CS 189/289A Introduction to Machine Learning
Fall 2024 Jennifer Listgarten, Saeed Saremi HW2

Due 09/25/24 11:59 PM PT

- Homework 2 consists of both written and coding questions.
- We prefer that you typeset your answers using LaTeX or other word processing software. If you haven't yet learned LaTeX, one of the crown jewels of computer science, now is a good time! Neatly handwritten and scanned solutions will also be accepted for the written questions.
- In all of the questions, **show your work**, not just the final answer.

Deliverables:

1. Submit a PDF of your homework to the Gradescope assignment entitled "HW 2 Written". **Please start each question on a new page.** If there are graphs, include those graphs in the correct sections. **Do not** just stick the graphs in the appendix. We need each solution to be self-contained on pages of its own.
    - **Replicate all of your code in an appendix**. Begin code for each coding question on a fresh page. Do not put code from multiple questions in the same page. When you upload this PDF on Gradescope, *make sure* that you assign the relevant pages of your code from the appendix to correct questions.
2. Submit all the code needed to reproduce your results to the Gradescope assignment entitled "HW 2 Code". Yes, you must submit your code twice: in your PDF write-up following the directions as described above so the readers can easily read it, and once in the format described below for ease of reproducibility.
    - You must set **random seeds** for all random utils to ensure reproducibility.
    - Do **NOT** submit any data files that we provided.
    - Please also include a short file named README listing your name, student ID, and instructions on how to reproduce your results.
    - Please take care that your code doesn't take up inordinate amounts of time or memory.

1 Multivariate Gaussians: A review

Multivariate Gaussian distributions crop up everywhere in machine learning, from priors on model parameters to assumptions on noise distributions. Being able to manipulate multivariate Gaussians also becomes important for analyzing correlations in data and preprocessing it for better regression and classification. We want to make sure to first cover the MVG fundamentals here.

Note that the probability density function of a non-degenerate (i.e. the covariance matrix is positive definite and, thus, invertible) multivariate Gaussian RV with mean vector, $\boldsymbol{\mu} \in \mathbb{R}^2$, and covariance matrix, $\Sigma \in \mathbb{R}^{2\times 2}$, is:

$$f(\mathbf{z}) = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{z}-\boldsymbol{\mu})\right)$$

(a) Consider a two dimensional, zero mean random variable $Z = \begin{bmatrix} Z_1 & Z_2 \end{bmatrix}^\top \in \mathbb{R}^2$. In order for the random variable to be jointly Gaussian, a necessary and sufficient condition which we call the *first characterization* is that

- Z_1 and Z_2 are each marginally Gaussian, and
- $Z_1|Z_2 = z$ is Gaussian, and $Z_2|Z_1 = z$ is Gaussian.

A *second characterization* of a jointly Gaussian zero mean RV $Z \in \mathbb{R}^2$ is that it can be written as $Z = AX$, where $X \in \mathbb{R}^2$ is a collection of i.i.d. standard normal RVs and $A \in \mathbb{R}^{2\times 2}$ is a matrix.

Let X_1 and X_2 be i.i.d. standard normal RVs. Let U denote a binary random variable uniformly that is equal to 1 with probability $\frac{1}{2}$ and -1 with probability $\frac{1}{2}$, independent of everything else.

For each of the below subproblems, complete the following *two* steps: (1) Using one of the characterizations given above, determine whether the RVs are jointly Gaussian. If using the second characterization, clearly specify the A matrix. (2) Calculate the covariance matrix of Z (regardless of whether the RVs are jointly Gaussian or not).

(i.) $Z_1 = X_1$ and $Z_2 = X_2$.

(ii.) $Z_1 = X_1$ and $Z_2 = X_1 + 2X_2$. If using the first characterization, assume that you already know $(Z_1|Z_2 = z)$ is Gaussian.

(iii.) $Z_1 = X_1$ and $Z_2 = -X_1$.

(iv.) $Z_1 = X_1$ and $Z_2 = UX_1$.

Solution: Before diving into the solution, recall that the covariance matrix of a vector random variable X with mean (vector) μ is given by $\Sigma = \mathbb{E}[(X-\mu)(X-\mu)^\top]$. In other words, entry i, j of the covariance matrix denotes the covariance between the random variables X_i and X_j, i.e., $\Sigma_{ij} = \text{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)]$. Additionally, two random variables U and V are said to be uncorrelated if $\text{Cov}(U, V) = 0$.

We will go through each pair of RVs now:

(i.) **First Characterization:** Z_1 and Z_2 are i.i.d. standard Gaussian, and so $(Z_1|Z_2 = z) \sim N(0,1)$. Also, $(Z_2|Z_1 = z) \sim N(0,1)$. Hence, the RVs are jointly Gaussian.

Second Characterization: $Z = AX$ with $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

(ii.) **First Characterization:** $Z_1 \sim N(0,1)$ and $Z_2 \sim N(0,5)$. The conditional distribution $(Z_2|Z_1 = z) \sim N(z,4)$, and we already know that $(Z_1|Z_2 = z)$ is normally distributed (if you are interested in calculating the precise distribution, see part (d)). Because the marginal distributions and the conditional distributions are Gaussian, the RVs are jointly Gaussian. For a more detailed derivation, recall that Z_2 is equal to $X_1 + 2X_2 = N(0,1) + 2N(0,1) = N(0,1) + N(0,4) = N(0,5)$. The conditional distribution $(Z_2|Z_1 = z) = z + 2X_2 = N(z,4)$.

Second Characterization: $Z = AX$ with $A = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}$.

Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & 1 \\ 1 & 5 \end{bmatrix}$.

This is because $\text{Cov}(Z_1, Z_2) = \text{Cov}(X_1, X_1 + 2X_2) = \text{Cov}(X_1, X_1) + \text{Cov}(X_1, 2X_2) = 1 + 0 = 1$, using the distributive property of covariance and the fact that X_1, X_2 are independent. Due to the symmetry property of covariance, we also know $\text{Cov}(Z_1, Z_2) = \text{Cov}(Z_2, Z_1)$.

(iii.) **First Characterization:** We have $Z_1 \sim N(0,1)$ and $Z_2 \sim N(0,1)$ marginally. However, we have $(Z_1|Z_2 = z) \sim N(-z, 0)$, which is a degenerate Gaussian. The other conditional distribution $(Z_2|Z_1 = z)$ is identical. Hence, the RVs are jointly Gaussian.

Second Characterization: $Z = AX$ with $A = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix}$.

Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$.

(iv.) **First Characterization:** As before, we have $Z_1 \sim N(0,1)$ and $Z_2 \sim N(0,1)$ marginally. In order to see this, write

$$\begin{aligned} p(Z_2 = z_2) &= p(Z_2 = z_2|U = 1)p(U = 1) + p(Z_2 = z_2|U = -1)p(U = -1) \\ &= \frac{1}{2}p(X_1 = z_2|U = 1) + \frac{1}{2}p(X_1 = -z_2|U = -1) \\ &= \frac{1}{2}p(X_1 = z_2) + \frac{1}{2}p(X_1 = -z_2) \\ &= \frac{1}{2}p(X_1 = z_2) + \frac{1}{2}p(X_1 = z_2) \\ &= p(X_1 = z_2) \end{aligned}$$

The random variable $(Z_2|Z_1 = z)$ is uniformly distributed on $\{-z, z\}$, and is therefore not Gaussian. The RVs are therefore not jointly Gaussian.

Covariance Matrix: $\Sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

This is because $\text{Cov}(Z_1, Z_2) = E[Z_1 Z_2] - E[Z_1]E[Z_2] = E[X_1^2 U] - 0 \cdot 0$. Since X_1^2 and

U are independent, $E[X_1^2 U] = E[X_1^2]E[U] = \text{Var}(X_1)E[U] = 1 \cdot 0 = 0$. Therefore $\text{Cov}(Z_1, Z_2) = 0$. Due to the symmetry property of covariance $\text{Cov}(Z_1, Z_2) = \text{Cov}(Z_2, Z_1)$.

(b) Show that two Gaussian random variables can be uncorrelated, but not independent (*Hint: use one of the examples in part (a)*). On the other hand, show that two uncorrelated, jointly Gaussian RVs are independent.

Solution: By definition, two random variables X and Y are independent iff $P(X = x, Y = y) = P(X = x)P(Y = y)$ for all x and y.

