IFT-6390 Fundamentals of Machine Learning
Professor: Ioannis Mitliagkas

Homework 2 - Theoretical part

David Bieber with edits from the instructor and TAs

- This homework must be done and submitted to Gradescope and can be done in groups of at most 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.

- You need to submit your solution as a pdf file on Gradescope using the homework titled `(6390: GRAD) Theoretical Homework 2`.

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

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.

Answer:

Expected prediction error at $x' = \mathbb{E}_{train,y'}[(y' - h_D(x'))^2]$
The answer below makes use of following properties of the Expectation operator
$\mathbb{E}[ab] = \mathbb{E}[a]\,\mathbb{E}[b]$ (if 'a' and 'b' are independent random variables)
$\mathbb{E}[(a+b)^2] = \mathbb{E}[a^2] + 2\,\mathbb{E}[ab] + \mathbb{E}[b^2]$

$$
\begin{aligned}
&= \mathbb{E}[(\underbrace{(y' - f(x'))}_{\text{a}} + \underbrace{(f(x') - h_D(x'))}_{\text{b}})^2] \\
&= \mathbb{E}[\underbrace{(y' - f(x'))^2}_{\epsilon^2}] + 2\,\mathbb{E}[\underbrace{(y' - f(x'))}_{\epsilon}(f(x') - h_D(x'))] + \mathbb{E}[(f(x') - h_D(x'))^2] \\
&= \sigma^2 + 2\underbrace{\mathbb{E}[\epsilon]}_{0}\,\mathbb{E}[f(x') - h_D(x')] + \mathbb{E}[(f(x') - h_D(x'))^2] \\
&= \sigma^2 + \mathbb{E}[(f(x') - h_D(x'))^2]
\end{aligned}
$$

We will now expand the second term of the above expression:

$$
\begin{aligned}
&= \mathbb{E}[(f(x') - h_D(x'))^2] \\
&= \mathbb{E}[((f(x') - \mathbb{E}[h_D(x')]) + (\mathbb{E}[h_D(x')] - h_D(x')))^2] \\
&= \mathbb{E}[(f - \mathbb{E}[h_D])^2] + 2\,\mathbb{E}[(f - \mathbb{E}[h_D])(\mathbb{E}[h_D] - h_D)] + \mathbb{E}[(\mathbb{E}[h_D] - h_D))^2]
\end{aligned}
$$

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

We note that the expression $(f(x') - \mathbb{E}[h_D(x')])$ is a constant term known as *bias* that is independent of the expectation operator. Thus, the above expression simplifies to:

$$\begin{aligned} &= bias^2 + 2bias \underbrace{\mathbb{E}[(\mathbb{E}[h_D] - h_D)]}_{0} + \underbrace{variance}_{\text{as defined above}} \\ &= bias^2 + variance \end{aligned}$$

Thus, $\mathbb{E}[(y' - h_D(x'))^2] = \sigma^2 + bias^2 + variance$

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:

Yes, $x' = (x - 1.5)^2$

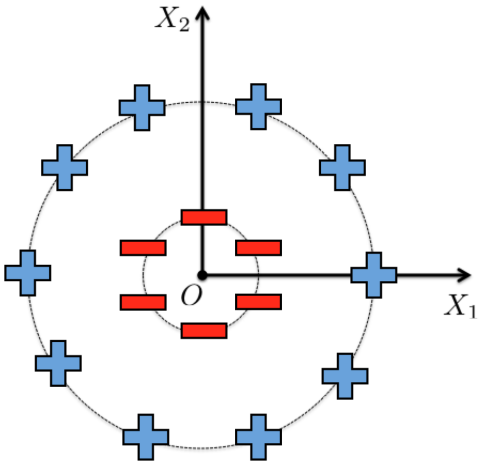

Figure 2:

   (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?

   Yes, $x' = \sqrt{X_1^2 + X_2^2}$

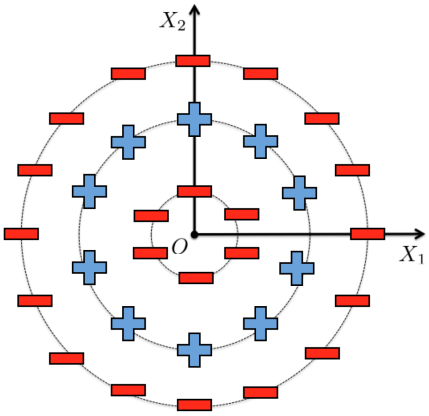

Figure 3:

(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.

