Homework 2 - Theoretical part

1. **Bias-Variance decomposition** [2 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 $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$. 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 $\mathbb{E}[(h_D(x') - y')^2]$. We will show that this error can be decomposed as a function of two meaningful terms:

   - The bias, which is the difference between the expected value of hypotheses at x' and the true value $f(x')$. Formally,

     $$\textit{bias} = \mathbb{E}[h_D(x')] - f(x')$$

   - The variance, which is how far hypotheses learned on different datasets are spread out from their mean $\mathbb{E}[h_D(x')]$. Formally,

     $$\textit{variance} = \mathbb{E}[(h_D(x') - \mathbb{E}[h_D(x')])^2]$$

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

[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): $\mathbb{E}[(h_D(x')] = \mathbb{E}_{(x_1,y_1)\sim p} \cdots \mathbb{E}_{(x_n,y_n)\sim p}\, \mathbb{E}[(h_{\{(x_1,y_1),\ldots,(x_n,y_n)\}}(x')]$.

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) [2 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:

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

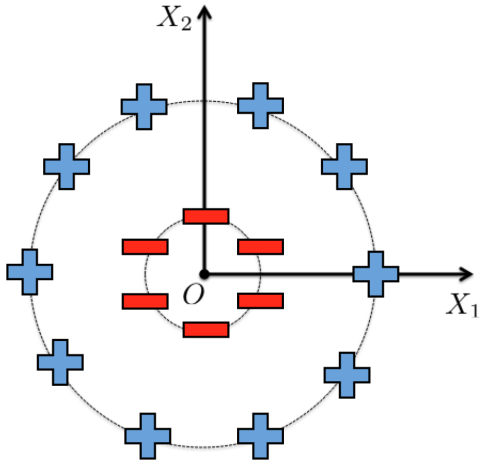

Figure 2:

(c) [4 points] Using ideas from the above two datasets, can you suggest a 2-D 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.

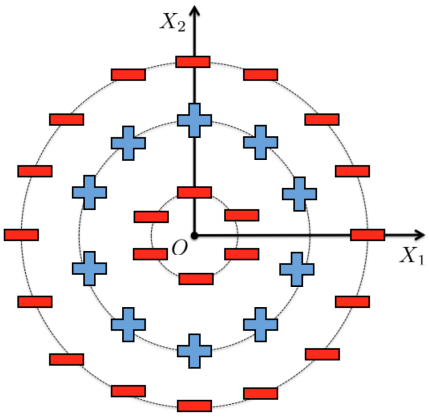

Figure 3:

3. **Optimization** [10 points]

Assume a quadratic objective function of the form:

$$f(x) = \frac{1}{2}x^T Ax + x^T b + a,$$

where $x \in \mathbb{R}^d$, $b \in \mathbb{R}^d$, $a \in \mathbb{R}$ and A is a $d \times d$ symmetric, positive definite matrix. This means that the matrix A admits the eigendecomposition $A = U\Lambda U^T$, where $U \in \mathbb{R}^{d\times d}$ is an orthonormal matrix and $\Lambda \in \mathbb{R}^{d\times d}$ is a diagonal matrix. The i-th column vector of U, denoted by $u_i \in \mathbb{R}^d$, represents the i-th eigenvector of A. The i-th diagonal element of Λ, denoted by $\lambda_i \in \mathbb{R}$, represents the i-th eigenvalue of A. We will assume here that all eigenvalues are unique. Furthermore, without loss of generality, the eigenvectors and eigenvalues in the decomposition can be ordered in such a way that

$$\lambda_1 > \lambda_2 > \ldots > \lambda_d > 0.$$

(a) Find all of the stationary points of $f(x)$ analytically, i.e. through a closed-form expression (Justify).

(b) Which of those stationary points are minima, maxima and saddle-points? Give a mathematically rigorous explanation why.

(c) Find the location, x^*, and value, $f(x^*)$, of the global minimum.

(d) Find the gradient of $f(x)$ at some point x. What are the dimensions of the gradient?

(e) Show how the gradient descent update rule looks like in this case by substituting $f(x)$ with its quadratic form above. Use the following notation: x_0 represents our point at initialization, x_1 represents our point after one step, etc.

(f) Consider the squared distance from optimum, $d(x_k) = \|x_k - x^*\|_2^2$. Find an exact expression (equality) of $d(x_k)$ that only depends on x_0 (not on other iterates x_i for $i > 0$), the number of iterations, k, as well as the eigenvectors, u_i, and eigenvalues, λ_i of A.

(g) Prove that there exist some assumptions on the hyperparameters of the algorithm, under which the sequence $d(x_k)$ converges to 0 as k goes to infinity. What are the exact necessary and sufficient conditions on the hyperparameters in order for $d(x_k)$ to converge to 0?