The last example in the previous part shows uncorrelated Gaussians that are not independent. In order to show that jointly Gaussian RVs (with individual variances σ_1^2 and σ_2^2) that are uncorrelated are also independent, assume without loss of generality that the RVs have zero mean, and notice that one can write the joint pdf as

$$\begin{aligned} f_Z(z_1, z_2) &= \frac{1}{(2\pi)\det\left(\Sigma_Z^{1/2}\right)} \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 & z_2 \end{bmatrix} (\Sigma_Z)^{-1} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) \\ &= \frac{1}{\sqrt{2\pi\sigma_1^2}} \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2\sigma_1^2} z_1^2\right) \exp\left(-\frac{1}{2\sigma_1^2} z_2^2\right) \\ &= f_{Z_1}(z_1) f_{Z_2}(z_2). \end{aligned}$$

The decomposition follows since Σ_Z is a diagonal matrix when the RVs are uncorrelated. Since we have expressed the joint PDF as a product of the individual PDFs, the RVs are independent.

(c) With the setup in (a), let $Z = VX$, where $V \in \mathbb{R}^{2\times 2}$, and $Z, X \in \mathbb{R}^2$. What is the covariance matrix Σ_Z? If X is not a multivariate Gaussian but has the identity matrix $I \in \mathbb{R}^{2\times 2}$ as its covariance matrix, is your computed Σ_Z still the covariance of Z?

Solution: The covariance matrix of a random vector Z (by definition) is given by $\mathbb{E}[(Z - \mathbb{E}[Z])(Z - \mathbb{E}[Z])^\top]$. Since $Z = VX$, we make that substitution to get

$$\begin{aligned} \Sigma_Z &= \mathbb{E}[(Z - \mathbb{E}[Z])(Z - \mathbb{E}[Z])^\top] \\ &= \mathbb{E}[(VX - \mathbb{E}[VX])(VX - \mathbb{E}[VX])^\top] \\ &= \mathbb{E}[(VX - V\,\mathbb{E}[X])(VX - V\,\mathbb{E}[X])^\top] \\ &= \mathbb{E}[V(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top V^\top] \\ &= V\,\mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\top]V^\top \\ &= V\Sigma_X V^T \end{aligned}$$

In lines 3 and 5, we applied the linearity of expectation to pull the constant matrix V out of the expectation (write it out in terms of the components of V to convince yourself!). Since $\Sigma_X = I$, it follows that $\Sigma_Z = VIV^T = VV^T$.

If V is written as:

$$V = \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}$$

then the result can also be written as:

$$\Sigma_Z = VV^T = \begin{pmatrix} V_{11}^2 + V_{12}^2 & V_{11}V_{21} + V_{12}V_{22} \\ V_{11}V_{11} + V_{12}V_{22} & V_{21}^2 + V_{22}^2 \end{pmatrix}$$

Yes, this relation is also true for other distributions for X, since we didn't use the Gaussian assumption in this proof.

Note that the result of this part means that if we used the second characterization to write a jointly Gaussian distribution (using $Z = AX$), we can get the covariance matrix by just computing AA^T.

(d) Given a jointly Gaussian zero mean RV $Z = [Z_1 \ Z_2]^\top \in \mathbb{R}^2$ with covariance matrix $\Sigma_Z = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{bmatrix}$, derive the conditional distribution of $(Z_1|Z_2 = z)$.

Hint: The following identity may be useful

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{b}{c} & 1 \end{bmatrix} \begin{bmatrix} \left(a - \frac{b^2}{c}\right)^{-1} & 0 \\ 0 & \frac{1}{c} \end{bmatrix} \begin{bmatrix} 1 & -\frac{b}{c} \\ 0 & 1 \end{bmatrix}$$

Solution:

One can do this from first principles, by manipulating the densities themselves. However, we will show a linear algebraic method to derive the density. Using the hint, we begin by writing

$$\Sigma^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1 \end{bmatrix} \begin{bmatrix} \left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix} \begin{bmatrix} 1 & -\frac{\Sigma_{12}}{\Sigma_{22}} \\ 0 & 1 \end{bmatrix}$$

We can now plug this into the density function. Recall that

$$\begin{aligned} f_{Z_1,Z_2}(z_1, z_2) &\propto \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 & z_2 \end{bmatrix} \Sigma^{-1} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) \\ &= \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{\Sigma_{12}}{\Sigma_{22}} & 1 \end{bmatrix} \begin{bmatrix} \left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix} \begin{bmatrix} 1 & -\frac{\Sigma_{12}}{\Sigma_{22}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) \\ &= \exp\left(-\frac{1}{2}\begin{bmatrix} z_1 - \frac{\Sigma_{12}}{\Sigma_{22}} z_2 & z_2 \end{bmatrix} \begin{bmatrix} \left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1} & 0 \\ 0 & \frac{1}{\Sigma_{22}} \end{bmatrix} \begin{bmatrix} z_1 - \frac{\Sigma_{12}}{\Sigma_{22}} z_2 \\ z_2 \end{bmatrix}\right) \end{aligned}$$

Now see that since the square matrix is diagonal, our density decomposes to yield

$$f_{Z_1,Z_2}(z_1, z_2) \propto \exp\left(-\frac{1}{2}\left(z_1 - \frac{\Sigma_{12}}{\Sigma_{22}} z_2\right)^2 \left(\Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)^{-1}\right) \exp\left(-\frac{1}{2\Sigma_{22}} z_2^2\right)$$

Thus, conditional on $Z_2 = z_2$, we see that $(Z_1|Z_2 = z_2) \sim N\left(\frac{\Sigma_{12}}{\Sigma_{22}} z_2, \Sigma_{11} - \frac{\Sigma_{12}^2}{\Sigma_{22}}\right)$.

2 Projections and Linear Regression

We are given $X \in \mathbb{R}^{n \times d}$ where $n > d$ and $\text{rank}(X) = d$. We are also given a vector $y \in \mathbb{R}^n$. Define the orthogonal projection of y onto $\text{range}(X)$ as $P_{\text{range}(X)}(y)$.

Background on orthogonal projections For any finite-dimensional subspace W (here, $\text{range}(X)$) of a vector space V (here, $\mathbb{R}^n$), any vector $v \in V$ can be decomposed as

$$v = w + u, \qquad w \in W, \quad u \in W^\perp,$$

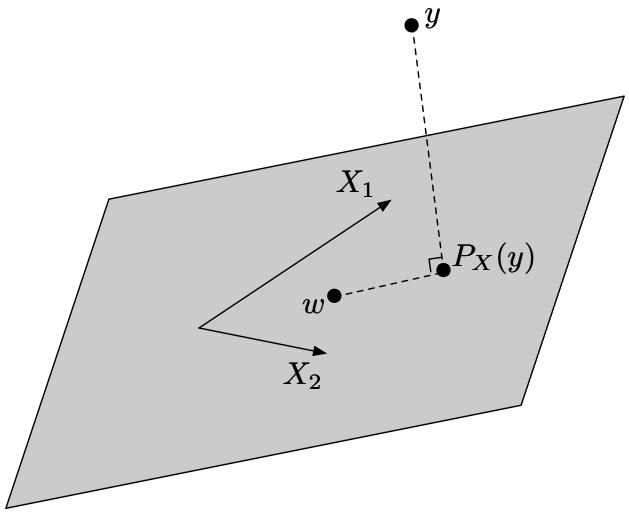
where $W^\perp$ is the orthogonal complement of W. Furthermore, this decomposition is unique: if $v = w' + u'$ where $w' \in W$, $u' \in W^\perp$, then $w' = w$ and $u' = u$. These two facts allow us to define P_W, the orthogonal projection operator onto W. Given a vector v with decomposition $v = w + u$, we define

$$P_W(v) = w.$$

It can also be shown using these two facts that P_W is linear. For more information on orthogonal projections, see `https://gwthomas.github.io/docs/math4ml.pdf`.

(a) Prove that $P_{\text{range}(X)}(y) = \underset{w \in \text{range}(X)}{\text{argmin}} \|y - w\|_2^2$.

Solution:



Let w be in the range of X. Then, by the Pythagorean theorem,

$$\begin{aligned}\left\|y - w\right\|_2^2 &= \left\|y - P_{\text{range}(X)}(y) + P_{\text{range}(X)}(y) - w\right\|_2^2 \\ &= \left\|y - P_{\text{range}(X)}(y)\right\|_2^2 + \left\|P_{\text{range}(X)}(y) - w\right\|_2^2 + 2(y - P_{\text{range}(X)}(y))^T(P_{\text{range}(X)}(y) - w)\end{aligned}$$

Note that $y - P_{\text{range}(X)}(y)$ is perpendicular to the range of X, while $P_{\text{range}(X)}(y) - w$ is in the range of X. Therefore the third term in the above sum equals 0. This leaves us with

$$\left\|y - w\right\|_2^2 = \left\|y - P_{\text{range}(X)}(y)\right\|_2^2 + \left\|P_{\text{range}(X)}(y) - w\right\|_2^2$$

The above is clearly minimized when $w^* = P_{\text{range}(X)}(y)$. Therefore, we get that $P_{\text{range}(X)}(y)$ minimizes $\left\|y - w\right\|_2^2$.