Yes, one possible answer is $\phi(x) = (r, r^2)$, where $r = \sqrt{x_1^2 + x_2^2}$
The kernel of the above feature map is:

$$K(x, x') = \phi(x)^T \phi(x') = (\sqrt{(x_1^2 + x_2^2)(x'^2_1 + x'^2_2)}, ((x_1^2 + x_2^2)(x'^2_1 + x'^2_2)))$$

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).

$$\begin{aligned} \nabla f(x) &= \frac{1}{2}(A + A^\top)x + b \\ Ax + b &= 0 \qquad \forall \text{ stationary points x} \\ x &= -A^{-1}b \end{aligned}$$

Since A is positive definite, it is invertible, and this is a unique solution.

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

$$H(f(x)) = A$$

Since the Hessian is given by A, and A is positive definite, the stationary point $x = -A^{-1}b$ is a local minimum.

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

Since there is only one local minimum, and the boundary is not minimal (in particular there is not boundary because we are doing unconstrained optimization), the local minimum is a global minimum.

$$
\begin{aligned}
x^* &= -A^{-1}b \\
f(x^*) &= \frac{1}{2}(-A^{-1}b)^T A(-A^{-1}b) + (-A^{-1}b)^T b + a \\
&= \frac{1}{2}b^T A^{-1}b - b^T A^{-1}b + a \\
&= a - \frac{1}{2}b^T A^{-1}b
\end{aligned}
$$

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

$$\begin{aligned}\nabla f(x) &= \frac{1}{2}(A + A^\top)x + b \\ &= Ax + b\end{aligned}$$

The gradient has dimension d.

(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.

$$\begin{aligned}x_{n+1} &= x_n - \alpha \nabla f(x_n) \\ &= x_n - \alpha(Ax_n + b) \\ &= (I - \alpha A)x_n - \alpha b\end{aligned}$$

(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.

$$\begin{aligned}x_{k+1} - x^* &= x_k - x^* - \alpha(Ax_k + b) \\ &= x_k - x^* - \alpha(A(x_k - x^*) + Ax^* + b) \\ &= (I - \alpha A)(x_k - x^*) - \alpha(Ax^* + b) \\ &= (I - \alpha A)(x_k - x^*) - \alpha(-AA^{-1}b + b) \\ &= (I - \alpha A)(x_k - x^*) \\ &= (I - \alpha A)^{k+1}(x_0 - x^*)\end{aligned}$$

Here α denotes the step size.

$$
\begin{aligned}
d(x_k) &= \|(I - \alpha A)^k (x_0 - x^*)\|_2^2 \\
&= \|(UU^T - \alpha U \Lambda U^T)^k (x_0 - x^*)\|_2^2 \\
&= \|U(I - \alpha\Lambda)^k U^T (x_0 - x^*)\|_2^2 \\
&= \|(I - \alpha\Lambda)^k U^T (x_0 - x^*)\|_2^2 \\
&= \|(I - \alpha\Lambda)^k c\|_2^2
\end{aligned}
$$

where we defined the vector $c = U^T(x_0 - x^*)$. Now,

$$
d(x_k) = \sum_{i=1}^{d} (1 - \alpha\lambda_i)^{2k} c_i^2
$$

where c_i is the i-th element of vector c. Substituting c with its definition above, and also the value of x^* with the expression that we found in the previous questions, answers the question.

(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?

If $-1 < 1 - \alpha\lambda_i < 1 \; \forall i$ then $d(x_k)$ will go to 0 as $k \to \infty$.

This occurs if $0 < \lambda_i < \frac{2}{\alpha}$ since $\alpha > 0$,
or equivalently if $0 < \alpha < \frac{2}{\lambda_1}$.

Using our result from (f) we see that $0 < \lambda_i < \frac{2}{\alpha}$ is a **sufficient** condition for $d(x_k) \to 0$, since the $(I - \alpha\Lambda)$ terms go to 0 as $k \to \infty$.
And unless $x_0 = x^*$ (in which case $d(x_k) = 0$ for all k) or similar conditions (see next paragraph) are met, we can see this is a **necessary** condition as well, since otherwise the $(I - \alpha\Lambda)$ terms go to ∞ and do not cancel.

