy and other information below.

Collaboration Policy

- **All write-ups must be done completely individually: they must be comprised of your own work, figures, and solutions.**
- You are free to discuss the problems with other students. However, you must acknowledge your collaborators as conflicts of interest on CMT.
- You should never use results from other students, the staff (from this year or from previous years), or other *online resources* in preparing your solutions to homework problems. In addition, students should never share their solutions (or staff solutions) with other students, including through public code repositories such as Github.

Submission: CMT & Dates

CMT is an online conference management software that you will use to submit and peer-grade your submissions. Instructions on how to use CMT should have been posted on Piazza. A few notes on deadlines.

- Register on the CMT site by 5:00 PM on Friday, September 24th.
- Submit your paper via the CMT site by 5:00 PM on Friday, October 1st.
- After submission on CMT, you will be assigned 3 papers to review.
- Solutions will be released on Canvas after 5:00 PM on Sunday, October 3rd.
- Reviews must be entered on the CMT site by 5:00 PM on Friday, October 8th.
- All reviews will be made visible shortly thereafter, and students will have a two day period in which to enter rebuttals of their reviews into CMT if they wish.

Procedure, grading, etc.

You will find a zip file with some useful code and data on the Canvas assignment page associated with this homework. You can do these assignments in any computational system that you are used to. We recommend Matlab or Python with the Pylab/Numpy/Scipy/Matplotlib cluster of packages and we'll try to provide help for those two systems. If you use anything else, you're on your own...

You will be turning in a single, readable "paper" (a single PDF file) with your solutions. **Please typeset your solutions and do NOT submit handwritten notes**. On Canvas we have provided a basic LaTeX template that you may use.

On the Canvas page we have provided some sample solutions from a previous year. The content in these examples is not directly relevant, but note that there are coherent explanations and nice illustrations of results in figures and tables. You should write your paper as if it is going to be read by someone who has taken a class in machine learning but has not read this assignment handout.

We will be emulating the process of submitting papers for publication to a conference. We will be using an actual conference review system (CMT) to have these papers peer reviewed (by the other students). This means that your answers have to be readable and understandable to your peers and, where possible, interesting. Note that when explanations are called for, you will need to convince the reviewers that you understand what you're talking about. The course staff will serve as the Program Committee for the conference and make all final decisions.

Grading rubric

Your paper must be anonymous (no identifying information should appear in the PDF file). If it is not, it will automatically receive a 20% deduction, and will be graded by a grumpy staff member.

The paper must be no more than 8 pages long in a font no smaller than 10 point. It should include whatever tables, graphs, plots, etc., are necessary to demonstrate your work and conclusions. **The paper itself should not include code.**

Each of the **three** parts of the assignment will be graded on a scale from 0 to 5 (where 0 is failing and 5 is an A) on two aspects:

- **Content:** Did the solution answer the questions posed? Were the answers correct? Were the experiments well-designed or examples well chosen?
- **Clarity:** Were the results written up clearly? Were the plots labeled appropriately and described well? Did the plots support the points in the paper? Did the discussion in the paper illuminate the plots?

As a reviewer, you will be asked to provide a score for each section, and at least two paragraphs of feedback, per review, explaining things that were done well and things that could have been improved upon.

Your overall score for this assignment will be:

- **80%**: The average of all scores on your assignment given by all three reviewers.
- **20%**: A score for the quality of your reviews. This will be full credit, by default. But we will skim reviews and examine some carefully and may reduce this grade for review commentary that is sloppy or wrong.

The course staff will spot-check submissions and reviews, paying careful attention to cases where there were rebuttals. The staff will act as the program committee and determine a final score. Our overall goals in this process are:

- To motivate you to work seriously on the problems and learn something about the machine learning material in the process
- To engage you in thinking critically and learning from other students' solutions to the problems

We will arrange to give full credit to anyone who submits a serious and careful solution to the problems and who gives evidence of having read carefully the solutions they were assigned and who writes thoughtful reviews of them.

The questions are the points that your paper should cover in order to receive full credit. Your presentation should roughly follow the order of these questions so that your reviewers can see what you're doing.

Introduction to Theoretical Section

The first part of the assignment consists of two theoretical problems that will allow you to apply your proof-oriented problem solving skills.