(b) An orthogonal projection is a linear transformation. That is, $P_{\text{range}(X)}(y) = Py$ for some projection matrix P.

Specifically, given $1 \leq d \leq n$, a matrix $P \in \mathbb{R}^{n \times n}$ is said to be a rank-d orthogonal projection matrix if

- $\text{rank}(P) = d$
- $P = P^T$
- $P^2 = P$.

Prove that P is a rank-d projection matrix if and only if there exists a $U \in \mathbb{R}^{n \times d}$ such that $P = UU^\top$ and $U^\top U = I$.

Hint Use the eigendecomposition of P to prove the forward direction.

Solution:

First, we prove that if P is a rank-d projection matrix, then there exists a U such that $P = UU^T$ and $U^T U = I$. Since P is symmetric, we can apply the eigendecomposition to get that $P = V\Lambda V^T$ for some orthogonal $V \in \mathbb{R}^{n \times n}$ and some diagonal $\Lambda \in \mathbb{R}^{n \times n}$. We argue that all eigenvalues of P (and therefore the diagonal elements of Λ) are binary-valued (i.e. 0 or 1). To see this, let v be an eigenvector of P with corresponding eigenvalue λ. Then,

$$Pv = \lambda v \implies P^2 v = P(\lambda v) \implies P^2 v = \lambda^2 v \implies Pv = \lambda^2 v$$

Therefore we have that $\lambda = \lambda^2$, necessitating that $\lambda \in \{0, 1\}$. Hence, the diagonals of Λ are binary-valued.

Now, let U denote the matrix whose columns are the columns of V that correspond to indices i for which $\Lambda_{ii} = 1$ (and not 0). For such a U, $P = V\Lambda V^T = UU^T$. We claim that this $U \in \mathbb{R}^{n \times d}$ and $U^T U = I$. Since P has rank d, there are d such 1-valued elements along the diagonal of Λ. This means that U has shape $n \times d$. Moreover, since the columns of U are a subset of those of V, they are orthonormal, implying $U^T U = I$.

Second, we prove the backward direction: if there exists some U such that $P = UU^T$ and $U^T U = I$, P satisfies all three properties of being a rank-d projection matrix.

- $P^T = (UU^T)^T = UU^T = P$
- $P^2 = UU^T UU^T = UU^T = P$

- Before proving that $\text{rank}(P) = d$, we prove the more general statement that $\text{rank}(A^T) = \text{rank}(AA^T)$ for any A. First, we show that $\mathcal{N}(AA^T) \subseteq \mathcal{N}(A^T)$:

$$AA^T x = 0 \implies x^T AA^T x = 0 \implies \left\|A^T x\right\|_2^2 = 0 \implies A^T x = 0.$$

Now, we show that $\mathcal{N}(A^T) \subseteq \mathcal{N}(AA^T)$:

$$A^T x = 0 \implies AA^T x = 0.$$

Therefore, $\mathcal{N}(A^T) = \mathcal{N}(AA^T)$. By the rank-nullity theorem, this implies that $\text{rank}(A^T) = \text{rank}(AA^T)$.

Going back to the original problem, we have $\text{rank}(P) = \text{rank}(UU^T) = \text{rank}(U^T)$. Because U^T has d orthonormal rows as $U^T U = I$, $\text{rank}(U^T) = d$.

(c) The Singular Value Decomposition theorem states that we can write any matrix X as

$$X = \sum_{i=1}^{\min\{n,d\}} \sigma_i u_i v_i^\top = \sum_{i:\sigma_i > 0} \sigma_i u_i v_i^\top$$

where $\sigma_i \geq 0$, and $\{u_i\}_{i=1}^n$ and $\{v_i\}_{i=1}^d$ are orthonormal bases for $\mathbb{R}^n$ and $\mathbb{R}^d$ respectively. Some of the singular values σ_i may equal 0, indicating that the associated left and right singular vectors u_i and v_i do not contribute to the sum, but sometimes it is still convenient to include them in the SVD so we have complete orthonormal bases for $\mathbb{R}^n$ and $\mathbb{R}^d$ to work with. Show that

(i) $\{u_i : \sigma_i > 0\}$ is an orthonormal basis for the columnspace of X

(ii) Similarly, $\{v_i : \sigma_i > 0\}$ is an orthonormal basis for the row space of of X
Hint: consider $X^\top$.

Solution: Since $\{u_i : \sigma_i > 0\}$ are orthonormal, it suffices to show that their span is the column space of X. The column space of X is defined as $\{Xw : w \in \mathbb{R}^d\}$. Note that

$$Xw = \left(\sum_{i:\sigma_i>0} \sigma_i u_i v_i^\top\right) w = \sum_{i:\sigma_i>0} \sigma_i u_i v_i^\top w = \sum_{i:\sigma_i>0} (\sigma_i v_i^\top w) u_i$$

Since $\sigma_i v_i^\top w \in \mathbb{R}$, it follows that any Xw can be written as a linear combination of $\{u_i : \sigma_i > 0\}$, as desired.

Since $\{v_i : \sigma_i > 0\}$ are orthonormal, it suffices to show that their span is the row space of X. The row space of X is defined as $\{X^\top w : w \in \mathbb{R}^n\}$, i.e., column space of $X^\top$. Note that

$$X^\top w = \left(\sum_{i:\sigma_i>0} \sigma_i u_i v_i^\top\right)^\top w = \left(\sum_{i:\sigma_i>0} \sigma_i v_i u_i^\top\right) w = \sum_{i:\sigma_i>0} \sigma_i v_i u_i^\top w = \sum_{i:\sigma_i>0} (\sigma_i u_i^\top w) v_i$$

Since $\sigma_i u_i^\top w \in \mathbb{R}$, it follows that any $X^\top w$ can be written as a linear combination of $\{v_i : \sigma_i > 0\}$, as desired.

(d) Let $X \in \mathbb{R}^{n \times d}$ such that $\text{rank}(X) = d$. Prove that $X(X^TX)^{-1}X^T$ is a rank-d orthogonal projection matrix.

Hint: Consider the SVD decomposition of X.

Solution: Using the statement proved in part (b), we will prove that $X(X^TX)^{-1}X^T$ equals UU^T for some $U \in \mathbb{R}^{n \times d}$ such that $U^TU = I$.

Let $X = U\Sigma V^\top$ denote the SVD of X, with $U \in \mathbb{R}^{n \times d}$, and $\Sigma \in \mathbb{R}^{d \times d}$, and $V \in \mathbb{R}^{d \times d}$. Then, $X^\top X = V\Sigma^2 V^\top$, and since $\text{rank}(X) = d$, Σ^2 is invertible, with $(X^\top X)^{-1} = V^\top \Sigma^{-2} V$. Hence,

$$X(X^\top X)^{-1}X^\top = U\Sigma V^\top (V\Sigma^{-2}V^\top)V\Sigma U^\top = U\Sigma\Sigma^{-2}\Sigma U^\top = UU^\top,$$

which shows that $X(X^\top X)^{-1}X^\top$ is the projection matrix $UU^\top$, where U is the left singular-vectors matrix of X. Since the left singular vectors of a matrix are orthogonal and $n > d$, it also follows that $U^TU = I$.

(e) Prove that $X(X^TX)^{-1}X^T$ is a projection onto $\text{range}(X)$.

Solution: From part (d), we know that $X(X^TX)^{-1}X^T = UU^T$ where U is the matrix of left singular vectors. Because $\text{range}(UU^T) = \text{range}(U) = \text{range}(X)$ where the first equality we know from part (b) and the second equality we know from part (c), we can deduce that $X(X^TX)^{-1}X^T$ projects all vectors onto $\text{range}(X)$.

(f) Show that $w^* = (X^TX)^{-1}X^Ty$ is the solution to the optimization problem

$$\underset{w}{\text{argmin}} \left\|y - Xw\right\|_2^2$$

using only facts proved in this problem.

Solution: We can re-express $\min_w \left\|y - Xw\right\|_2^2$ as $\min_{z \in \text{range}(X)} \left\|y - z\right\|_2^2$. From parts (a) and (e), we know that $z^* = P_{\text{range}(X)}(y) = X(X^TX)^{-1}X^Ty$. This entails that $w^* = (X^TX)^{-1}X^Ty$.

