IFT-6390 Fundamentals of Machine Learning
Professor: Ioannis Mitliagkas

Homework 2 - Theoretical part

- This homework must be done and submitted to Gradescope and can be done in groups of up to 2 students. You are welcome to discuss with students outside of your group but the solution submitted by a group must be its own. Note that we will use Gradescope's plagiarism detection feature. All suspected cases of plagiarism will be recorded and shared with university officials for further handling.

- Only one student should submit the homework and you should add your group member on the submission page on Gradescope.

1. **Bias-Variance decomposition** [5 points]

   Consider the following data generation process: an input point x is drawn from an unknown distribution and the output y is generated using the formula

   $$y = f(x) + \epsilon,$$

   where f is an unknown deterministic function and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. This process implicitly defines a distribution over inputs and outputs; we denote this distribution by p.

   Given an i.i.d. training dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ drawn from p, we can fit the hypothesis h_D that minimizes the empirical risk with the squared error loss function. More formally,

   $$h_D = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} (y_i - h(x_i))^2$$

   where $\mathcal{H}$ is the set of hypotheses (or function class) in which we look for the best hypothesis/function.

   The expected error[1] of h_D on a fixed data point (x', y') is given by $\mathbb{E}[(h_D(x') - y')^2]$. Two meaningful terms that can be defined are:

   - The bias, which is the difference between the expected value of hypotheses at x' and the true value $f(x')$. Formally,
     $$bias = \mathbb{E}[h_D(x')] - f(x')$$
   - The variance, which is how far hypotheses learned on different datasets are spread out from their mean $\mathbb{E}[h_D(x')]$. Formally,
     $$variance = \mathbb{E}[(h_D(x') - \mathbb{E}[h_D(x')])^2]$$

[1] Here the expectation is over random draws of the training set D of n points from the unknown distribution p. For example (and more formally): $\mathbb{E}[(h_D(x')] = \mathbb{E}_{(x_1,y_1)\sim p} \cdots \mathbb{E}_{(x_n,y_n)\sim p}\, \mathbb{E}[(h_{\{(x_1,y_1),\ldots,(x_n,y_n)\}}(x')]$.

Show that the expected prediction error on (x', y') can be decomposed into a sum of 3 terms: $(bias)^2$, *variance*, and a *noise* term involving ϵ. You need to justify all the steps in your derivation.

2. **Feature Maps** [8 points]

In this exercise, you will design feature maps to transform an original dataset into a linearly separable set of points. For the following questions, if your answer is '*yes*', write the expression for the proposed transformation; and if your answer is '*no*', write a brief explanation. You are expected to provide explicit formulas for the feature maps, and these formulas should only use common mathematical operations.

(a) [3 points] Consider the following 1-D dataset (Figure 1). Can you propose a 1-D transformation that will make the points linearly separable?

Figure 1 – 1-D dataset. The points between $2k$ and $2k+1$ are labeled by X. The points between $2k+1$ and $2k+2$ are labeled by O.

(b) [3 points] Consider the following 2-D dataset (Figure 2). Can you propose a transformation into 1-D that will make the data linearly separable?

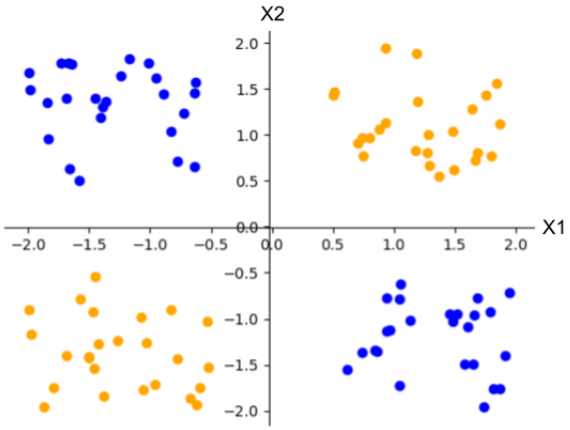

Figure 2 – 2D dataset.

(c) [4 points] Using ideas from the above two datasets, can you suggest a transformation of the following dataset (as shown in Figure 3) that makes it linearly separable? If '*yes*', also provide the kernel corresponding to the feature map you proposed. Remember that $K(x, y) = \phi(x) \cdot \phi(y)$, so you can find ϕ and do the dot product to find an expression for the kernel.