1 Generalization Error of the 1-NN Classifier

In this problem, we will investigate the generalization error of the k-NN binary classifier for $k = 1$. First, let's introduce the setting and some notation, most of which should be familiar from the lectures:

- Let $\mathcal{X} = [0,1]^d$ and $\mathcal{Y} = \{0,1\}$ denote the input and output domain spaces and d denote the dimension of the input space
- Let $\mathcal{D}$ be the distribution of samples over $\mathcal{X} \times \mathcal{Y}$, $\mathcal{D}_{\mathcal{X}}$ denote the marginal distribution over $\mathcal{X}$, and $\mathcal{D}^m$ denote $\underbrace{\mathcal{D} \times \mathcal{D} \times \cdots \times \mathcal{D}}_{m \text{ times}}$
- We use the 0-1 loss: for a classifier h and input-output pair (x, y), $\ell(h, (x, y)) = \mathbb{1}[h(x) \neq y]$. In other words, $\ell(h, (x, y)) = 1$ if $h(x) \neq y$, and $\ell(h, (x, y)) = 0$ otherwise.
- Let $\eta(x) = \Pr[y = 1|x]$ be the conditional distribution of y given an input x
- Let $\mathcal{L}_{\mathcal{D}}(h)$ be the generalization error of a hypothesis h: $\mathcal{L}_{\mathcal{D}}(h) \triangleq \mathbb{E}_{(x,y)\in\mathcal{D}}[\ell(h, (x, y))]$
- Let $\text{NN}_S(x)$ be the nearest neighbor of x in S, i.e. the $x_i \in S$ minimizing $\|x - x_i\|$, where $\|\cdot\|$ is the Euclidean norm. If there are multiple such points, choose one arbitrarily. Let h_S be the 1-NN classifier given a set of samples $S \sim \mathcal{D}^m$, i.e. to find $h_S(x)$, we first find the input-output pair $(x_i, y_i) \in S$ such that $\|x - x_i\|$ is minimized and take the corresponding y_i as the label. Again, if there are multiple such points, we choose one i arbitrarily.
- Let $h^\star(x)$ be the Bayes optimal classifier $h^\star(x) = \mathbb{1}[\eta(x) > \frac{1}{2}]$

- Let $[d] \triangleq \{1, 2, \cdots, d\}$

For this problem, we'll introduce one additional critical assumption, that η is c-Lipschitz for some $c > 0$. In other words, for all $x, x' \in \mathcal{X}$,

$$|\eta(x) - \eta(x')| \leq c\|x - x'\|$$

This assumption is like a continuity assumption, ensuring that our distribution is well behaved. It means that if two points x and x' are close together, then their expected label distributions are similar. With these definitions in place, the ultimate goal of this problem is to prove the following theorem:

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\mathcal{L}_{\mathcal{D}}(h_S)] \leq 2\mathcal{L}_{\mathcal{D}}(h^\star) + 2c\sqrt{d}m^{-\frac{1}{d+1}} \tag{1}$$

1. Before we prove the theorem, let's make sure we understand some of the underlying intuition behind what we are about to show!
   (a) Intuitively, explain why the c-Lipschitzness assumption is necessary for proving generalization guarantees of the k-NN classifier. In other words, what intuitively could go wrong if η is not Lipschitz? In addition, explain why you might expect worse generalization error bounds as c increases.
   (b) What does the result in Eq. 1 imply about the generalization error of the 1-NN classifier if we choose a larger and larger set S? Compare to the theorem shown in lecture 2, slide 17.

2. You'll first prove a combinatorial lemma that may seem irrelevant to proving Eq. 1. However, as you'll see in part 4, this lemma will be crucial in bounding the quantity $\mathbb{E}_{x\sim\mathcal{D}_{\mathcal{X}}, S\sim\mathcal{D}^m}[\|x - \text{NN}_S(x)\|]$, where $\|\cdot\|$ is the standard Euclidean norm. You will prove the following lemma:

   Let $C_1, C_2, \cdots, C_r$ be r arbitrary disjoint subsets of $\mathcal{X} = [0,1]^d$, and $S = \{(x_i, y_i)\}_{i=[m]}$, $(x_i, y_i) \overset{iid}{\sim} \mathcal{D}$ be a random sample of m points from $\mathcal{D}$. Let $P_i = \Pr_{x\sim\mathcal{D}_{\mathcal{X}}}[x \in C_i]$ be the probability that a randomly chosen x will land inside C_i (more intuitively, how much of the total probability space C_i covers). In this lemma, you will show that

   $$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}\left[\sum_{i:C_i\cap S=\emptyset} P_i\right] \leq \frac{r}{me} \tag{2}$$

   (a) Since there's a lot of notation here, before you try diving into the proof of the lemma, ensure that you take a moment to understand what the lemma is saying! Intuitively, explain what you are about to prove. Why does the bound increase if r increases and decrease if m increases?
   (b) First, show that

   $$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}\left[\sum_{i:C_i\cap S=\emptyset} P_i\right] \leq r \max_i P_i e^{-P_i m}$$