3 Some MLEs

For this question, assume you observe n (data point, label) pairs $(x_i, y_i)_{i=1}^n$, with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for all $i = 1, \ldots, n$. We denote X as the data matrix containing all the data points and y as the label vector containing all the labels:

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}, \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n.$$

(a) Ignoring y for now, suppose we model the data points as coming from a d-dimensional Gaussian with diagonal covariance:

$$\forall i = 1, \ldots, n, \quad x_i \overset{i.i.d.}{\sim} N(\mu, \Sigma); \quad \Sigma = \begin{bmatrix} \sigma_1^2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_d^2 \end{bmatrix}.$$

If we consider $\mu \in \mathbb{R}^d$ and $(\sigma_1^2, \ldots, \sigma_d^2)$, where each $\sigma_i^2 > 0$, to be unknown, the parameter space here is $2d$-dimensional. When we refer to Σ as a parameter, we are referring to the d-tuple $(\sigma_1^2, \ldots, \sigma_d^2)$, but inside a linear algebraic expression, Σ denotes the diagonal matrix $\text{diag}(\sigma_1^2, \ldots, \sigma_d^2)$.

Solve the following problems:

(i) Prove that log-likelihood $\ell(\mu, \Sigma) = \log p(X \mid \mu, \Sigma)$ is equal to

$$-\frac{n}{2}\left(d \log(2\pi) - \sum_{j=1}^{d} \log\left(\frac{1}{\sigma_j^2}\right)\right) - \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu).$$

(ii) Find the MLE of μ assuming Σ is known.

(iii) Find the MLE of Σ assuming μ is known.
Hint: you can re-parameterize σ_j^2 by defining $v_j = \frac{1}{\sigma_j^2}$

(iv) Find the joint MLE of (μ, Σ) in terms of the maximum likelihood estimates computed above.

Solution: i) The log likelihood is

$$\begin{aligned} \log p(X \mid \mu, \Sigma) &= \log \prod_{i=1}^{n} p(x_i \mid \mu, \Sigma) \\ &= \log \prod_{i=1}^{n} (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu)\right\} \\ &= \sum_{i=1}^{n} \log(2\pi)^{-d/2} + \log(|\Sigma|)^{-1/2} - \frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu) \\
&= -\frac{n}{2}\Big(d\log(2\pi) + \log(|\Sigma|)\Big) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu) \\
&= -\frac{n}{2}\left(d\log(2\pi) + \log\left(\prod_{j=1}^{d}\sigma_j^2\right)\right) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu) \\
&= -\frac{n}{2}\left(d\log(2\pi) + \sum_{j=1}^{d}\log\left(\sigma_j^2\right)\right) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu) \\
&= -\frac{n}{2}\left(d\log(2\pi) - \sum_{j=1}^{d}\log\left(\frac{1}{\sigma_j^2}\right)\right) - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)
\end{aligned}
$$

ii) To compute the MLE of μ, we set the gradient equal to 0:

$$\nabla_\mu \log p(X\,|\,\mu,\Sigma) = -\sum_{i=1}^{n}\Sigma^{-1}(\mu - x_i) = 0.$$

Solving for μ, we get the stationary point $\mu^* = \frac{1}{n}\sum_{i=1}^{n} x_i$, which is the sample mean. To see that μ^* is the maximizer, we show the Hessian of the log-likelihood is negative definite.

$$\nabla_\mu^2 \log p(X\,|\,\mu,\sigma^2) = -\Sigma^{-1} \prec 0.$$

Thus, μ^* maximizes the log-likelihood, and therefore the likelihood, because log is strictly increasing.

iii) To find the MLE of Σ, we compute the MLE of each σ_j^2 (we will see they are independent of each other). Furthermore, we reparameterize σ_j^2 with $\nu_j = \frac{1}{\sigma_j^2}$.

$$
\begin{aligned}
\log p(X\,|\,\mu,\nu) &= -\frac{n}{2}\left(d\log(2\pi) - \sum_{i=1}^{d}\log(\nu_i)\right) - \sum_{i=1}^{n}\sum_{j=1}^{d}\frac{\nu_j}{2}(x_{ij}-\mu_j)^2, \\
\frac{\partial}{\partial\nu_j}\log p(X\,|\,\mu,\nu) &= \frac{n}{2\nu_j} - \frac{1}{2}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2, \\
\nabla_\nu^2 \log p(X\,|\,\mu,\nu) &= \operatorname{diag}\left(-\frac{n}{2\nu_1^2},\ldots,-\frac{n}{2\nu_d^2}\right) \prec 0
\end{aligned}
$$

This returns the MLE $\nu_j^*(\mu) = \frac{n}{\sum_{i=1}^{n}(x_{ij}-\mu_j)^2}$ for $j = 1,\ldots,d$. We know that $(\nu_1^*(\mu),\ldots,\nu_d^*(\mu))$ is a maximizer because the Hessian is negative definite. Because ν_j corresponds one to one with σ_j^2,

$$\forall j = 1,\ldots,d, \quad (\sigma_j^2)^*(\mu) = \frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2$$

maximizes the log-likelihood. (Note the second derivative with respect to a σ_j^2 is not, in fact, always negative, so further justification is needed if you derived with respect to σ_j^2).

iv) The joint MLE is simply $(\mu^*, (\Sigma)^*(\mu^*))$, where $\Sigma^*(\mu^*)_{jj} = \sigma_j^*(\mu^*)$; we first maximize over μ, and then maximize over Σ. This is valid because $\max_{\mu,\Sigma}(\ldots) = \max_{\Sigma} \max_{\mu}(\ldots)$.

(b) Suppose that we have a training set $\{(x_i, y_i) \mid i = 1 \ldots n\}$ of n independent examples but in which the residual terms had different variances. That is, we assume

$$y_i \sim N(w^T x_i, \sigma_i^2).$$

Show that the MLE estimate of w can be found by solving the following optimization problem

$$w_{\text{MLE}} = \underset{w}{\text{argmin}} \left\| A(Xw - y) \right\|_2^2.$$

Clearly state what the matrix A equals.

Solution:

$$\begin{aligned}
w_{\text{MLE}} &= \underset{w}{\text{argmax}} \prod_{i=1}^{n} p(y_i \mid w^T x_i, \sigma_i^2) \\
&= \underset{w}{\text{argmax}} \prod_{i=1}^{n} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left\{ -\frac{(y_i - w^T x_i)^2}{2\sigma_i^2} \right\} \\
&= \underset{w}{\text{argmax}} \sum_{i=1}^{n} \log \frac{1}{\sigma_i \sqrt{2\pi}} - \frac{(y_i - w^T x_i)^2}{2\sigma_i^2} \\
&= \underset{w}{\text{argmin}} \sum_{i=1}^{n} \frac{(y_i - w^T x_i)^2}{\sigma_i^2} \\
&= \underset{w}{\text{argmin}} \, (Xw - y)^T \Sigma^{-1} (Xw - y)
\end{aligned}$$

where $\Sigma^{-1} = \text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2)$. Because we can take the positive square root of all $1/\sigma_i^2$ diagonal terms, we can compute $\Sigma^{-1/2}$. This allows us to refactor the above minimization problem into the desired form:

$$\begin{aligned}
w_{\text{MLE}} &= \underset{w}{\text{argmin}} \, (Xw - y)^T \Sigma^{-1/2} \Sigma^{-1/2} (Xw - y) \\
&= \left\| \Sigma^{-1/2} (Xw - y) \right\|_2^2
\end{aligned}$$

Therefore, $A = \text{diag}(1/\sigma_1, \ldots, 1/\sigma_n)$.

(c) Consider the Categorical$(\theta_1, \theta_2, \ldots, \theta_k)$ distribution. Recall, for categorical distributions, there are two constraints on θ_k:

- $\theta_k \geq 0$ for all k
- $\sum_{k=1}^{K} \theta_k = 1$

The distribution describes a random process that selects one of the K possible categories, with category k being chosen with probability θ_k.