The condition that is required here is that x_0 projected onto the vectorspace spanned of some of the eigenvectors equals the projection of x^* on those same eigenvectors. If this is satisfied, then

we **need** only have $0 < \alpha < \frac{2}{\lambda_{max}}$ where the max is taken over the remaining eigenvalues. This result is obtained by observing from (d) and (e) that gradient descent operates independently across the directions given by the eigenvectors, so if the result is already optimal in a given eigenvector direction, we need not optimize in that direction.

In the best case, $x_0 = x^*$, there are **no necessary conditions**.

(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.

$$
\begin{aligned}
\rho &= \exp\left(\lim_{k\to\infty} \frac{1}{2k} \ln d(x_k)\right) \\
&= \exp\left(\lim_{k\to\infty} \frac{1}{2k} \ln \left[\sum_{i=1}^{d} c_i^2 (1-\alpha\lambda_i)^{2k}\right]\right) \\
&\qquad \text{for constants } c_i \text{ defined in (f).} \\
&= \exp\left(\lim_{k\to\infty} \frac{1}{2k} \ln \left[\sum_{j} c_j^2 (1-\alpha\lambda_j)^{2k}\right]\right) \\
&\qquad\qquad \Big(\text{where } j \in \arg\max |1-\alpha\lambda_j| \\
&\qquad\qquad\quad \text{since for } i \neq j,\ \frac{c_i^2(1-\alpha\lambda_i)^{2k}}{c_j^2(1-\alpha\lambda_j)^{2k}} \to 0 \text{ as } k \to \infty\Big) \\
&= c_d'^2(1-\alpha\lambda_d) \text{ or } -c_1'^2(1-\alpha\lambda_1) \\
&\qquad\qquad \text{if there is a single } j \text{ or else,} \\
&= c_1'^2(1-\alpha\lambda_1) + c_d'^2(1-\alpha\lambda_d) \\
&\qquad\qquad \text{when } 1-\alpha\lambda_1 = -(1-\alpha\lambda_d).
\end{aligned}
$$

We note that $j = 1$, $j = d$, or j takes on both values 1 and d (in the case that $1 - \alpha\lambda_1 = -(1 - \alpha\lambda_d)$.

So, among the eigenvalues, only the largest and smallest eigenvalues may affect the rate of convergence.

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

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

Recall that for the same constants c_i as in (f), we have:
$d(x_k) = \sum_{i=1}^{d} c_i^2 (1-\alpha\lambda_i)^{2k}$.
Then,

$$
\begin{aligned}
\rho &= c_j'^2(1-\alpha\lambda_j) \\
&\qquad \text{where } c_j' \text{ is a constant and } j \text{ maximizes } |1-\alpha\lambda_j| \\
C\rho^{2k} &= C'(1-\alpha\lambda_d)^{2k} \\
&\geq C''\sum_{i=1}^{d}(1-\alpha\lambda_d)^{2k} \\
&\geq C''\sum_{i=1}^{d}(1-\alpha\lambda_i)^{2k} \\
&\geq C'''d(x_k) \\
&\qquad \text{for constants } C', C'', C'''.
\end{aligned}
$$

Letting $C^{(4)} = \frac{C}{C'''}$, we have $C^{(4)}\rho^{2k} \geq d(x_k)$ as desired.

(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.

In order to minimize ρ while satisfying our constraints, we select α to minimize $|1-\alpha\lambda_i| > 0$ for all λ_i and also require that $\alpha > 0$.

To do this, we solve $-(1-\alpha\lambda_1) = 1-\alpha\lambda_d$, which gives $\boxed{\alpha = \frac{2}{\lambda_1+\lambda_d}}$.

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

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

$$\begin{aligned} \text{Let us define, } R(\boldsymbol{\theta}) &= \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 \\ R(\boldsymbol{\theta}) &= (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} [(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})] \\ \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) &= -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \end{aligned}$$

Setting the gradient $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta})$ to 0, we get

$$\begin{aligned} \mathbf{X}^T \mathbf{X} \boldsymbol{\theta}^* &= \mathbf{X}^\mathbf{T} \mathbf{y} \\ \boldsymbol{\theta}^* &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^\mathbf{T} \mathbf{y} \end{aligned}$$

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.

A is invertible if $Ax = 0$ implies $x = 0$.
$\mathbf{X}^\top \mathbf{X}$ is invertible if $\mathbf{X}^\top \mathbf{X} x = 0$ implies $x = 0$.

Assume $\mathbf{X}^\top \mathbf{X} x = 0$.
Then $x^\top \mathbf{X}^\top \mathbf{X} x = 0$, or equivalently $(\mathbf{X}x)^\top (\mathbf{X}x) = 0$.
Then $\mathbf{X}x = 0$. $\mathbf{X}x = 0$ implies $x = 0$ iff $\mathbf{X}$ is full rank (rank d).

