CS 475 Machine Learning (Fall 2023): Assignment 1

Due on September 26th, 2023 at 4:00 PM ET

Instructions: Please read these instructions carefully and follow them precisely. Feel free to ask the instructor if anything is unclear!

1. Please submit your solutions electronically via Gradescope.
2. Please submit a PDF file for the written component of your solution including derivations, explanations, etc. You can create this PDF in any way you want: typeset the solution in LATEX (preferred/recommended), or type it in Word or a similar program and convert/export to PDF. We recommend that you restrict your solutions to the space allocated for each problem; you may need to adjust the white space by tweaking the argument to `\vspace{xpt}` command. Please name this document <firstname-lastname>-sol1.pdf.
3. **Late submissions:** You have a total of 96 late hours for the entire semester that you may use as you deem fit. After you have used up your quota, there will be a penalty of 50% of your grade on a late homework if submitted within 48 hours of the deadline and a penalty of 100% of your grade on the homework for submissions that are later than 48 hours past the deadline.
4. **What is the required level of detail?** When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include the figure as well as the code used to plot it (and clearly explain in the README what the relevant files are). If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important. When submitting code, please make sure it's reasonably documented, and describe succinctly in the written component of the solution what is done in each `py`-file.

Name: Tianjian Bai

I. Supervised Learning

In class, we defined machine learning to be "the design of algorithms that learn rules from data, adapt to changes, and improve their performance on a certain task as they gain more experience." In this problem, we will give a formal description of what we mean by task, experience and how we measure performance of a learning algorithm in the context of supervised learning. We will motivate a natural learning algorithm and describe some of the key issues in machine learning.

Problem 1(a) [15 points] Begin by describing the formal setup for the (supervised) learning problem, with a special emphasis on

(i) motivating the definition of task as an uncertain relationship between inputs and outputs;

(ii) describing how to measure the performance of a predictor on the task;

(iii) what would be the ideal goal of learning, and why it is infeasible;

(iv) the role of training data; and

(v) what do we need for the learning to succeed (e.g., how training and test data are drawn, the need for an inductive bias, etcetera).

Problem 1(b) [15 points] Define what we formally mean by an algorithm and discuss in detail how we would evaluate the performance of a learning algorithm, both theoretically and empirically. What are the key issues that we should consider when designing a good machine learning algorithms? How do we address then in practice?

Problem 1(c) [15 points] Motivate the learning algorithm based on empirical risk minimization.

Problem 1(d) [15 points] Define the Bayes optimal predictor and discuss the role of Bayes optimal risk in defining the baseline to characterize the theoretical performance for learning.

Advice: Throughout your description, please leverage examples from real-world machine learning problems to illustrate various points. Add any remarks that you find interesting or helpful.

1. (a)
   i. Given 2 sets $\mathcal{X}$ and $\mathcal{Y}$ corresponding to our input and label set, respectively, we seek to construct a prediction rule $f : \mathcal{X} \to \mathcal{Y}$ which accurately predicts some $Y \in \mathcal{Y}$ from an input $X \in \mathcal{X}$. We additionally make the assumption that $(X, Y) \sim P_{XY}$, intuitively representing the inherent uncertainty between the inputs and outputs. For example, we may seek to predict the performance of a publicy-traded security using time series data and technical indicators, or create a predictor that can determine the number of humans in an image.

   ii. The performance of a predictor on this task is evaluated with a loss function, $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, which maps a tuple of the true label $Y \in \mathcal{Y}$ and its prediction $f(X) \in \mathcal{Y}$ to a real number related to their "closeness." Common loss functions include square loss and log loss.

   iii. The ideal goal for learning is to minimize the expectation of the loss, also called the risk, which is defined by $\mathbb{E}_{XY}[l(Y, f(X))]$. Intuitively, this defines how much error the predictor is privy to on average. This minimum and optimal risk is also called Bayes' Risk. However, obtaining Bayes' Risk is generally infeasible because we do not know the true distribution P_{XY}, thus we can not compute the true expectation, only estimate it.

   iv. The goal of training data is to provide a proxy $\{(X_i, Y_i)\}_{i=1}^{N}$ of the distribution P_{XY}. This data is used to learn the prediction rule f which is evaluated using the empirical risk $\frac{1}{N}\sum_{i=1}^{N} l(Y_i, f(X_i))$, which converges to the true risk under the Law of Large Numbers. This training data is often obtained by randomly sampling or annotating a set of data from the input set.

   v. In order for learning to succeed we also need to apply inductive bias to the task at hand—a set of assumptions and prior knowledge about the problem such as the class of prediction function, the learning algorithm, how to construct and represent the input and label sets, what loss function to choose etc. We also need to obtain a sufficient amount of training and test data to accurately approximate risk. These data sets are separated during the learning and evaluation processes in order to ensure that our predictor actually generalizes to the problem rather than simply overfitting the training data.