Ignoring the data points X, suppose that for all i from 1 to n, we sample y_i from a categorical distribution:

$$y_i \overset{i.i.d.}{\sim} \text{Categorical}(\theta_1, \ldots, \theta_K).$$

Compute the MLE of $\theta = (\theta_1, \ldots, \theta_K)$. Use the fact that the KL divergence is nonnegative:

$$\text{KL}(\pi \,||\, \theta) = \sum_{\omega \in \Omega} \pi(\omega) \log\left(\frac{\pi(\omega)}{\theta(\omega)}\right) \geq 0.$$

Solution: The log likelihood is

$$\log p(y \,|\, \theta) = \log\left[\prod_{i=1}^{n} \prod_{k=1}^{K} \theta_k^{\mathbf{1}\{y_i = k\}}\right] = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbf{1}\{y_i = k\} \log \theta_k = \sum_{k=1}^{K} n_k \log \theta_k.$$

We have defined $n_k = \sum_{k=1}^{K} \mathbf{1}\{y_i = k\}$ as the number of outcomes that equal k. Note $\hat{\theta} = \left(\frac{n_1}{n}, \ldots, \frac{n_K}{n}\right)$ defines a probability distribution, specifically the Categorical we get by replacing each probability with the empirical probability of that category. Now

$$\text{KL}(\hat{\theta} \,||\, \theta) \geq 0 \iff \sum_{k=1}^{K} \hat{\theta}_k \log\left(\frac{\hat{\theta}_k}{\theta_k}\right) \geq 0 \iff \sum_{k=1}^{K} \hat{\theta}_k \log \hat{\theta}_k \geq \sum_{k=1}^{K} \hat{\theta}_k \log \theta_k$$

$$\iff \sum_{k=1}^{K} \frac{n_k}{n} \log\left(\frac{n_k}{n}\right) \geq \sum_{k=1}^{K} \frac{n_k}{n} \log(\theta) \iff \sum_{k=1}^{K} n_k \log\left(\frac{n_k}{n}\right) \geq \sum_{k=1}^{K} n_k \log(\theta).$$

Thus, $\log p(y \,|\, \hat{\theta}) \geq \log p(y \,|\, \theta)$ for any θ. Thus, $\hat{\theta} = \left(\frac{n_1}{n}, \ldots, \frac{n_K}{n}\right)$ is the maximizer.

(d) Again consider X fixed. This time, we suppose that each y_i is binary-valued (0 or 1). We choose to model y as

$$y_i \overset{ind.}{\sim} \text{Ber}(s(x_i^\top w)) \quad \forall i = 1, \ldots, n,$$

where $s(z) = \frac{1}{1+e^{-z}}$ is the *sigmoid* function and $\text{Ber}(p)$ denotes the Bernoulli distribution which takes value 1 with probability p and 0 with probability $1 - p$.

(i) Write down the log-likelihood $\ell(w) = \log p(y \,|\, w)$ and show that finding the MLE of w is equivalent to minimizing the cross entropy between $\text{Ber}(y_i)$ and $\text{Ber}(s(x_i^\top w))$ for each i:

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} H(\text{Ber}(y_i), \text{Ber}(s(x_i^\top w))). \tag{1}$$

Definition of cross entropy: given two discrete probability distributions $\pi : \Omega \to [0, 1]$ and $\theta : \Omega \to [0, 1]$ on some outcome space Ω, we define the cross entropy between π and θ as

$$H(\pi, \theta) = \sum_{\omega \in \Omega} -\pi(\omega) \log \theta(\omega).$$

Solution:

$$\begin{aligned}\log p(y \mid w) &= \log\left(\prod_{i=1}^{n} s(x_i^\top w)^{y_i}(1 - s(x_i^\top w))^{1-y_i}\right) \\ &= \sum_{i=1}^{n} y_i \log s(x_i^\top w) + (1 - y_i)\log\Big(1 - s(x_i^\top w)\Big) \\ &= -\sum_{i=1}^{n} H(\text{Ber}(y_i), \text{Ber}(s(x_i^\top w))).\end{aligned}$$

Maximizing the sum of the negative cross entropies is equivalent to minimizing the sum of the cross entropies.

(ii) Show that (1) (and therefore finding the MLE) is equivalent to the following problem:

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \log\Big(1 + \exp\big(-z_i x_i^\top w\big)\Big) \tag{2}$$

where $z_i = 1$ if $y_i = 1$ and $z_i = -1$ if $y_i = 0$.
Note: both (1) and (2) are referred to as logistic regression.

Solution: Let's first look at a single term

$$\begin{aligned}H(\text{Ber}(y_i), \text{Ber}(s(x_i^\top w))) &= -y_i \log\Big(s(x_i^\top w)\Big) - (1 - y_i)\log\Big(1 - s(x_i^\top w)\Big) \\ &= -y_i \log\left(\frac{1}{1 + e^{-x_i^\top w}}\right) - (1 - y_i)\log\left(1 - \frac{1}{1 + e^{-x_i^\top w}}\right) \\ &= -y_i \log\left(\frac{1}{1 + e^{-x_i^\top w}}\right) - (1 - y_i)\log\left(\frac{e^{-x_i^\top w}}{1 + e^{-x_i^\top w}}\right) \\ &= -y_i \log\left(\frac{1}{1 + e^{-x_i^\top w}}\right) - (1 - y_i)\log\left(\frac{1}{1 + e^{x_i^\top w}}\right) \\ &= y_i \log\left(1 + e^{-x_i^\top w}\right) + (1 - y_i)\log\left(1 + e^{x_i^\top w}\right) \\ &= \log\left(1 + e^{-z_i x_i^\top w}\right).\end{aligned}$$

The last equality follows from casework: if $y_i = 1$, then $z_i = 1$ and the term equals $\log\Big(1 + e^{-x_i^\top w}\Big) = \log\Big(1 + e^{-z_i x_i^\top w}\Big)$; if $y_i = 0$, then $z_i = -1$ and the term equals $\log\Big(1 + e^{x_i^\top w}\Big) = \log\Big(1 + e^{-z_i x_i^\top w}\Big)$. Thus,

$$\min_w \sum_{i=1}^{n} H(\text{Ber}(y_i), \text{Ber}(s(x_i^\top w))) = \min_w \sum_{i=1}^{n} \log\left(1 + e^{-z_i x_i^\top w}\right).$$

(iii) Let $J(w) = \log\Big(1 + \exp(-z x^\top w)\Big)$ where, again, $z = 1$ if $y = 1$ and $z = -1$ if $y = 0$ (we are only considering a single (x, y) pair in this subpart). Prove the following:

i. J is not strictly convex.
Hint: A necessary condition for a twice-differentiable function to be strictly convex is that its Hessian is positive definite.

ii. The gradient descent update rule for minimizing $J(w)$ with learning rate ϵ is

$$w' = w - \epsilon\left(\frac{1}{1+e^{-x^T w}} - y\right)x$$

Solution: Following subpart (ii), let us rewrite $J(w)$ as

$$-y\log\left(\frac{1}{1+e^{-x^\top w}}\right) - (1-y)\log\left(\frac{e^{-x^\top w}}{1+e^{-x^\top w}}\right)$$

Setting $s = s(x^\top w)$, this is

$$-y\log(s) - (1-y)\log(1-s)$$

From lecture, we know that

$$\nabla_w s = \frac{\exp(-x^\top w)}{(1+\exp(-x^\top w))^2}x = s(1-s)x$$

Then, applying the chain rule, we see that

$$\begin{aligned}
\nabla_w J &= -\frac{y}{s}(\nabla_w s) + \frac{1-y}{1-s}(\nabla_w s) \\
&= \left[-\frac{y}{s}s(1-s)\right]x + \left[\frac{1-y}{1-s}s(1-s)\right]x \\
&= [-y(1-s) + (1-y)s]x \\
&= (s-y)x
\end{aligned}$$

The Hessian is just the derivative of the gradient, i.e.

$$\begin{aligned}
\nabla_w^2 J &= \frac{\mathrm{d}}{\mathrm{d}w}((s-y)x) \\
&= x\frac{\mathrm{d}s}{\mathrm{d}w} \\
&= x(s(1-s)x^\top) \\
&= s(1-s)xx^\top
\end{aligned}$$

Since $s \in (0,1)$, it follows that $s(1-s) > 0$. Now, we check for the definiteness of $\nabla^2 J(w)$:

- Suppose $v \in \text{Span}(x)$, i.e., $v = \alpha x$ for some $\alpha \in \mathbb{R}$. Then,

$$v^\top[\nabla_w^2 J]v = s(1-s)v^\top xx^\top v = s(1-s)(x^\top v)^2 = s(1-s)\alpha^2(x^\top x)^2 > 0$$

- Suppose $v \in \text{Span}^\perp(x)$, i.e., v is orthogonal to x. Then,

$$v^\top[\nabla_w^2 J]v = s(1-s)v^\top xx^\top v = s(1-s)(x^\top v)^2 = s(1-s)(0)^2 = 0$$

Thus, $\nabla_w^2 J$ is only PSD, and not PD, which implies that $J(w)$ is convex but not strictly convex. This means that the minimizer w^* of

$$\min_{w \in \mathbb{R}^d} \log\left(1 + \exp\left(-zx^\top w\right)\right)$$

is not necessarily unique.