So $\mathbf{X}^\top \mathbf{X}$ is invertible iff $\mathbf{X}$ is full rank (rank d).

In terms of the dataset:
The number of data points n must be at least d and the rank must be d – there must be d linearly independent data points. Otherwise, $\mathbf{X}^\top \mathbf{X}$ is not invertible.

(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 $\mathbf{X}^\top \mathbf{X}$?

$$\begin{aligned}
R(\boldsymbol{\theta}) &= \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda \|\boldsymbol{\theta}\|_2^2 \\
R(\boldsymbol{\theta}) &= (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta} \\
\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} [(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}] \\
\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) &= -2\mathbf{X}^\mathbf{T} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \mathbf{2}\lambda \boldsymbol{\theta}
\end{aligned}$$

Setting the gradient $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta})$ to 0, we get

$$\begin{aligned}
\mathbf{X}^T \mathbf{X} \boldsymbol{\theta}^* + \lambda \mathbf{I} \boldsymbol{\theta}^* &= \mathbf{X}^\mathbf{T} \mathbf{y} \\
\boldsymbol{\theta}^* &= (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-\mathbf{1}} \mathbf{X}^\mathbf{T} \mathbf{y}
\end{aligned}$$

We no longer require $\mathbf{X}^\top \mathbf{X}$ be invertible. Now, it is $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ that must be invertible. Since $\mathbf{X}^\top \mathbf{X}$ is positive semidefinite, $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ is positive definite (since $\lambda > 0$), and so is always invertible.

ii. Explain why the ridge regression estimator is likely to be more robust to issues of high variance compared with the least squares estimator.
The extra λ term serves to regularize the parameters found by the estimator. Since they are pushed toward zero by the regularization term, they are less susceptible to change due to changes in the sampled data.
Individual outliers due to data variance, for example, will have greater influence on the parameters estimated by the least squares estimator than on those estimated by the ridge regression estimator.

iii. How does the value of λ affect the bias and the variance of the estimator?
As λ increases, the bias increases and the variance decreases. At $\lambda = 0$, the estimator has minimal bias and maximum variance, and as $\lambda \to \infty$ the bias increases and the variance tends toward 0.

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.

risk $= \sum_D (y - h(x))^2$

(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.

$$\begin{aligned}
\mathop{\mathbb{E}}_{D\sim p}[\text{error}_{LOO}] &= \mathop{\mathbb{E}}_{D\sim p}\left[\frac{1}{n}\sum_{i=1}^{n}\ell(h_{D\setminus i}(x_i), y_i)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathop{\mathbb{E}}_{D\sim p}\left[\ell(h_{D\setminus i}(x_i), y_i)\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathop{\mathbb{E}}_{\substack{D'\sim p,\\(x,y)\sim p}}[\ell(h_{D'}(x), y)] \\
&= \mathop{\mathbb{E}}_{\substack{D'\sim p,\\(x,y)\sim p}}\left[(y - h_{D'}(x))^2\right]
\end{aligned}$$

This shows that $error_{LOO}$ is an unbiased estimator of the risk of $h_{D'}$, an $n-1$ point dataset, so almost an unbiased estimator of the risk of h_D as n gets large.

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?

Here we assume the complexity of multiplying an $a \times b$ matrix by a $b \times c$ matrix is $\mathcal{O}(abc)$.
Then calculating $\boldsymbol{\theta}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$ takes $\mathcal{O}\left(dnd + d^3 + ddn + dn\right) = \boxed{\mathcal{O}\left(d^3 + d^2 n\right)}$.

(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?

$$\text{error}_{LOO} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \left[(\mathbf{X}_{-i}^\top \mathbf{X}_{-i})^{-1} \mathbf{X}_{-i}^\top \mathbf{y}_{-i} \right]^\top x_i \right)^2$$

Evaluating this takes $\mathcal{O}\left(n \cdot (d^3 + d^2 n)\right) = \boxed{\mathcal{O}\left(d^3 n + d^2 n^2\right)}$ time.

(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?