   Hint: For $a \in [0,1], n \geq 0$, we have that $(1-a)^n \leq e^{-an}$

(c) Conclude that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}\left[\sum_{i:C_i\cap S=\emptyset} P_i\right] \leq \frac{r}{me}$$

3. In this part, you will finally begin to derive an initial bound for the generalization error of the 1-NN classifier! By the end of this part, you will have proved that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\mathcal{L}_{\mathcal{D}}(h_S)] \leq 2\mathcal{L}_{\mathcal{D}}(h^\star) + c\cdot \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}},S\sim\mathcal{D}^m}[\|x-\text{NN}_S(x)\|] \tag{3}$$

As before, make sure you understand what Eq. 3 intuitively means before diving into the proof, though we won't ask you to write it up this time!

(a) Show that for any two points $x, x' \in \mathcal{X}$, we have

$$\Pr_{y\sim\eta(x),y'\sim\eta(x')}[y\neq y'] \leq 2\eta(x)(1-\eta(x)) + c\|x-x'\|$$

(b) Using the result in part 3(a), show that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\mathcal{L}_{\mathcal{D}}(h_S)] \leq 2\mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}}}[\eta(x)(1-\eta(x))] + c\cdot \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}},S\sim\mathcal{D}^m}[\|x-\text{NN}_S(x)\|]$$

(c) Derive an expression for the error of the Bayes optimal classifier $\mathcal{L}_{\mathcal{D}}(h^\star)$, and conclude that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\mathcal{L}_{\mathcal{D}}(h_S)] \leq 2\mathcal{L}_{\mathcal{D}}(h^\star) + c\cdot \mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}},S\sim\mathcal{D}^m}[\|x-\text{NN}_S(x)\|]$$

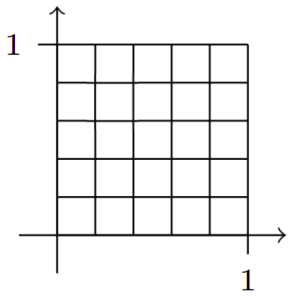
4. In this part, you will wrap everything up, refining the previous bound you proved in part 3(c) and leveraging the lemma you proved in part 2 by bounding the term $\mathbb{E}_{x\sim\mathcal{D}_{\mathcal{X}},S\sim\mathcal{D}^m}[\|x-\text{NN}_S(x)\|]$:

(a) Let $\epsilon = 1/T$ for some integer T, let $r = T^d$, and let $C_1,\cdots,C_r$ be the covering of $\mathcal{X} = [0,1]^d$ with d-dimensional hypercubes with side length ϵ.

More formally, for each d-tuple $(\alpha_1,\cdots,\alpha_d) \in [T]^d$, there exists a C_i that consists precisely of the points $x = (x_1,\cdots,x_d)$ such that $x_j \in [(\alpha_j-1)/T, \alpha_j/T]$ for all $j\in[d]$, i.e.:

$$C_i = \{x = (x_1,\cdots,x_d) : \forall j\in[d], x_j \in [(\alpha_j-1)/T, \alpha_j/T]\}$$

A sample covering $C_1,\cdots,C_{25}$ corresponding to $d=2, T=5, r=25$ is shown below:



Use part 2 to show that

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}},S\sim\mathcal{D}^m}[\|x-\text{NN}_S(x)\|]\leq\sqrt{d}\left(\frac{r}{me}+\epsilon\right)$$

Hint: Consider separately the two scenarios of whether or not x is in the same box as one of the m samples in S

(b) Next, choose ϵ appropriately to show that

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}_{\mathcal{X}},S\sim\mathcal{D}^m}[\|x-\text{NN}_S(x)\|]\leq 2\sqrt{d}m^{-\frac{1}{d+1}}$$

Finally, wrapping it all up, that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{D}^m}[\mathcal{L}_{\mathcal{D}}(h_S)]\leq 2\mathcal{L}_{\mathcal{D}}(h^\star)+2c\sqrt{d}m^{-\frac{1}{d+1}}$$

2 Stable Algorithms also Generalize

In this problem, we will investigate an intricate connection between the stability of an algorithm and its generalization abilities. First, some notation:

- Let $\mathcal{D}$ be a distribution, and $S=(z_1,\cdots,z_m)\overset{i.i.d.}{\sim}\mathcal{D}^m$ be a training set of m samples. In addition, let $S^{i,z}$ be the samples of S, but with sample i replaced with z; i.e., $(z_1,\cdots,z_{i-1},z,z_{i+1},\cdots,z_m)$
- For a learning algorithm A, we define $A(S)$ and $A(S^{i,z})$ to be the classifier resulting from running A on the samples S, $S^{i,z}$, respectively. We only consider *symmetric* learning algorithms A, meaning that if S and S' are permutations of each other, then $A(S)=A(S')$.
- As before, let $\ell(h,z)$ to be the loss of the classifier h on input z
- Let $\mathcal{L}_{\mathcal{D}}(h)$ be the generalization error of a hypothesis h: $\mathcal{L}_{\mathcal{D}}(h)\triangleq\mathbb{E}_{z\in\mathcal{D}}[\ell(h,z)]$
- Let $\mathcal{L}_S(h)$ be the empirical error of a hypothesis h evaluated on samples $S=(z_1,\cdots,z_m)$. Formally,

$$\mathcal{L}_S(h)=\frac{1}{m}\sum_{j=1}^{m}\ell(h,z_j)$$

  We also define $\mathcal{L}_{S^{i,z}}$ similarly as the empirical error evaluated on samples $S^{i,z}$.
- Let $[m]\triangleq\{1,2,\cdots,m\}$

In this problem, we say that an algorithm A is $\beta(m)$-stable, $\beta(m)>0$, if for all $S\in\mathcal{D}^m,z\in\mathcal{D},i\in[m]$, we have

$$\sup_{z'\in\mathcal{D}}|\ell(A(S),z')-\ell(A(S^{i,z}),z')|\leq\beta(m)$$

Intuitively, this means that even if any sample is replaced with any other sample, the overall loss of the classifier found by algorithm A can increase by at most $\beta(m)$.

Let A be a $\beta(m)$-stable algorithm and ℓ be a loss function bounded by M (i.e., that $|\ell(A(S), z)| \leq M$ for all $z \in \mathcal{D}$ and all training sets S). We will show, in a series of sub-problems, that for any $\epsilon > 0, m \geq 1$, we have

$$\Pr_{S \in \mathcal{D}^m}[|\mathcal{L}_{\mathcal{D}}(A(S)) - \mathcal{L}_S(A(S))| > \epsilon + \beta(m)] \leq 2\exp\left(-\frac{m\epsilon^2}{2(m\beta(m) + M)^2}\right) \tag{4}$$

1. As with the first question, make sure you understand what's going on here! Intuitively, explain your understanding of Eq. 4. Explain what happens to the RHS of Eq. 4 as $\beta(m)$ increases. What if M increases? ϵ? Does this match your intuition?

2. First, show that in expectation, the generalization gap $\mathcal{L}_D - \mathcal{L}_S$ is bounded by the stability, $\beta(m)$, i.e.: for all $i \in [m]$,

   $$\mathop{\mathbb{E}}_{S \sim \mathcal{D}^m}[\mathcal{L}_{\mathcal{D}}(A(S)) - \mathcal{L}_S(A(S))] = \mathop{\mathbb{E}}_{S \sim \mathcal{D}^m, z' \sim \mathcal{D}}[\ell(A(S), z') - \ell(A(S^{i,z'}), z')] \leq \beta(m)$$

   From this, we get a bound on the expected generalization gap. Next, we proceed to show a high-probability bound on the generalization gap. To do this, we will ultimately use the following inequality (which you may leverage without proof):

   (McDiarmid's inequality) Let $X = X_1 \cdots, X_m$ be m independent random variables taking values from some set A, and assume that $f : A^m \to \mathbb{R}$ satisfies the following boundedness condition:

   $$\sup_{x_1, \cdots, x_m, x_i'} |f(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_m) - f(x_1, \cdots, x_{i-1}, x_i', x_{i+1}, \cdots, x_m)| \leq c_i$$

   for all $i \in [m]$, where $x_i \in A$ is a realization of X_i. Then, for any $\epsilon > 0$, we have

   $$\Pr[|f(X_1, \cdots, X_m) - \mathbb{E}[f(X_1, \cdots, X_m)]| \geq \epsilon] \leq 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^m c_i^2}\right)$$

3. Show that for all $S \in \mathcal{D}^m$, $i \in [m]$, and $z' \in \mathcal{D}$,

   $$|\mathcal{L}_{\mathcal{D}}(A(S)) - \mathcal{L}_D(A(S^{i,z'})| \leq \beta(m)$$

4. Show that for all $S \in \mathcal{D}^m$, $i \in [m]$, and $z' \in \mathcal{D}$,

   $$|\mathcal{L}_S(A(S)) - \mathcal{L}_{S^{i,z'}}(A(S^{i,z'}))| \leq \frac{2M}{m} + \beta(m)$$

5. Finally, use McDiarmid's inequality on the quantity $\mathcal{L}_{\mathcal{D}}(A(S)) - \mathcal{L}_S(A(S))$ to show Eq. 4.

