Homework 1 of CS 291K (Fall 2022)

University of California, Santa Barbara

Due on Oct 11, 2022 (Tuesday)

Notes:

- Solutions to the homework questions are mostly terse.
- You are welcome to discuss with your peers or the TAs if you run into challenges, but you need to declare any such discussion and everyone needs to write their own solutions independently.

---

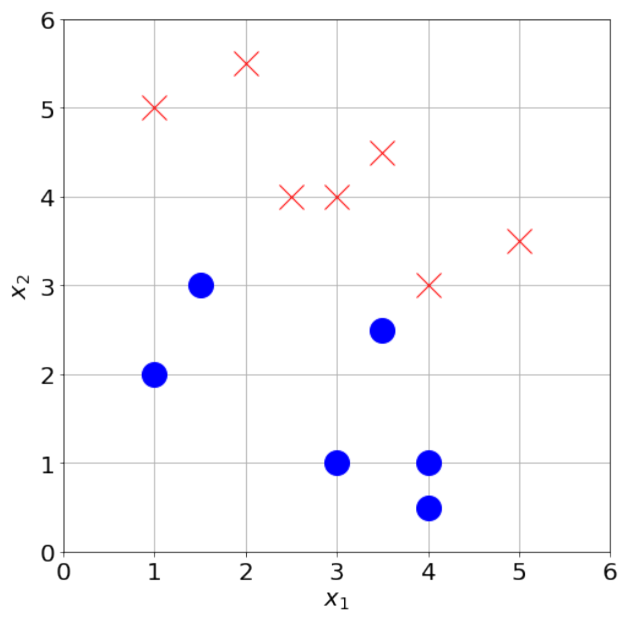
1 Why should I do this homework?

This homework covers the basics of machine learning. Through the theoretical exercise in Q1-4 and the coding exercise in Q5, you will gain deeper understanding of what machine learning is about. In Problem 1, you will develop an understanding of the idea of a "decision boundary". In Problem 2, you will do a simple theoretical exercise to see the gist of statistical learning theory, and to understand how data splitting works. In Problem 3, you will gain deeper understanding of linear regression and logistic regression. In Problem 4, you will work out the convergence analysis of gradient descent. In Problem 5, you will train a logistic regression classifier end-to-end and learn different components of a simple ML system.

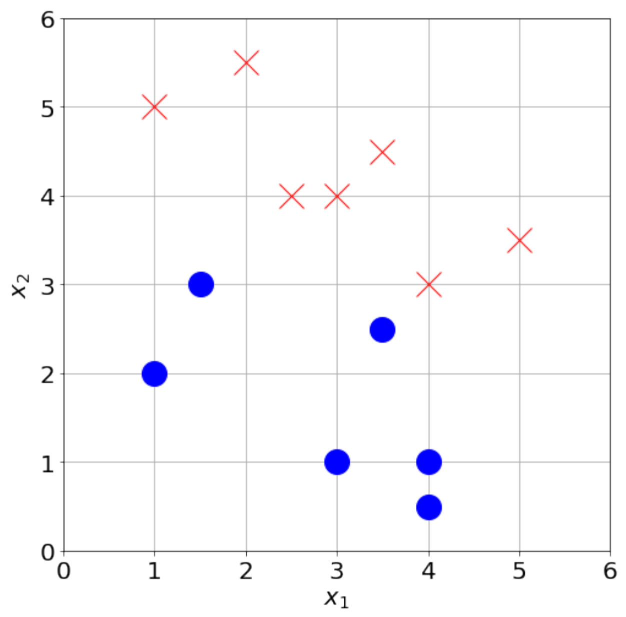
2 Homework problems

Problem 1. Classifiers and decision boundaries (15')

(a) (5') Let the feature space be $\mathbb{R}^2$, i.e., two continuous features. Consider a decision tree classifier with depth $= 2$ (branching on one variable by thresholding on each level, the variables to branch on can repeat). What are the possible boundaries representable by this classifier on a 2D feature plane? Is there a depth 2 decision tree that gives perfect classification in the example below?



(b) (5') Draw the decision boundary of a 1-nearest neighbor classifier in the example above.



(c) (5') Prove that the decision boundary of any (nontrivial) linear classifier is a hyperplane (an affine function). (Hint: for the above example with two variables, you may write it in the form of $x_2 = ax_1 + b$ for some slope a and offset b)

Problem 2. Training error, test error, Hold-out data and cross validation (15')

The next step is to build some tool for evaluating the performance of a classifier learned by a machine learning algorithm.

Let h be a classifier. More formally, $h : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ is the label space. And we denote the space of all classifiers to be $\mathcal{H}$. Let $(x_1, y_1), ..., (x_n, y_n)$ be the training dataset used for training, and assume that they are drawn i.i.d. (independently and identically distributed) from some unknown distributions $\mathcal{D}$ defined on $\mathcal{X} \times \mathcal{Y}$.

The most natural performance metric is the classification error, which measures the expected error rate.

$$\text{Err}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathbf{1}(h(x) \neq y)]$$

where $\mathbf{1}(\cdot)$ is the indicator function that outputs 1 is the condition is true and 0 otherwise.

We can also define the error that we calculate on a dataset. We denote the empirical error on this data set as

$$\hat{\text{Err}}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(h(x) \neq y).$$

Furthermore, we define the generalization error to be

$$\text{GenErr} = \max_{h\in\mathcal{H}} |Err(h) - \hat{Err}(h)|$$

— the difference between the training error and the *expected* error on a **new data point** from the same distribution $\mathcal{D}$ on **all** classifier $h \in \mathcal{H}$.

