CS 189 Introduction to Machine Learning
Spring 2018 HW1

1 Getting Started

Read through this page carefully. You may typeset your homework in latex or submit neatly handwritten/scanned solutions. Please start each question on a new page. Deliverables:

1. Submit a PDF of your writeup, **with an appendix for your code**, to assignment on Gradescope, "HW1 Write-Up". If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.
2. If there is code, submit all code needed to reproduce your results, "HW1 Code".
3. If there is a test set, submit your test set evaluation results, "HW1 Test Set".

After you've submitted your homework, watch out for the self-grade form.

(a) Who else did you work with on this homework? In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

(b) Please copy the following statement and sign next to it. We just want to make it *extra* clear so that no one inadverdently cheats.

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

This homework is due **Monday, June 25 at 10 p.m.**

2 The accuracy of learning decision boundaries

This problem exercises your basic probability (e.g. from 70) in the context of understanding why lots of training data helps to improve the accuracy of learning things.

For each $\theta \in (1/3, 2/3)$, define $f_\theta : [0,1] \to \{0,1\}$, such that

$$f_\theta(x) = \begin{cases} 1 \text{ if } x > \theta \\ 0 \text{ otherwise.} \end{cases}$$

The function is plotted in Figure 1.

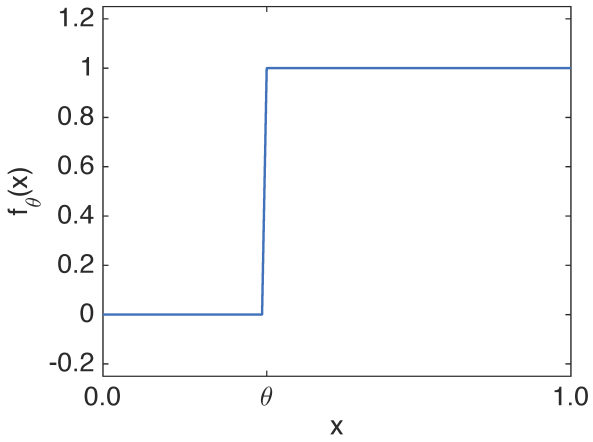

Figure 1: Plot of function $f_\theta(x)$ against x.

We draw samples $X_1, X_2, \ldots, X_n$ uniformly at random and i.i.d. from the interval $[0,1]$. Our goal is to learn an estimate for θ from n random samples $(X_1, f_\theta(X_1)), (X_2, f_\theta(X_2)), \ldots, (X_n, f_\theta(X_n))$.

Let $T_{min} = \max(\{\frac{1}{3}\} \cup \{X_i | f_\theta(X_i) = 0\})$. We know that the true θ must be larger than T_{min}.

Let $T_{max} = \min(\{\frac{2}{3}\} \cup \{X_i | f_\theta(X_i) = 1\})$. We know that the true θ must be smaller than T_{max}.

The gap between T_{min} and T_{max} represents the uncertainty we will have about the true θ given the training data that we have received.

(a) **What is the probability that $T_{max} - \theta > \epsilon$ as a function of ϵ? And what is the probability that $\theta - T_{min} > \epsilon$ as a function of ϵ?**

(b) Suppose that you would like the estimator $\hat{\theta} = (T_{max} + T_{min})/2$ for θ that is ϵ-close (defined as $|\hat{\theta} - \theta| < \epsilon$, where $\hat{\theta}$ is the estimation and θ is the true value) with probability at least $1 - \delta$. Both ϵ and δ are some small positive numbers. **Please bound or estimate how big of an n do you need?** You do not need to find the optimal lowest sample complexity n, an approximation using results of question (a) is fine.

(c) Let us say that instead of getting random samples $(X_i, f(X_i))$, we were allowed to choose where to sample the function, but you had to choose all the places you were going to sample in advance. **Propose a method to estimate θ. How many samples suffice to achieve an estimate that is ϵ-close as above?** (**Hint:** You need not use a randomized strategy.)

(d) Suppose that you could pick where to sample the function adaptively — choosing where to sample the function in response to what the answers were previously. **Propose a method to estimate θ. How many samples suffice to achieve an estimate that is ϵ-close as above?**

(e) In the three sampling approaches above: random, deterministic, and adaptive, **compare the scaling of n with ϵ (and δ as well for the random case).**

(f) **Why do you think we asked this series of questions? What are the implications of those results in a machine learning application?**

3 Eigenvalue and Eigenvector review

A square matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ has a (right) eigenvalue $\lambda \in \mathbb{C}$ and (right) eigenvector $\mathbf{x} \in \mathbb{C}^d \setminus 0$ if $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$. Left eigenvalues and eigenvectors are defined analogously — $\mathbf{x}^T\mathbf{A} = \lambda \mathbf{x}^T$. Since the definition is scale invariant (if $\mathbf{x}$ is an eigenvector, then $t\mathbf{x}$ is an eigenvector for any $t \neq 0$), we adopt the convention that each eigenvector has norm 1.