1. (b) An algorithm is some process that ingests training data and generates a corresponding predictor for the task at hand. A good machine learning algorithm does not overfit the training data and manages to generalize to data it has not seen (such as the test data). Accurate evaluation of overfitting can be obtained through the train/test split, and other techniques such as regularization are often used. The efficiency of the algorithm is also a concern and this often reduces to the efficiency of the optimization procedure, such as gradient descent. The performance of a learning algorithm can be defined by the Generalization Error, $\mathbb{E}_{D_n}\left[\mathbb{E}\left[l(Y, \hat{f}(X)\right]\right]$, which is an expectation w.r.t the random training data. In practice, we use the validation error $\frac{1}{n}\sum_{i=1}^{n} l(Y_i', \hat{f}(X_i'))$ to estimate generalization error, where $\{(X_i', Y_i')\}_{i=1}^{n}$ is our test data.

   (c) Learning algorithms based on empirical risk minimization (which is defined in part (a)(iv)) take advantage of the Law of Large Numbers, as the empirical risk converges to true risk for large n. Thus, minimizing empirical risk is a proxy towards creating a predictor that obtains Bayes' Risk, also known as Bayes' optimal predictor.

   (d) Bayes' optimal predictor is a function $f*$ which obtains the minimum risk, or $R(f*) \leq R(f)$ for all f. Although this optimal rule is not computable because we do not know the true distribution P_{XY}, it establishes a theoretical upper bound on how well any predictor can perform for a given problem and loss function. In practice, Bayes' optimal risk can be used to assess how close a model is to the theoretical best performance.

II. Regression

In this set of problems we will look at the regression problem and the maximum likelihood (ML) approach, with the goal to understand a bit better some of their properties.

We will start with simple linear regression, defined by

$$\hat{y} = f(\mathrm{x}; \mathrm{w}) = \mathrm{w} \cdot \mathrm{x} = \sum_{j=0}^{d} w_j x_j \tag{1}$$

It is assumed in (1) that we have augmented the inputs $\mathrm{x} \in \mathbb{R}^d$ by adding 1 as the "zeroth" dimension, $x_0 \equiv 1$. This assumption is valid for the remainder of this problem set, unless stated otherwise.

Assume that we are given n training examples $(\mathrm{x}_1, y_1), (\mathrm{x}_2, y_2), \ldots, (\mathrm{x}_n, y_n)$, where each input data point x_i has d real-valued features. The goal of regression is to learn to predict y from x. We refer to $\mathrm{X} = [\mathrm{x}_1, \ldots, \mathrm{x}_n]^\top \in \mathbb{R}^{n \times (d+1)}$ as the design matrix and $\mathrm{y} = [y_1, \ldots, y_n]^\top$ as the response vector.

Let $\widehat{\mathrm{w}}$ be the linear regression parameters estimated using the least squares procedure from data, and w^* be the best possible linear regression parameters for the underlying $p(\mathrm{x}, y)$.

It was shown in the lectures that the sample correlation between prediction errors made by $\widehat{\mathrm{w}}$ are uncorrelated with the training data $\{\mathrm{x}_i\}_{i=1}^n$, and a more general statement was made but not proven: that the prediction errors are in fact uncorrelated with values of any linear function of x computed on the training inputs.

Later, we made use of a similar fact regarding the prediction errors made by w_*. Here we will prove this more general fact. First, to make things precise, let us recall the definition of correlation between two (continuous) random variables U and V . Let $p_{u,v}$ be their joint probability density, and p_u and p_v the corresponding marginal densities. Then, the correlation (coefficient) between U and V is defined as

$$\rho(U,V) \equiv \frac{\mathbb{E}_{p_{u,v}}[(U - \mathbb{E}_{p_u}[U])(V - \mathbb{E}_{p_v}[V])]}{\sqrt{\mathrm{var}(U)\ \mathrm{var}(V)}} \tag{2}$$

where $\mathrm{var}(U)$ and $\mathrm{var}(V)$ are the variances of U and V under their respective marginal distributions. Intuitively, correlation measures how well one variable is linearly predictable from the other. We will now prove a fact from which it follows that prediction errors of w_* are uncorrelated with any linear function of x:

Problem 2 [15 points] Show rigorously that for any $a \in \mathbb{R}^{d+1}$,

$$\mathbb{E}_{p(\mathrm{x},y)}\left[(y - \mathrm{w}_*{}^\top \mathrm{x})\mathrm{a}^\top \mathrm{x}\right] = 0 \tag{3}$$

Advice: You want to write explicitly what the definition of w as the best linear regression under $p(x,y)$ implies, in terms of minimizing the expected loss (which, to remind you, is an integral w.r.t. x and y) similarly to how we treated the empirical loss in the case of $\widehat{w}$.