The gradient update rule for minimizing $J(w)$ is

$$\begin{aligned} w' &= w - \epsilon \cdot \nabla_w J(w) \\ w' &= w - \epsilon(s - y)x \\ w' &= w - \epsilon\left(\frac{1}{1 + e^{-x^\top w}} - y\right)x \end{aligned}$$

4 Geometry of Ridge Regression

You recently learned ridge regression and how it differs from ordinary least squares. In this question we will explore the properties of ridge regression in more depth. Recall that the ridge regression problem is given by the following optimization problem:

$$\min_{w} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \nu\|\mathbf{w}\|_2^2. \tag{3}$$

The solution to ridge regression is given by

$$\hat{\mathbf{w}}_\mathbf{r} = (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}. \tag{4}$$

(a) One reason why we might want to have small weights $\mathbf{w}$ has to do with the sensitivity of the predictor to its input. Let $\mathbf{x}$ be a d-dimensional list of features corresponding to a new test point. Our predictor is $\mathbf{w}^\top\mathbf{x}$. What is an upper bound on how much our prediction could change if we added noise $\boldsymbol{\epsilon} \in \mathbb{R}^d$ to a test point's features $\mathbf{x}$, in terms of $\|\mathbf{w}\|_2$ and $\|\boldsymbol{\epsilon}\|_2$?

Hint: Use the Cauchy-Schwarz inequality.

Solution: The change in prediction is given by

$$|\mathbf{w}^\top(\mathbf{x} + \boldsymbol{\epsilon}) - \mathbf{w}^\top\mathbf{x}| = |\mathbf{w}^\top\boldsymbol{\epsilon}| \leq \|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2 \tag{5}$$

where the second step is a result of the Cauchy-Schwarz inequality. To understand the intuition as to why this is true, you can divide both sides by $\|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2$ which gives you

$$\left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|_2}, \frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|_2} \right\rangle \leq 1$$

In words, this inequality is saying that if you project a unit vector $\frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|_2}$ onto some direction $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$, the resulting length must be less than or equal to 1. The prediction will thus vary from the original prediction within a distance of at most $\|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2$.

(b) Note that in computing $\hat{\mathbf{w}}_\mathbf{r}$, we are trying to invert the matrix $\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I}$ instead of the matrix $\mathbf{X}^\top\mathbf{X}$. If $\mathbf{X}^\top\mathbf{X}$ has eigenvalues $\sigma_1^2, \ldots, \sigma_d^2$, what are the eigenvalues of $(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}$? Comment on why adding the regularizer term $\nu\mathbf{I}$ can improve the inversion operation numerically.

Solution: The eigenvalues of $(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}$ are given by $(\sigma_1^2 + \nu)^{-1}, (\sigma_2^2 + \nu)^{-1}, \cdots, (\sigma_d^2 + \nu)^{-1}$.

In order to see this, note that if v_i is an eigenvector of $\mathbf{X}^\top\mathbf{X}$ with eigenvalue σ_i^2, then we have

$$(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})v_i = \sigma_i^2 v_i + \nu v_i = (\sigma_i^2 + \nu)v_i.$$

We also know that the eigenvalues of the inverse of a matrix are the inverses of the eigenvalues, therefore the eigenvalues of $(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}$ are $(\sigma_i^2 + \nu)^{-1}$.

Notice that since $\sigma_i^2 \geq 0$ the matrix $\mathbf{X}^\top\mathbf{X}$ is positive semi-definite, but some eigenvalues may be very close to 0. Consequently, small eigenvalues can lead to numerical instability, since the inverse blows these up. By adding a regularizer, we increase the values of the eigenvalues of the original matrix above a threshold ν, and thus improve the inversion operation.

(c) Let the number of parameters $d = 4$ and the number of datapoints $n = 6$, and let the eigenvalues of $\mathbf{X}^\top\mathbf{X}$ be given by 500, 10, 1, and 0.001. We must now choose between two regularization parameters $\nu_1 = 50$ and $\nu_2 = 0.1$. Which do you think is a better choice for this problem and why?

Solution: $\nu = 0.1$ is a better option. We are looking for a regularization constant for better numerical conditioning without changing the problem substantially.

Notice that the eigenvalue 0.001 could cause us numerical issues as noted above, which we would like to eliminate. We would therefore like to preserve the relative size of the original eigenvalues. By choosing $\nu = 50$, we also eliminate the effect of the eigenvalue 1, thereby altering our problem unnecessarily.

(d) Another advantage of ridge regression can be seen for under-determined systems. Say we have the data drawn from a $d = 5$ parameter model, but only have $n = 4$ training samples of it, i.e. $\mathbf{X} \in \mathbb{R}^{4\times 5}$. Now this is clearly an underdetermined system, since $n < d$. Show that ridge regression with $\nu > 0$ results in a unique solution, whereas ordinary least squares has an infinite number of solutions.

Hint: To make this point, it may be helpful to consider $\mathbf{w} = \mathbf{w}_0 + \mathbf{w}^*$ where $\mathbf{w}_0$ is in the null space of $\mathbf{X}$ and $\mathbf{w}^*$ is a solution.

Solution: First, we show that ridge regression always leads to a unique solution. We know that the minimizer is given by

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}.$$

We also know that the eigenvalues of the matrix $\mathbf{X}^\top\mathbf{X} + \nu\mathbf{I}$ are all at least ν, and so the matrix is invertible, thus leading to a unique solution.

For ordinary least squares, let us assume that $\mathbf{w}'$ minimizes $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$, i.e., $\|\mathbf{X}\mathbf{w}' - \mathbf{y}\|_2 \leq \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$ for all $\mathbf{w} \in \mathbb{R}^d$.

Since $d > n$, the matrix $\mathbf{X}$ has a non-trivial nullspace. We can take any vector $\mathbf{w_0}$ in the nullspace of $\mathbf{X}$ and consider the new vector $\mathbf{w}''(\alpha) = \mathbf{w}' + \alpha\mathbf{w_0}$.

Notice that $\|\mathbf{X}\mathbf{w}''(\alpha) - \mathbf{y}\|_2 = \|\mathbf{X}\mathbf{w}' - \mathbf{y}\|_2 \leq \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$ for all $\mathbf{w} \in \mathbb{R}^d$ because $\mathbf{w_0}$ is in the null space of $\mathbf{X}$. Thus, the vector $\mathbf{w}''(\alpha)$ is a minimizer for any choice of α. We have thus shown that the least squares problem has an infinite number of solutions.

(e) What will the solution to ridge regression (4) converge to if you take the limit $\nu \to 0$? Your answer should be a simple expression in terms of $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}, \mathbf{y}$, and ν where $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the SVD of $\mathbf{X}$.

Solution: Plugging the singular value decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ into the ridge regression solution, we get

$$\begin{aligned}
\mathbf{w}^*(\nu) &= (\mathbf{X}^T\mathbf{X} + \nu\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \\
&= (\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T + \nu\mathbf{I})^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y} \\
&= (\mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T + \nu\mathbf{I})^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}
\end{aligned}$$

$$
\begin{aligned}
&= [\mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma} + \nu\mathbf{I})\mathbf{V}^T]^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y} \\
&= (\mathbf{V}^T)^{-1}(\mathbf{\Sigma}^T\mathbf{\Sigma} + \nu\mathbf{I})^{-1}\mathbf{V}^{-1}\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T y \\
&= \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma} + \nu\mathbf{I})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}
\end{aligned}
$$

and for $\nu \to 0$ this converges to the unique solution $\mathbf{w}^* = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{y}$, also called the *minimum norm solution*. (Notice here that $\mathbf{\Sigma}^{-1}$ has the obvious definition of just involving the relevant square matrix of nonzero singular values.)

This is also closely connected to the idea of the Moore-Penrose inverse. This problem continues the trend of illustrating the utility of the SVD while reasoning about such problems. It really is a workhorse of machine learning reasoning.

(f) Tikhonov regularization is a general term for ridge regression, where the implicit constraint set takes the form of an ellipsoid instead of a ball. In other words, we solve the optimization problem

$$\mathbf{w} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \nu\|\mathbf{\Gamma}\mathbf{w}\|_2^2$$

for some full rank matrix $\mathbf{\Gamma} \in \mathbb{R}^{d\times d}$. Derive a closed form solution for $\mathbf{w}$.