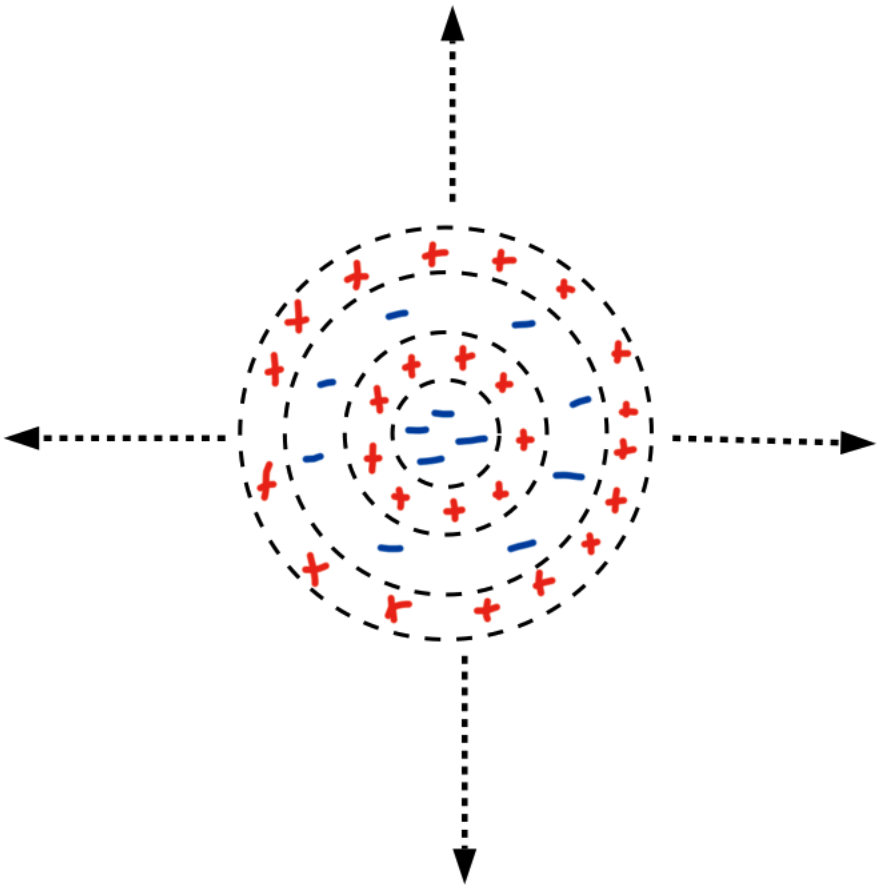

Figure 3 – Another 2D dataset. The points between the areas of radius $2k$ and $2k+1$ are labeled by -. The points between the areas of radius $2k+1$ and $2k+2$ are labeled by +.

3. **Derivatives and Gradients** [14 points]

   In this question, we will look at derivatives and gradients. Chapter 5 of Mathematics for Machine Learning book can be used as reference for this question.

   (a) [2 points] Compute the derivative $f'(x)$ for:

   $$f(x) = log\left(x^4\right) sin\left(x^3\right)$$

   (b) [2 points] Compute the derivative $f'(x)$ for:

   $$f(x) = exp\left(\frac{-1}{2\sigma}(x-\mu)^2\right)$$

Here σ,$\mu \in \mathbb{R}$.

(c) [4 points] Consider the following functions:

$$f_1(x) = sin(x_1) cos(x_2),\ x \in \mathbb{R}^2$$
$$f_2(x, y) = x^T y$$

Here $x, y \in \mathbb{R}^n$.

$$f_3(x) = xx^T$$

Here $x \in \mathbb{R}^n$.

i. What are the dimensions of $\frac{\partial f_i}{\partial x}$?
ii. Compute the jacobians.

(d) [6 points] Compute the derivatives $\frac{df}{dx}$ of the following functions:
i. Use the chain rule. Provide the dimensions of every derivative.

$$f(z) = exp\left(-\frac{1}{2}z\right)$$
$$z = g(y) = y^T S^{-1} y$$
$$y = h(x) = x - \mu$$

Here $x, \mu \in \mathbb{R}^D, S \in \mathbb{R}^{D \times D}$.

ii.

$$f(x) = tr\left(xx^T + \sigma I\right)$$

Here $x \in \mathbb{R}^D$ and $tr(A)$ is the trace of A.

iii. Use the chain rule to compute the derivatives and provide the dimensions of the partial derivative as well.

$$f = tanh(z)$$

Here $f \in \mathbb{R}^M$.

$$z = Ax + b$$

Here $x \in \mathbb{R}^N$, $A \in \mathbb{R}^{M \times N}$, $b \in \mathbb{R}^M$.

4. **Bayes Risk** [10 points]

In this exercise, we will show that the Bayes classifier (assuming we use the true underlying target distribution) minimize the true risk over all possible classifiers.

Recall that the goal of binary classification is to learn a mapping f from the input space, $\mathcal{X}$, to to the class space, $\mathcal{Y} = \{0, 1\}$. We can measure the loss of a classifier f using the $0-1$ loss; i.e.,

$$\ell(\hat{y}, y) = \mathbb{1}_{\{\hat{y} \neq y\}} = \begin{cases} 1, & \text{if } \hat{y} \neq y \\ 0, & \text{otherwise} \end{cases}$$

Recall that the true risk of f is defined by

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{P}} \left[\ell(f(x), y)\right]$$

where $\mathcal{P}$ is the underlying target distributioShow that, for anyn.