From the previous question we have

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

We just need to show that

$$y_i - \mathbf{y}_{-i}^\top \mathbf{X}_{-i} (\mathbf{X}_{-i}^\top \mathbf{X}_{-i})^{-1} \mathbf{x}_i = \frac{y_i - \mathbf{w}^{*\top} \mathbf{x}_i}{1 - \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i}.$$

We have

$$\begin{aligned} &(y_i - \mathbf{y}_{-i}^\top \mathbf{X}_{-i} (\mathbf{X}_{-i}^\top \mathbf{X}_{-i})^{-1} \mathbf{x}_i)(1 - \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i) \\ &\quad = y_i - \mathbf{y}_{-i}^\top \mathbf{X}_{-i} (\mathbf{X}_{-i}^\top \mathbf{X}_{-i})^{-1} \mathbf{x}_i \\ &\quad\quad - y_i \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i + \mathbf{y}_{-i}^\top \mathbf{X}_{-i} (\mathbf{X}_{-i}^\top \mathbf{X}_{-i})^{-1} \mathbf{x}_i \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{x}_i. \end{aligned} \tag{4}$$

Using the fact that $\mathbf{X}^\top\mathbf{X} = \mathbf{X}_{-i}^\top\mathbf{X}_{-i} + \mathbf{x}_i\mathbf{x}_i^\top$ the last term in the sum can be developed as follows:

$$\begin{aligned}
&\mathbf{y}_{-i}^\top\mathbf{X}_{-i}(\mathbf{X}_{-i}^\top\mathbf{X}_{-i})^{-1}\mathbf{x}_i\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i \\
&\quad = \mathbf{y}_{-i}^\top\mathbf{X}_{-i}(\mathbf{X}_{-i}^\top\mathbf{X}_{-i})^{-1}\mathbf{x}_i - \mathbf{y}_{-i}^\top\mathbf{X}_{-i}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i.
\end{aligned}$$

Pluggin this back into Eq. (4) we obtain

$$\begin{aligned}
&(y_i - \mathbf{y}_{-i}^\top\mathbf{X}_{-i}(\mathbf{X}_{-i}^\top\mathbf{X}_{-i})^{-1}\mathbf{x}_i)(1 - \mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i) \\
&\quad = y_i - y_i\mathbf{x}_i^\top(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i - \mathbf{y}_{-i}^\top\mathbf{X}_{-i}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i \\
&\quad = y_i - (y_i\mathbf{x}_i^\top + \mathbf{y}_{-i}^\top\mathbf{X}_{-i})(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i \\
&\quad = y_i - (\mathbf{y}^\top\mathbf{X})(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i \\
&\quad = y_i - \mathbf{w}^{*\top}\mathbf{x}_i
\end{aligned}$$

where we used the fact that $\mathbf{y}^\top\mathbf{X} = y_i\mathbf{x}_i^\top + \mathbf{y}_{-i}^\top\mathbf{X}_{-i}$ in the equality before last. The result directly follows.

Evaluating this formula takes $\mathcal{O}\left(d^3 + d^2n\right)$ to compute $\mathbf{w}^*$ and $(\mathbf{X}^\top\mathbf{X})^{-1}$, and $\mathcal{O}\left(d^2\right)$ per point for additional computation, so a total run time of $\boxed{\mathcal{O}\left(d^3 + d^2n\right)}$, which is a factor of n better than in (d).

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 functions 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

$$J(\mathbf{W}) = \sum_{i=1}^{n} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \tag{5}$$

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 $\mathbf{W}$)*

$$\begin{aligned}
J(\mathbf{W}) &= \sum_{i=1}^{n} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \\
&= \sum_{i=1}^{n} \sum_{k=1}^{p} (\mathbf{W}_{:k}^\top \mathbf{x}_i - \mathbf{y}_{ik})^2 \\
&= \sum_{k=1}^{p} \sum_{i=1}^{n} (\mathbf{W}_{:k}^\top \mathbf{x}_i - \mathbf{y}_{ik})^2 \\
&= \sum_{k=1}^{p} J_k(\mathbf{W}_{:k})
\end{aligned}$$

We note that there is no overlap in the parameters $W_{:k}$ between the p terms of the outer sum. Hence, minimizing $J(\mathbf{W})$ is equivalent to independently minimizing each of the p $J_k(\mathbf{W}_{:k})$s.