Solution: The objective is

$$
\begin{aligned}
f(\mathbf{w}) &= \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \nu\|\mathbf{\Gamma}\mathbf{w}\|_2^2 \\
&= \mathbf{w}^\top(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{\Gamma}^\top\mathbf{\Gamma})\mathbf{w} - \mathbf{y}^\top\mathbf{X}\mathbf{w} + \mathbf{y}^\top\mathbf{y}
\end{aligned}
$$

Setting the gradient with respect to $\mathbf{w}$ to zero, we obtain

$$\mathbf{0} = \mathbf{w}^\top(\mathbf{X}^\top\mathbf{X} + \nu\mathbf{\Gamma}^\top\mathbf{\Gamma}) - \mathbf{y}^\top\mathbf{X}.$$

Thus, the solution is now given by $\mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \nu\mathbf{\Gamma}^\top\mathbf{\Gamma})^{-1}\mathbf{X}^\top\mathbf{y}$, and one can again verify that this is a minimizer by showing that the Hessian is positive definite since the matrix $\mathbf{\Gamma}$ has full rank, implying that $\mathbf{\Gamma}^\top\mathbf{\Gamma}$ is itself positive definite.

5 Robotic Learning of Controls from Demonstrations and Images

Huey, a home robot, is learning to retrieve objects from a cupboard, as shown in Fig. 1. The goal is to push obstacle objects out of the way to expose a goal object. Huey's robot trainer, Anne, provides demonstrations via tele-operation. When tele-operating the robot, Anne can look at the images captured by the robot and provide controls to Huey remotely.

During a demonstration, Huey records the RGB images of the scene for each of the n timesteps, $x_1, ..., x_n$, where $x_i \in \mathbb{R}^{30\times30\times3}$ and the controls for his body for each of the n timesteps, $u_1, \ldots, u_n$, where $u_i \in \mathbb{R}^3$. The controls correspond to making small changes in the 3D pose (i.e. translation and rotation) of his body. Examples of the data are shown in the figure.

Under an assumption (sometimes called the Markovian assumption) that all that matters for the current control is the current image, Huey can try to learn a linear *policy* π (where $\pi \in \mathbb{R}^{2700\times3}$) which linearly maps image states to controls (i.e. $\pi^\top x = u$). We will now explore how Huey can recover this policy using linear regression.

Note the dimensions in this problem! Previously, you saw linear regression in problems in which the learned weight w^* was a vector and the predicted value y was a scalar. Here, we are predicting 3D controls. This means that the learned policy is a matrix. In essence, we are performing 3 regressions at the same time, one for each element of the predicted control u.

Please stick to **numpy** (and **numpy.linalg**) only for performing any computations in this assignment. We will ask that you edit the file `robotic_ridge_code.py` directly, instead of working in a Python notebook, and submit it to the Gradescope autograder after you are finished. Please don't rename the file, or change any of the function signatures!

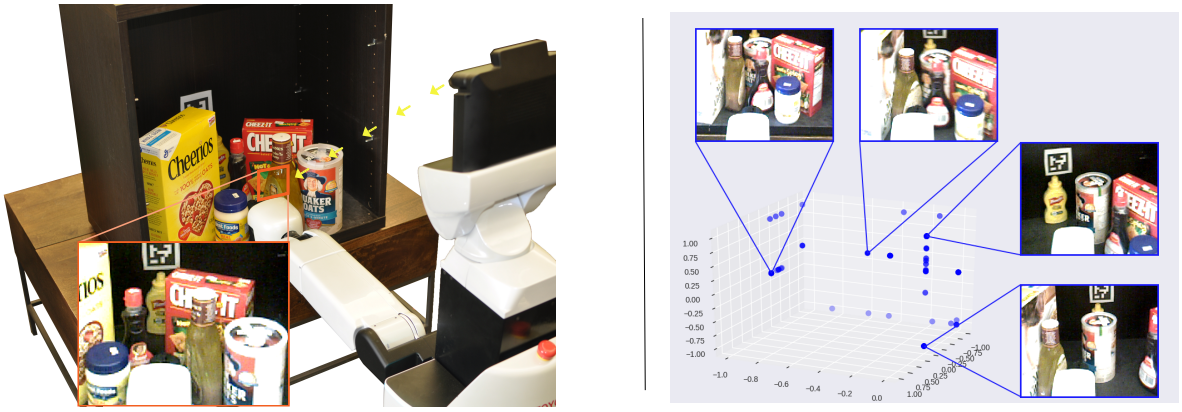


Figure 1: A) Huey trying to retrieve a mustard bottle. An example RGB image of the workspace taken from his head mounted camera is shown in the orange box. The angle of the view gives Huey an eye-in-hand perspective of the cupboard he is reaching into. B) A scatter plot of the 3D control vectors, or u labels. Notice that each coordinate of the label lies within the range of $[-1, 1]$ for the change in position. Example images, or states x, are shown for some of the corresponding control points. The correspondence is indicated by the blue lines.

(a) To get familiar with the structure of the data, **please visualize the 0th, 10th and 20th images in the training dataset. Also find their corresponding control vectors.**
Note: the training and testing images are currently stored as float32 numpy arrays, with pixel values in the range [0.0, 255.0]. You may have to convert to these images to the np.uint8 format to visualize them.

Solution:

Figure 2: The 0th, 10th and 20th images in the training set.

The corresponding controls are $[0, -1, 0]$, $[-1, -0.45111084, -1]$, and $[0, 0, 0.37368774]$.

(b) Load the n training examples from `x_train.p` and compose the matrix X, where $X \in \mathbb{R}^{n \times 2700}$. Note that you will need to flatten the images and reduce them to a single vector. The flattened image vector will be denoted by $\bar{x}$ (where $\bar{x} \in \mathbb{R}^{2700 \times 1}$). Next, load the n examples from `y_train.p` and compose the matrix U, where $U \in \mathbb{R}^{n \times 3}$. Try to perform ordinary least squares by forming the matrix $(X^\top X)^{-1} X^\top$ for solving

$$\min_{\pi} ||X\pi - U||_F$$

in order to learn the optimal *policy* $\pi^* \in \mathbb{R}^{2700 \times 3}$. **Report what happens as you attempt to do this and explain why**.

Solution: The matrix is singular and not invertible. The reason is that there isn't enough data to cover the high dimensional image space, and so many solutions exist that can solve the problem above. Specifically, we only train the policy on 91 images. However, a single RGB image is $30 \times 30 \times 3$ in dimensionality. So, with that many parameters, we can basically fit any arbitrary set of controls (this is called the temptation to memorize in machine learning). We can use ridge regression though to help condition the optimization to favor solutions with a small ℓ_2 norm.

(c) Now try to perform ridge regression:

$$\min_{\pi} \; ||X\pi - U||_F^2 + \lambda ||\pi||_F^2$$

on the dataset for regularization values $\lambda = \{0.1, 1.0, 10, 100, 1000\}$. Measure the average squared Euclidean distance for the accuracy of the policy on the training data:

$$\frac{1}{n}\sum_{i=0}^{n-1} ||\bar{x}_i^T \pi - u_i^\top||_2^2$$

In the expression above, we are taking the ℓ_2 norm of a row vector, which here we take to mean the ℓ_2 norm of the column vector we get by transposing it. **Report the training error results for each value of λ.**

Solution:

The learned policy should match the training data with very low error. For all values of λ, in the specified range, you should see a training error that is either on the order of or below 10^{-8}, which is a very good fit for the dataset. It should be noted that the ability to fit training data perfectly does not necessarily mean the robot will perform well on unseen data. Even with the ridge penalties set to these values, the policy has apparently just memorized the controls.

Why was it able to do this? Remember, all the controls are small numbers between -1 and 1 while the training image data has large numbers. Consequently, small numbers for the parameters will allow these controls to be recovered. Furthermore, there are so many parameters that the work of memorizing the controls can be distributed across them. This further shrinks the parameters required to do this memorization. The ridge penalties are simply incapable of discouraging this memorization.

(d) Next, we are going to try standardizing the states. For each pixel value in each data point, $\bar{x}$, perform the following operation:

$$\bar{x} \mapsto \frac{\bar{x}}{255} \times 2 - 1$$

We know that the maximum pixel value is 255, so this operation rescales the data to be in the range $[-1, 1]$. **Repeat the previous part and report the average squared training error for each value of λ.**

Solution:

The average fitting errors for $\lambda = \{0.1, 1.0, 10, 100, 1000\}$ are $\{0.0, 0.0, 0.0016, 0.035, 0.25\}$ (the first two zeros are negligible errors on the order of 10^{-7} and 10^{-5}). With standardization applied, we see that, as the regularization term is decreased, the training loss is lowered. This can be interpreted as the variance of the model increasing as the possible function class is expanded.