(h) The distance that you computed above is said to converge to 0 at an exponential rate (some other research communities use the term linear rate for the same type of convergence). We often care about the asymptotic rate of convergence, defined for this squared distance as

$$\rho = \exp\left(\lim_{k\to\infty} \frac{1}{2k} \ln d(x_k)\right),$$

where $\ln(\cdot)$ denotes the natural logarithm. Keep in mind that this rate depends on both the objective function, but also on the choice of hyperparameters.

Find an expression of ρ that only depends on the eigenvalues of A and the hyperparameter values.

(i) Prove that, for any choice of hyperparameter values, there exist constants $k_0 \geq 0$ and $C > 0$ such that

$$d(x_k) \leq C\rho^{2k}, \quad \forall k > k_0.$$

(j) Based on the above, we gather that in order to get fast convergence, we need a small ρ value. Find a value for the hyperparameter(s) of gradient descent that achieves the fastest asymptotic convergence rate possible.

4. **Least Squares Estimator and Ridge Regression** [10 points]

(a) In the problem of linear regression, we are given n observations $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, where each input $\mathbf{x}_i$ is a d-dimensional vector. Our goal is to estimate a linear predictor $f(.)$ which predicts y given $\mathbf{x}$ according to the formula

$$f(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\theta}, \tag{1}$$

Let $\mathbf{y} = [y_1, y_2 \ldots y_n]^\top$ be the $n \times 1$ vector of outputs and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n]^\top$ be the $n \times d$ matrix of inputs. One possible way to estimate the parameter $\boldsymbol{\theta}$ is through minimization of the sum of squares. This is the least squares estimator:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2. \tag{2}$$

i. Show that the solution of this minimization problem is given by

$$\boldsymbol{\theta}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

ii. When will the matrix $\mathbf{X}^\top \mathbf{X}$ be invertible and when will it be non-invertible? Give your answer in terms of properties of the dataset.

(b) A variation of the least squares estimation problem known as ridge regression considers the following optimization problem:

$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_2^2 \tag{3}$$

where $\lambda > 0$ is a regularization parameter. The regularizing term penalizes large components in $\boldsymbol{\theta}$ which causes the optimal $\boldsymbol{\theta}$ to have a smaller norm.

i. Derive the solution of the ridge regression problem. Do we still have to worry about the invertibility of $\mathbf{X}^\top \mathbf{X}$?

ii. Explain why the ridge regression estimator is likely to be more robust to issues of high variance compared with the least squares estimator.

iii. How does the value of λ affect the bias and the variance of the estimator?

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

Let $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training sample 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 consists in comparing each output y_i with the prediction made by the hypothesis returned by the learning algorithm trained on all the data except the ith sample (x_i, y_i).

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

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

where 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. Similarly to exercise 4, 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?

6. **Multivariate Regression** [10 points]

We consider the problem of learning a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^p$ from input-output training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ where each $\mathbf{x}_i$ is a d-dimensional vector and each $\mathbf{y}_i$ is a p-dimensional vector. We choose our hypothesis class to be the set of linear functions from $\mathbb{R}^d$ to $\mathbb{R}^p$, that is function satisfying $f(\mathbf{x}) = \mathbf{W}^\top\mathbf{x}$ for some $d \times p$ regression matrix $\mathbf{W}$, and we want to minimize the squared error loss function

$$J(\mathbf{W}) = \sum_{i=1}^{n} \|\mathbf{W}^\top\mathbf{x}_i - \mathbf{y}_i\|_2^2 \tag{4}$$

over the training data.

Let $\mathbf{W}^*$ be the minimizer of the empirical risk:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}\in\mathbb{R}^{d\times p}} J(\mathbf{W}).$$

(a) Derive a closed-form solution for $\mathbf{W}^*$ as a function of the data matrices $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{Y} \in \mathbb{R}^{n\times p}$.
(hint: once you have expressed $J(\mathbf{W})$ as a function of $\mathbf{X}$ and $\mathbf{Y}$, you may find the matrix cookbook useful to compute gradients w.r.t. to the matrix $\mathbf{W}$)

(b) Show that solving the problem from the previous question is equivalent to independently solving p independent classical linear regression problems (one for each component of the output vector), and give an example of a multivariate regression task where performing **independent** regressions for each output variables is not the best thing to do.

(c) The low rank regression algorithm addresses the issue described in the previous question by imposing a low rank constraint on the regression matrix $\mathbf{W}$. Intuitively, the low rank constraint encourages the model to capture linear dependencies in the components of the output vector.

Propose an algorithm to minimize the squared error loss over the training data subject to a low rank constraint on the regression matrix $\mathbf{W}$:

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W}) \qquad \text{s.t. } \operatorname{rank}(\mathbf{W}) \leq R.$$

(hint: There are different ways to do that. You could for example leverage the fact that $\operatorname{rank}(\mathbf{W}) \leq R$ *if and only if there exists* $\mathbf{A} \in \mathbb{R}^{d \times R}$ *and* $\mathbf{B} \in \mathbb{R}^{R \times p}$ *such that* $\mathbf{W} = \mathbf{AB}$*.)*