Usually, we assume that $\mathcal{P}$ is unknown and we infer f from a dataset drawn from $\mathcal{P}$. For this exercise, we will consider the Bayes classifier built using the target distribution $\mathcal{P}$, which is defined by

$$f^*(x) = \begin{cases} 1, & \text{if } \eta(x) \geq 1/2 \\ 0, & \textit{otherwise} \end{cases}$$

where $\eta(x) \equiv P(Y = 1|X = x)$.

You will show that for any function $g : \mathcal{X} \to \mathcal{Y}$ we have $R(g) \geq R(f^*)$

(a) First, show that $R(f) = P_{(x,y)\sim\mathcal{P}}(f(x) \neq y)$.

(b) Show that, for any $g : \mathcal{X} \to \mathcal{Y}$,

$$P(g(x) \neq y \mid X = x) = 1 - \left[\mathbb{1}_{\{g(x)=1\}}\eta(x) + \mathbb{1}_{\{g(x)=0\}}\left(1 - \eta(x)\right)\right]$$

(c) Using the answer to the previous question, and the fact that $\mathbb{1}_{\{g(x)=0\}} = 1 - \mathbb{1}_{\{g(x)=1\}}$, show that, for any $g : \mathcal{X} \to \mathcal{Y}$,

$$P\left(g(x) \neq Y|X = x\right) - P\left(f^*(x) \neq Y|X = x\right) = (2\eta(x) - 1)\left(\mathbb{1}_{\{f^*(x)=1\}} - \mathbb{1}_{\{g(x)=1\}}\right)$$

(d) Finally, show that, for any $g : \mathcal{X} \to \mathcal{Y}$,

$$(2\eta(x) - 1)\left(\mathbb{1}_{\{f^*(x)=1\}} - \mathbb{1}_{\{g(x)=1\}}\right) \geq 0$$

(e) Conclude.

5. **Leave one out cross-validation** [10 points]

Let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training sample set drawn i.i.d. from an unknown distribution p. Recall that leave-one-out cross-validation (LOO-CV) on a dataset of size n is the k-fold cross-validation technique we discussed in class for the special case where $k = n-1$. To estimate the risk (a.k.a. the test error) of a learning algorithm using D, LOO-CV involves

comparing each output y_i with the prediction made by the hypothesis of learning algorithm trained on all the data except the ith sample (x_i, y_i).

Formally, if we denote the hypothesis returned by the learning algorithm trained on $D \setminus \{(x_i, y_i)\}$ as $h_{D\setminus i}$, the leave-one-out error is given by

$$\text{error}_{LOO} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h_{D\setminus i}(x_i), y_i)$$

where $\mathcal{L}$ is the loss function.

In this exercise, we will investigate some interesting properties of this estimator.

Leave-one-out is unbiased

(a) Recall the definition of the risk of a hypothesis h for a regression problem with the mean squared error loss function.

(b) Let D' denote a dataset of size $n-1$. Show that

$$\mathop{\mathbb{E}}_{D\sim p}[\text{error}_{LOO}] = \mathop{\mathbb{E}}_{\substack{D'\sim p \\ (x,y)\sim p}}[(y - h_{D'}(x))^2]$$

where the notation $D \sim p$ means that D is drawn i.i.d. from the distribution p and where h_D denotes the hypothesis returned by the learning algorithm trained on D. Explain how this shows that error_{LOO} is an (almost) unbiased estimator of the risk of h_D.

Complexity of leave-one-out We will now consider LOO in the context of linear regression where inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are d-dimensional vectors. We use $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$ to denote the input matrix and the vector of outputs.

(c) Assuming that the time complexity of inverting a matrix of size $m \times m$ is in $\mathcal{O}(m^3)$, what is the complexity of computing the solution of linear regression on the dataset D?

(d) Using $\mathbf{X}_{-i} \in \mathbb{R}^{(n-1)\times d}$ and $\mathbf{y}_{-i} \in \mathbb{R}^{(n-1)}$ to denote the data matrix and output vector obtained by removing the ith row of $\mathbf{X}$ and the ith entry of $\mathbf{y}$, write down a formula of the LOO-CV error for linear regression. What is the complexity of evaluating this formula?

(e) It turns out that for the special case of linear regression, the leave-one-out error can be computed more efficiently. Show that in the case of linear regression we have

$$\text{error}_{LOO} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \mathbf{w}^{*\top}\mathbf{x}_i}{1 - \mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i}\right)^2$$

where $\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$ is the solution of linear regression computed on the whole dataset D. What is the complexity of evaluating this formula?