(**Remark:** $\text{Err}(\cdot)$ is just a special case of the risk function $R(\cdot)$ from the lecture. Similarly $\hat{\text{Err}}(\cdot)$ is a special case of the empirical risk function $\hat{R}(\cdot)$ (with the dependence on a particular dataset hidden, it can be specified explicitly too, e.g., in Part (c)).)

(a) (5') Let $\hat{h}$ be the learned classifier and h^* be the optimal classifier., i.e.

$$\hat{h} = \arg\min_{h} \hat{\text{Err}}(h), \quad h^* = \arg\min_{h} \text{Err}(h)$$

Prove that:

$$\text{Err}(\hat{h}) \leq \text{Err}(h^*) + 2\text{GenErr}.$$

(b) (5') Apply Hoeffding's inequality and union bound (see the two hammers we learned in Lecture 2) to provide a high probability bound for GenErr under the assumption that the hypothesis class is finite.

(c) (5') In practice, we can evaluate a classifier with data splitting. We randomly partition the data into a training data set and a holdout dataset.

Show that

$$\mathbb{E}[\hat{\text{Err}}_{\text{Holdout}}(\hat{h})] = \mathbb{E}[\text{Err}(\hat{h})],$$

where $\hat{\text{Err}}_{Holdout}$ denotes the empirical error calculated on the holdout set. Notice that $\hat{h}$ is random (because its training data is random), but here let us fix $\hat{h}$ and take the expectation over only the distribution of the *relevant* part of the dataset that is assumed to be drawn iid.

(Hint: If you don't know where to start, look up "Law of Total Expectation" and think about what to condition on.)

Problem 3. Linear regression and logistic regression (15')

(a) (5') Derive the maximum likelihood estimate of a linear Gaussian model.

Let $y_i = x_i^T\theta^* + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Fix $x_1, ..., x_n$. Write down the log-likelihood function. You may look up the probability density of a Gaussian random variable. This is also known as Conditional loglikelihood maximization because $x_1, .., x_n$ are not random (we are conditioning on them).

Show that the max-likelihood objective is equivalent to the empirical risk minimization objective that we focused in the lecture for the problem of linear regression under square losses.

Derive the close-form solution by taking the gradient and setting the gradient to 0.

(b) (5') Linear regression under distribution shift. So far we have considered the case when the input X is fixed. What happens if the new data points are drawn from a distribution Q. Calculate the expected square loss on the new test distribution, i.e., $\mathbb{E}[(\hat{\theta}^T x - y)^2]$.

The expectation is over the training labels $y_i \sim \mathcal{N}(x_i^T\theta^*, \sigma^2)$ for $i \in [n]$ and the test data $x \sim Q, y \sim \mathcal{N}(x^T\theta^*, \sigma^2)$. (The feature vectors of the training data $x_1, ..., x_n$ are fixed / not random; but the test data point x is random)

Simplify your solution as much as possible.

(Hint: $\hat{\theta}$ is random because its training data points are random).

(c) (5') Derive the gradient of the cross-entropy loss with respect to the parameter matrix $[\theta_1, ..., \theta_k] \in \mathbb{R}^{d\times k}$ a linear score function for a k-class logistic regression problem. Let $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = [k]$.

$$\ell([\theta_1, ..., \theta_k], (x, y)) = -\sum_{j=1}^{k} \mathbb{I}(y = j) \log([\text{Soft-Argmax}(x^T[\theta_1, ..., \theta_k])]_j).$$

(Hint: The solution should be simple and make intuitive sense.)

Write a sentence or two about the interpretation of the stochastic gradient descent algorithm that processes one data point at a time based on what the gradient looks like?

Problem 4. First-order Optimization (15')

In this problem we consider the unconstrained optimization problem

$$\min_x f(x).$$

You many assume $x \in \mathbb{R}^d$ (this works with infinite dimensional spaces too e.g., any Hilbert space).

(a) (5') In the lecture, we learned the convergence analysis of using gradient descent to minimize a general smooth function f to a stationary point. Replicate the analysis of what we learned from the lecture here. Then write what the *iteration complexity* bound is to obtain an ϵ-stationary point, i.e., the number of iterations needed to find a solution $\hat{x}$ that satisfies $\|\nabla f(\hat{x})\|_2^2 \le \epsilon$.

You may assume that f is differentiable and L-smooth.

(b) (10') Convergence of gradient descent under Polyak-Lojasiewicz condition (PL condition).

We say a function f satisfies μ-PL-condition if for any x

$$f(x) - f(x^*) \leq \frac{1}{2\mu}\|\nabla f(x)\|^2.$$

Show, by combining the **Descent Lemma** we learned from the class and the PL-condition, that running gradient descent from an initial point x_0 for k iterations satisfies

$$f(x_k) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f^*).$$

Write down an iteration complexity bound – the number of iterations needed to find a solution $\hat{x}$ such that $f(x_k) - f(x^*) \leq \epsilon$.

Problem 5 Elements of a Machine Learning System (40' and bonus 20')

Follow the instructions here: [Project Instructions] and use the StartupKit [Here]. You are required to do everything that is stated in the instruction page. But please **ignore the Grading rules** there.

The actual grading rules are the following:

(1) If you complete the basic coding requirement you get 30', the report will give you 10'.

(2) If you manage to get above 90% test accuracy on the leaderboard, you get 10' bonus.

(3) If you get above 95% test accuracy on the leaderboard you get another 10' bonus.

Only feature engineering is allowed for participating in the leaderboard. You are not allowed to change models (i.e., must use linear logistic regression). If your new feature is too large and gradescope cannot run, then they do not count.You may ask the TA to run your code and we will consider giving you a certain amount of bonus points at our discretion.