(a) **Compute the right and left eigenvalues and eigenvectors** of the following matrices. You may use a computer for questions v) and vi).

i) $\mathbf{A} = \begin{bmatrix} 2 & -4 \\ -1 & -1 \end{bmatrix}$

ii) $\mathbf{B} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$

iii) $\mathbf{A}^2$

iv) $\mathbf{B}^2$

v) $\mathbf{AB}$

vi) $\mathbf{BA}$

(b) **Compute the singular value decompositions** of the matrices above. In addition, please compute the SVD of: $\mathbf{C} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \\ 2 & -4 \\ -1 & -1 \end{bmatrix}$. For $\mathbf{B}$ and $\mathbf{B}^2$ compute by hand. For the rest of the matrices you may use a computer.

(c) **Show** from the definition of a right eigenvalue that the quantity λ is an eigenvalue with associated eigenvector $\mathbf{x}$ iff for all $1 \leq i \leq d$, we have

$$(\lambda - A_{ii})x_i = \sum_{j \neq i} A_{ij}x_j.$$

(d) Now for an arbitrary eigenvalue λ of $\mathbf{A}$ and its associated eigenvector $\mathbf{x}$, choose index i such that $|x_i| \geq |x_j|$ for all $j \neq i$. For such an index i, **show** that

$$|\lambda - A_{ii}| \leq \sum_{j \neq i} |A_{ij}|.$$

You have just proved Gershgorin's circle theorem, which states that all the eigenvalues of a $d \times d$ matrix lie within the union of d disks in the complex plane, where disk i has center A_{ii}, and radius $\sum_{j \neq i} |A_{ij}|$.

4 Fun with least squares

In ordinary least squares we learn to predict a *target* scalar $y \in \Re$ given a *feature* vector $\mathbf{x} \in \mathbb{R}^d$. Each element of $\mathbf{x}$ is called a feature, which could correspond to a scientific *measurement*. For example, the i-th element of $\mathbf{x}$, denoted by $(\mathbf{x})_i$, could correspond to the velocity of a car at time i. y could represent the final location (say just in one direction) of the car.

For the purpose of predicting y from $\mathbf{x}$ we are given n samples $(\mathbf{x}_i, y_i)$ with $i = 1, \ldots, n$ (where feature vectors and target scalars are observed in pairs), which we also call the training set. In this problem we want to predict the unobserved target y corresponding to a new $\mathbf{x}$ (not in the training set) by some linear prediction $\hat{y} = \mathbf{x}^\top \widehat{\mathbf{w}}$ where the *weight* $\widehat{\mathbf{w}} \in \Re^d$ minimizes the least-squares training cost

$$\sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

where in the matrix $\mathbf{X} \in \Re^{n \times d}$, the transposed sample feature vectors $\mathbf{x}_i^\top$ constitute the d-dimensional row vectors, and the n-dimensional vectors of training measurements $\mathbf{x}^j = ((\mathbf{x_1})_j, \ldots, (\mathbf{x_n})_j)^\top$ for $j = 1, \ldots, d$ correspond to the column vectors (see Figure 2). and $\mathbf{y} = (y_1, \ldots, y_n)^\top$.

Let us actually build on the example mentioned above and view the measurements $(\mathbf{x_i})_j$ of each sample $\mathbf{x}_i$ as a sequence of measurements, e.g. velocities of car i, over time $j = 1, \ldots, d$.

(a) Is this problem in a supervised or unsupervised learning setting? **Please explain.**

(b) Suppose that we want to learn (from our training set) to predict the final location y from only the first t measurements. Denoting the prediction of y from the first t measurements by $\hat{y}^t$, we thus want to use $(\mathbf{x})_j, j = 1, \ldots, t$ to predict y. If we now learn how to obtain $\hat{y}^t$ for each $t = 1, \ldots, d$, we end up with a sequence of estimators $\hat{y}^1, \ldots, \hat{y}^d$ for each car.

Provide a method to obtain $\hat{y}^t$ for each t. Note that we will obtain a different model for each t.