Introduction to Hands-On Section

The second part of this assignment deals with a hands-on implementation question.

In this problem, we provide a dataset to evaluate our classifiers on a set of synthetic (artificially generated) 2 dimensional (2D) data sets, numbered 1 through 4, designed to illustrate various points about

classification. Each of the four synthetic datasets is divided into train, validation, and test sets (available in the data folder). Make sure to test your implementations on all datasets. However, when not explicitly asked otherwise, feel free to include in the write-up only the results that you consider most relevant.

The problem below explore the effect of various forms of regularization on logistic regression. In addition to your final paper submission, you will be asked to upload supplementary material consisting of the code you utilized to answer the questions below. **We will check your code to ensure you do not copy existing online implementations.**

3 Logistic Regression

In this question, we will guide you to understand and empirically implement logistic regression, as well as the underlying impact to regularization on training performance. Let us consider two-class classification problems, where each sample $x_i \in \mathbb{R}^d$ is either labeled as 1 or 0. Recall that logistic regression attempts to find a *linear classifier* (i.e., a hyperplane), and it does so by modeling the conditional probability as:

$$\mathbb{P}(y = 1|x; w, w_0) = \sigma(w^T x + w_0), \tag{5}$$

where $\sigma(\cdot)$ represents the sigmoid function, $w \in \mathbb{R}^d$ and $w_0 \in \mathbb{R}$ represent the weights and bias (respectively), and $x \in \mathbb{R}^d$ represents an input. The underlying parameters of logistic regression may be found using maximum likelihood estimation. In particular, we maximize the expression

$$L(w) = \log \mathbb{P}(y|X; w) = \sum_{i=1}^{N} \frac{1}{N} \Big[y_i \log \sigma(w^T x_i + w_0) + (1 - y_i) \log(1 - \sigma(w^T x_i + w_0)) \Big]$$

where N represents the number of inputs and X represents the set of inputs $\{x_1, \cdots, x_N\}$.

1. Implement logistic regression on each of the 4 provided datasets using maximum likelihood estimation. You may find the function `scipy.optimize.minimize` helpful. Report the underlying training and testing accuracies of your algorithm and provide an explanation for the performance (you may find that visualizing the plotted decision boundaries on each dataset to be helpful) of logistic regression for each dataset. We provide some boilerpoint code in `logistic_test.py`

2. To remedy the issue that several of the provided datasets are not linearly separable, we may instead utilize a mapping function $\phi([x_0, x_1]) = [x_0, x_1, x_0^2, x_1^2, x_0 x_1, \sin(x_0), \sin(x_1), \cos(x_0), \cos(x_1)]$ which maps input features to a higher dimensional linearly seperable coordinate space. Report training and testing performance of logistic regression using features from this mapping function and explain the new underlying performance (you may find the plotted decision boundaries on each dataset to be helpful). Do you notice anything odd about decision boundaries? Attach an example of an odd decision boundary.

3. An issue with the previous approach is that in this higher dimensional space, it is significantly easier for the algorithm to overfit to the underlying training dataset. One way to mitigate this effect is to add L2 regularization weights (we suggest exploring values of $0.001, 0.01, 0.1$) to the logistic regression algorithm. How does this impact performance on each of the 4 provided datasets? Also, provide a comparison plot of a decision boundary with the addition of L2 regularization compared to no regularization on dataset 1.

4. An alternative regularization strategy is to add L1 regularization weights (we suggest exploring values of $0.001, 0.01, 0.1$) to the underlying training algorithm. How does this affect training and test performance on the 4 datasets? How does it affect the inferred weights w compared to L2 regularization? Compare the decision boundary obtained from a model fit with L1 regularization compared to L2 regularization on dataset 1.

5. We may utilize algorithmic stability as a way to measure the extent to which a given logistic regression algorithm may overfit to a training dataset. Measure the approximate algorithmic stability (defined as β in Problem 2) of logistic regression under separate L1 regularization weights (0.0, 0.001, 0.1) on dataset 4 and 1. Since it is computationally intractable (and impossible since we do not have access to the underlying probability distribution) to compute the overall algorithmic stability of your model, we recommend you approximate this value by comparing the NLL loss of a classifier trained on the entire dataset compared to M separate classifiers trained on M datasets with different missing datapoints (evaluated at the missing datapoint). Explain the differences in stability between the two datasets.