By definition, w_* minimizes the risk $\iint_{\mathcal{X}\times\mathcal{Y}} (y - w_*^T x)^2 p(x,y)dxdy$

$\Rightarrow \frac{\partial}{\partial w_*}\iint_{\mathcal{X}\times\mathcal{Y}} (y - w_*^T x)^2 p(x,y)dxdy = \iint_{\mathcal{X}\times\mathcal{Y}} \frac{\partial}{\partial w_*}(y - w_*^T x)^2 p(x,y)dxdy$

$= -2\iint_{\mathcal{X}\times\mathcal{Y}} x(y - w_*^T x)p(x,y)dxdy = 0 \Rightarrow \iint_{\mathcal{X}\times\mathcal{Y}} x(y - w_*^T x)p(x,y)dxdy = 0$

$\Rightarrow \iint_{\mathcal{X}\times\mathcal{Y}} a^T x(y - w_*^T x)p(x,y)dxdy = a^T * 0 = 0$

$\Rightarrow \mathbb{E}\left[(y - w_*^T x)a^T x\right] = 0$ ■

Problem 3 [10 points] Explain (succinctly but precisely) how the original statement, regarding zero empirical correlation between the prediction errors made by $\widehat{\mathrm{w}}$ estimated by least squares with any linear function of the training $\{\mathrm{x}_i\}$, follows from (3).

Suppose our training data $\{(X_i, Y_i)\}_{i=1}^N$ defines a distribution $\tilde{p}(x, y)$ where $\tilde{p}(X_j, Y_j) = \frac{1}{N}$.
$\hat{w}$ which minimizes $\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{w} \cdot x_i)^2$ is equivalent to the best linear regression parameters under the distribution $\tilde{p}$.
It follows that $\mathbb{E}_{\tilde{p}(x,y)}\left[(y_i - \hat{w}^T x_i)(a^T x_i)\right] = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{w}^T x_i)x_{ij} = 0$ where $a = e_j$
$\Rightarrow \sum_{i=1}^{N}(y_i - \hat{w}^t x_i)x_{ij} = 0$ for $\hat{w}$ which minimizes LSQ on the training data. ■

Next, we consider the effect of scaling the input in the regression problem. This becomes relevant, for instance, in polynomial regression with high order models, to prevent very large or very small values and the resulting potential for numerical instability (think about the case of $x = 0.01$ or $x = 1000$ for a 10^{th}-order model). It seems that a good solution could be to multiply each column of the design matrix X by a suitable number so that the range of that column (i.e. of the corresponding dimension in the regression input space) be fixed, say, to $[-1, 1]$.

Suppose that the data are scaled by multiplying the j-th dimension of the input by a non-zero number c_j. We will denote a single normalized data point by $\tilde{\mathrm{x}} \equiv [1, c_1 x_1, \ldots, c_d x_d]^\top$ and the resulting design matrix by $\tilde{\mathrm{X}}$.

Problem 4 [15 points] Let $\widehat{\mathrm{w}}$ be the least squares estimate of the regression parameters from the unscaled X, and let $\tilde{\mathrm{w}}$ be the solution obtained from the scaled $\tilde{\mathrm{X}}$. Show that the scaling does not change optimality, in the sense that $\widehat{\mathrm{w}} \cdot \mathrm{x} = \tilde{\mathrm{w}} \cdot \tilde{\mathrm{x}}$
Hint: You may find it helpful to express the scaling as a linear operator, yielding a matrix-product expression, and to equip yourself with a matrix reference, such as the Matrix Cookbook (look it up if you are not already familiar with it!)

We can represent $\tilde{X} = XC$ where $C \in \mathbb{R}^{d+1 \times d+1}$, $C_{11} = 1, C_{ii} = c_{i-1}, C_{ij} = 0$ for $i \neq j$.
C is a diagonal matrix where each non-zero element corresponds to a scaling factor. Additionally, we can write that $\tilde{x} = Cx$.
We know that $\tilde{w} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T y = ((XC)^T XC)^{-1}(XC)^T y = (C^T X^T XC)^{-1}C^T X^T y$
Note that C^T is symmetric, and $X^T X$ is symmetric $\Rightarrow X^T XC$ is symmetric because each column is only scaled by a constant. Recall that A is symmetric $\Rightarrow A^{-1}$ exists, thus we know that $(C^T)^{-1}$ and $(X^T XC)^{-1}$ exists.
$\Rightarrow (C^T X^T XC)^{-1}C^T X^T y = (X^T XC)^{-1}(C^T)^{-1}C^T X^T y = (X^T XC)^{-1}X^T y$
$= C^{-1}(X^T X)^{-1}X^T y = C^{-1}\hat{w}$ by definition of the least squares estimate.
Now consider $\tilde{w}^T\tilde{x} = (C^{-1}\hat{w})^T Cx = \hat{w}^T(C^{-1})^T Cx = \hat{w}^T(C^T)^{-1}Cx = \hat{w}^T C^{-1}Cx = \hat{w}^T x$ ■