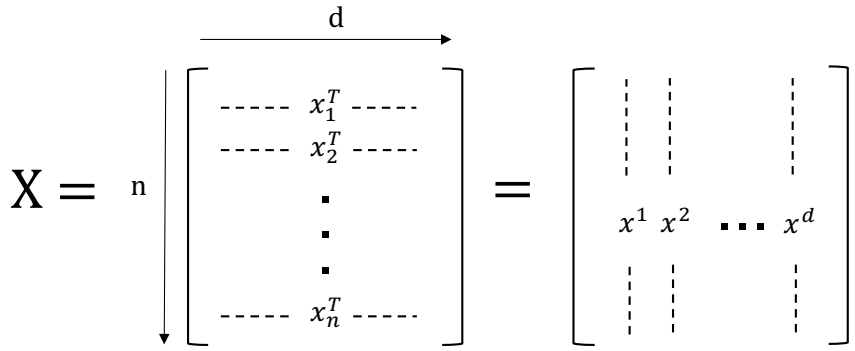

Figure 2: Dimensions, column and row vector notation for matrix $\mathbf{X}$

(c) Someone suggests that maybe the measurements themselves are partially predictable from the previous measurements, which suggests employing a two stage strategy to solve the original prediction problem: First we predict the t-th measurement $(\mathbf{x})_t$ based on the previous measurements $(\mathbf{x})_1, \ldots, (\mathbf{x})_{t-1}$. Then we look at the differences (sometimes deemed the "innovation") between the actual t-th measurement we obtained and our prediction for it, i.e. $(\Delta\mathbf{x})_t := (\mathbf{x})_t - (\widehat{\mathbf{x}})_t$ for $t > 1$ and assume $(\Delta\mathbf{x})_1 = (\mathbf{x})_1$. Finally, we use $(\Delta\mathbf{x})_1, \ldots, (\Delta\mathbf{x})_t$ to obtain a prediction $\tilde{y}^t$.

In order to learn the maps which allow us to (1) take $(\mathbf{x})_1, \ldots, (\mathbf{x})_{t-1}$ to obtain $(\Delta\mathbf{x})_1, \ldots, (\Delta\mathbf{x})_t$ and (2) take $(\Delta\mathbf{x})_1, \ldots, (\Delta\mathbf{x})_t$ to predict $\tilde{y}^t$, we again use our training set. Specifically for each t, in stage (1), we fit the vectors of training measurements $\mathbf{x}^1, \ldots, \mathbf{x}^{t-1}$ linearly to $\mathbf{x}^t$ using least squares for each t. In stage (2), we use the innovation vectors $(\Delta\mathbf{x}^1, \ldots, \Delta\mathbf{x}^t)$ to predict $\mathbf{y}^t$ again using least squares. Let's define the matrix $\tilde{\mathbf{X}}^t := (\Delta\mathbf{x}^1, \ldots, \Delta\mathbf{x}^t)$ and $\tilde{\mathbf{X}} = \tilde{\mathbf{X}}^d$.

Show how we can learn the best linear predictions $\widehat{\mathbf{x}}^t$ from $\mathbf{x}^1, \ldots, \mathbf{x}^{t-1}$. Then **provide an expression** for $\tilde{\mathbf{y}}^t$ depending on the innovations $\Delta\mathbf{x}^1, \ldots, \Delta\mathbf{x}^t$.

When presented with a new feature vector $\mathbf{x}$, are the sequence of final predictions of the one-stage training $\hat{y}^t$ in (b) and two-stage training $\tilde{y}^t$ in (c) the same? **Explain your reasoning.**

(d) **Which well-known procedure do the steps to obtain $\tilde{\mathbf{X}}$ from $\mathbf{X}$ remind you of**? (**HINT:** Think about how the column vectors in $\tilde{\mathbf{X}}$ are geometrically related.)

Is there an efficient way to update the weight vector $\widehat{\mathbf{w}}^t$ from $\widehat{\mathbf{w}}^{t-1}$ when computing the sequence of predictions $\tilde{y}^t$? Here, $\widehat{\mathbf{w}}^t$ are the weights for the second stage in (c).

(e) Now let's consider the more general setting where we now want to predict a target vector $\mathbf{y} \in \Re^k$ from a feature vector $\mathbf{x} \in \Re^d$, thus having a training set consisting of observations $(\mathbf{x}_i, \mathbf{y}_i)$ for $i = 1, \ldots, n$.

Instead of learning a weight vector $\mathbf{w} \in \Re^d$, we now want a linear estimate $\widehat{\mathbf{y}} = \hat{\mathbf{W}}\mathbf{x}$ with a weight matrix $\hat{\mathbf{W}} \in \Re^{k \times d}$ instead. From our samples, we obtain wide matrices $\mathbf{Y} \in \Re^{k \times n}$ with columns $\mathbf{y}_1, \ldots, \mathbf{y}_n$ and $\mathbf{X} \in \Re^{d \times n}$ with columns $\mathbf{x}_1, \ldots, \mathbf{x}_n$ (note that this is the transpose of $\mathbf{X}$ in Figure 2!). In order to learn $\hat{\mathbf{W}}$ we now want to minimize $\|\mathbf{Y} - \mathbf{W}\mathbf{X}\|_F^2$ where $\|\cdot\|_F$ denotes the Frobenius norm of matrices, i.e. $\|\mathbf{L}\|_F^2 = \mathrm{tr}(\mathbf{L}^\top \mathbf{L})$.