We therefore will proceed by setting $\nabla J_k(\mathbf{W}_{:k}) = 0$ for each k, in order to solve the the minimizing $\mathbf{W}_{:k}$s.

$$
\begin{aligned}
J_k(\mathbf{W}_{:k}) &= \sum_{i=1}^{n} (\mathbf{W}_{:k}^\top \mathbf{x}_i - \mathbf{y}_{ik})^2 \\
&= (X\mathbf{W}_{:k} - Y_{:k})^\top (X\mathbf{W}_{:k} - Y_{:k}) \\
&= \|X\mathbf{W}_{:k} - Y_{:k}\|_2^2 \\
\nabla_{\mathbf{W}_{:k}} J_k(\mathbf{W}_{:k}) &= 0 \\
2(X\mathbf{W}_{:k} - Y_{:k}) &= 0 \\
X\mathbf{W}_{:k} &= Y_{:k} \\
X^\top X\mathbf{W}_{:k} &= X^\top Y_{:k} \\
\mathbf{W}_{:k} &= (X^\top X)^{-1} X^\top Y_{:k}
\end{aligned}
$$

Therefore

$$\mathbf{W}_{:k}^* = (X^\top X)^{-1} X^\top Y_{:k}$$

and

$$\mathbf{W}^* = (X^\top X)^{-1} X^\top Y$$

(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.

$$
\begin{aligned}
J(\mathbf{W}) &= \sum_{i=1}^{n} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \\
&= \sum_{i=1}^{n} \sum_{k=1}^{p} (\mathbf{W}_{:k}^\top \mathbf{x}_i - \mathbf{y}_{ik})^2 \\
&= \sum_{k=1}^{p} \sum_{i=1}^{n} (\mathbf{W}_{:k}^\top \mathbf{x}_i - \mathbf{y}_{ik})^2 \\
&= \sum_{k=1}^{p} J_k(\mathbf{W}_{:k})
\end{aligned}
$$

Note that there is no overlap in the parameters $W_{:k}$ between the p terms of the outer sum. Hence, minimizing $J(\mathbf{W})$ is equivalent to independently minimizing each of the p $J_k(\mathbf{W}_{:k})$s. Since each of these has the form of a regression single variable regression problem, we have shown solving (a) is equivalent to independently solving p independent classical linear regression problems as desired.

Independently performing regression per output variable is not the best thing to do when the output variables are highly correlated, especially e.g. if there is causal structure between the output variables.

As an example, suppose the output variables are 1) the temperature in Montreal on a given day, 2) the number of people sitting outside at Mila on that same day, and 3) some measure of productivity of Mila students such as lines of code written. To complete the example, we provide as inputs 1) the number of students physically at Mila on that day, 2) the day of the year, and 3) the number of announcements sent on Mila Slack in the previous week. In this example, independent predictions of the output variables may yield worse performance that using a low rank weight matrix to jointly predict the output variables, since there is some underlying structure that can be used to learn from one variable about the likely values of the others.

(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 $\mathbf{W}$:

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

We first note that $\operatorname{rank}(\mathbf{W}) \leq R$ iff there exists $\mathbf{A} \in \mathbb{R}^{d \times R}$ and $\mathbf{B} \in \mathbb{R}^{R \times p}$ such that $\mathbf{W} = \mathbf{AB}$.

We can therefore reparameterize $\mathbf{W} = \mathbf{AB}$, and seek to minimize

$$\min_{\mathbf{A} \in \mathbb{R}^{d \times R} \text{ and } \mathbf{B} \in \mathbb{R}^{R \times p}} J(\mathbf{AB})$$

$$\begin{aligned} J(\mathbf{AB}) &= \sum_{i=1}^{n} \|(\mathbf{AB})^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \\ &= \sum_{i=1}^{n} \|\mathbf{B}^\top \mathbf{A}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \end{aligned}$$

We propose to minimize $J(\mathbf{AB})$ via gradient descent. The algorithm is as follows:

1. Initialize $\mathbf{A}$ and $\mathbf{B}$ (the choice of initialization is not central to our proposed algorithm). Additionally choose a learning rate α and a stopping threshold ϵ.
2. Compute $\nabla_{\mathbf{A}} J(\mathbf{AB})$, $\nabla_{\mathbf{B}} J(\mathbf{AB})$.
3. Compute $\mathbf{A}' = \mathbf{A} - \alpha \nabla_{\mathbf{A}} J(\mathbf{AB})$ and $\mathbf{B}' = \mathbf{B} - \alpha \nabla_{\mathbf{B}} J(\mathbf{AB})$.
4. Update $\mathbf{A}$ to $\mathbf{A}'$ and $\mathbf{B}$ to $\mathbf{B}'$. If the difference between the old and new value of $\mathbf{A}$ exceeds ϵ, repeat steps 2-4.