Show how to find $\hat{\mathbf{W}} = \arg\min_{\mathbf{W} \in \Re^{k \times d}} \|\mathbf{Y} - \mathbf{W}\mathbf{X}\|_F^2$ using vector calculus as reviewed in Discussion 0 and 1.

(f) In the setting of problem (e), **argue why** the computation of the best linear prediction $\widehat{\mathbf{y}}$ of a target vector $\mathbf{y}$ using a feature vector $\mathbf{x}$ can be solved by separately finding the best linear prediction for each measurement $(\mathbf{y})_j$ of the target vector $\mathbf{y}$.

5 A Simple Classification Approach

Make sure to submit the code you write in this problem to "HW1 Code" on Gradescope.

Classification is an important problem in applied machine learning and is used in many different applications like image classification, object detection, speech recognition, machine translation and others.

In *classification*, we assign each datapoint a class from a finite set (for example the image of a digit could be assigned the value $0, 1, \ldots, 9$ of that digit). This is different from *regression*, where each datapoint is assigned a value from a continuous domain like $\mathbb{R}$ (for example features of a house like location, number of bedrooms, age of the house, etc. could be assigned the price of the house).

In this problem we consider the simplified setting of classification where we want to classify data points from $\mathbb{R}^d$ into *two* classes. For a linear classifier, the space $\mathbb{R}^d$ is split into two parts by a hyperplane: All points on one side of the hyperplane are classified as one class and all points on the other side of the hyperplane are classified as the other class.

The goal of this problem is to show that even a regression technique like linear regression can be used to solve a classification problem. This can be achieved by regressing the data points in the training set against -1 or 1 depending on their class and then using the level set of 0 of the regression function as the classification hyperplane (i.e. we use 0 as a threshold on the output to decide between the classes).

Later in lecture we will learn why linear regression is not the optimal approach for classification and we will study better approaches like logistic regression, SVMs and neural networks.

(a) The dataset used in this exercise is a subset of the MNIST dataset. The MNIST dataset assigns each image of a handwritten digit their value from 0 to 9 as a class. For this problem we only keep digits that are assigned a 0 or 1, so we simplify the problem to a two-class classification problem.

Download and visualize the dataset (example code included). Include three images that are labeled as 0 and three images that are labeled as 1 in your submission.

(b) We now want to use linear regression for the problem, treating class labels as real values $y = -1$ for class "zero" and $y = 1$ for class "one". In the dataset we provide, the images have already been flattened into one dimensional vectors (by concatenating all pixel values of the two dimensional image into a vector) and stacked as rows into a feature matrix $\mathbf{X}$. We want to set up the regression problem $\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ where the entry y_i is the value of the class (-1 or 1) corresponding to the image in row $\mathbf{x}_i^\top$ of the feature matrix. **Solve this regression**

problem for the training set and report the value of $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ as well as the weights $\mathbf{w}$. For this problem you may only use pure Python and numpy (no machine learning libraries!).

(c) Given a new flattened image $\mathbf{x}$, one natural rule to classify it is the following one: It is a zero if $\mathbf{x}^\top \mathbf{w} \leq 0$ and a one if $\mathbf{x}^\top \mathbf{w} > 0$. **Report what percentage of the digits in the training set are correctly classified by this rule. Report what percentage of the digits in the test set are correctly classified by this rule.**

(d) **Why is the performance typically evaluated on a separate test set (instead of the training set) and why is the performance on the training and test set similar in our case?** We will cover these questions in more detail later in the class.

(e) Somebody suggests to use 0 (for class 0) and 1 (for class 1) as the entries for the target vector $\mathbf{b}$. Try out how well this is doing (make sure to adapt the classification rule, i.e. the threshold set for the outputs). **Report what percentage of digits are correctly classified using this approach on the training set and test set.** How are the performances of the two approaches if you add a bias column to the feature matrix $\mathbf{X}$? A bias column is a column of all ones, i.e. the new feature matrix $\mathbf{X'}$ is

$$\mathbf{X'} = \begin{bmatrix} \mathbf{x}_0^\top & 1 \\ \mathbf{x}_1^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^\top & 1 \end{bmatrix}$$

Report what percentage of digits are correctly classified using regression targets $0/1$ and $-1/1$ with bias on the training set and test set. Try to explain the results!

6 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn the material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.