We also observe that smaller values in the input vectors is forcing the learned policy to have bigger values for its parameters, if it wants to memorize the controls. The regularization penalty is now able to begin to discourage this behavior.

(e) Evaluate both *policies* (i.e. with and without standardization) on the new validation data `x_test.p` and `y_test.p` for the different values of λ. **Report the average squared Euclidean**

loss and qualitatively explain how changing the values of λ affects the performance in terms of bias and variance.

Solution:

The average fitting errors for $\lambda = \{0.1, 1.0, 10, 100, 1000\}$ are $\{0.87, 0.86, 0.83, 0.72, 0.73\}$ with standardization and $\{0.77, 0.77, 0.77, 0.77, 0.77\}$ without standardization.

The results with standardization illustrate that because the state space is so high dimensional, the policy has trouble generalizing and is, thus, adding bias (i.e. increasing λ can help generalization). However, increasing it too much can lead to worse performance.

We empirically see that without standardization the test error is higher; in the next section, we will examine how we can characterize this.

It should be noted that the ability of Huey to generalize is still quite inadequate for a real robot policy. Later in the course, we will explore how to get significantly lower error on a larger dataset using convolutional neural networks.

(f) To better understand how standardizing improved the loss function, we are going to evaluate the *condition number* κ of the optimization problem above, which is defined as

$$\kappa = \frac{\sigma_{\max}(X^T X + \lambda I)}{\sigma_{\min}(X^T X + \lambda I)}$$

or the ratio of the maximum singular value to the minimum singular value of the relevant matrix. Roughly speaking, the condition number of the optimization process measures how stable the solution will be when some error exists in the observations. More precisely, given a linear system $Ax = b$, the condition number of the matrix A is the maximum ratio of the relative error in the solution x to the relative error of b.

For the regularization value of $\lambda = 100$, **report the condition number with the standardization technique applied and without**.

Solution: The condition number without standardization is $\kappa = 52711697.67$ and with standardization is $\kappa = 444.73$. Thus, by standardizing our data, we are able to significantly reduce the ratio of the eigenvalues, which makes our optimization less sensitive to noise in the data when performing matrix inversion.

Solution:

```
import numpy as np
import numpy.linalg as LA
import pickle
from PIL import Image

def load_data() -> tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray]:
    x_train = pickle.load(open('x_train.p', 'rb'), encoding='latin1')
    y_train = pickle.load(open('y_train.p', 'rb'), encoding='latin1')
    x_test = pickle.load(open('x_test.p', 'rb'), encoding='latin1')
    y_test = pickle.load(open('y_test.p', 'rb'), encoding='latin1')
    return x_train, y_train, x_test, y_test
```

```python
def visualize_data(images: np.ndarray, controls: np.ndarray) -> None:
    """
    Args:
        images (ndarray): image input array of size (n, 30, 30, 3).
        controls (ndarray): control label array of size (n, 3).
    """
    for i in [0, 10, 20]:
        instance = images[i]
        print("image size is: ", instance.shape)
        im = Image.fromarray(instance)
        im.save(f'train_image{i}.png')
        print("the corresponding control is: ", controls[i])


def compute_data_matrix(images: np.ndarray, controls: np.ndarray, standardize: bool = False) -> tuple[np.
    ↪ ndarray, np.ndarray]:
    """
    Args:
        images (ndarray): image input array of size (n, 30, 30, 3).
        controls (ndarray): control label array of size (n, 3).
        standardize (bool): boolean flag that specifies whether the images should be standardized or not

    Returns:
        X (ndarray): input array of size (n, 2700) where each row is a flattened image
        Y (ndarray): label array of size (n, 3) where row i corresponds to the control for X[i]
    """
    Y = []
    X = []

    # load training data
    for x, y in zip(images, controls):
        # convert to float64 from unint8
        x = 1.0 * x
        if standardize:
            # standardize image
            x = (x / 255.0) * 2.0 - 1.0
        X.append(x.flatten())
        Y.append(y)

    # convert list to matrix
    X = np.vstack(X)
    Y = np.vstack(Y)
    return X, Y


def ridge_regresion(X: np.ndarray, Y: np.ndarray, lmbda: float) -> np.ndarray:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        Y (ndarray): label array of size (n, 3).
        lmbda (float): ridge regression regularization term

    Returns:
        pi (ndarray): learned policy of size (2700, 3)
    """
    # calculate the Gramian of X
    X_G = np.matmul(X.T, X)

    # add weighted indentity
    X_G = X_G + np.eye(X_G.shape[0]) * lmbda

    X_G_inv = LA.inv(X_G)

    pi = []
    # decouple the matrix ridge regression problem into multiple vector ridge regression problems
    for i in range(3):
        pi.append(X_G_inv @ X.T @ Y[:, i])
    pi = np.stack(pi, axis=1)
```

```
    return pi


def ordinary_least_squares(X: np.ndarray, Y: np.ndarray) -> np.ndarray:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        Y (ndarray): label array of size (n, 3).

    Returns:
        pi (ndarray): learned policy of size (2700, 3)
    """
    ### NOTE: the following line will throw an error due to X_G being rank deficient ###
    return ridge_regresion(X, Y, lmbda=0)


def measure_error(X: np.ndarray, Y: np.ndarray, pi: np.ndarray) -> float:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        Y (ndarray): label array of size (n, 3).
        pi (ndarray): learned policy of size (2700, 3)

    Returns:
        error (float): the mean Euclidean distance error across all n samples
    """
    prediction = X @ pi
    error = LA.norm(prediction - Y, ord='fro') ** 2
    n, _ = X.shape
    return error / n


def compute_condition_number(X: np.ndarray, lmbda: float) -> float:
    """
    Args:
        X (ndarray): input array of size (n, 2700).
        lmbda (float): ridge regression regularization term

    Returns:
        kappa (float): condition number of the input array with the given lambda
    """
    # calculate the Gramian of X
    X_G = np.matmul(X.T, X)

    # add weighted indentity
    X_G = X_G + np.eye(X_G.shape[0]) * lmbda

    # computed the singular values of the ridge Gramian
    _, s, _ = LA.svd(X_G)

    # by convention, s[0] is sigma_max and s[-1] is sigma_min
    condition_number = s[0] / s[-1]
    return condition_number


if __name__ == '__main__':

    x_train, y_train, x_test, y_test = load_data()
    print("successfully loaded the training and testing data")

    LAMBDA = [0.1, 1.0, 10.0, 100.0, 1000.0]

    print("\npart (a)")
    visualize_data(x_train, y_train)

    X_train, Y_train = compute_data_matrix(x_train, y_train, standardize=False)
    X_test, Y_test = compute_data_matrix(x_test, y_test, standardize=False)
    X_train_std, Y_train_std = compute_data_matrix(x_train, y_train, standardize=True)
```

```
X_test_std, Y_test_std = compute_data_matrix(x_test, y_test, standardize=True)

print("\npart (b)")

try:
    ordinary_least_squares(X_train, Y_train)
except:
    print("rank deficient matrices don't have an inverse")

print("\npart (c)")
for lmbda in LAMBDA:
    pi = ridge_regresion(X_train, Y_train, lmbda)
    print(pi.shape)
    print("lambda: ", lmbda)
    print("training error: ", measure_error(X_train, Y_train, pi))

print("\npart (d)")

for lmbda in LAMBDA:
    pi = ridge_regresion(X_train_std, Y_train_std, lmbda)
    print("lambda: ", lmbda)
    print("training error: ", measure_error(X_train_std, Y_train_std, pi))

print("\npart (e)")
for lmbda in LAMBDA:
    pi = ridge_regresion(X_train, Y_train, lmbda)
    print("lambda: ", lmbda)
    print("testing error: ", measure_error(X_test, Y_test, pi))

for lmbda in LAMBDA:
    pi = ridge_regresion(X_train_std, Y_train_std, lmbda)
    print("lambda: ", lmbda)
    print("testing error: ", measure_error(X_test_std, Y_test_std, pi))

print("\npart (f)")
kappa = compute_condition_number(X_train, lmbda=100)
print(f"condition number without standardization: {kappa}")

kappa = compute_condition_number(X_train_std, lmbda=100)
print(f"condition number with standardization: {kappa}")
```

6 Honor Code

1. **List all collaborators. If you worked alone, then you must explicitly state so.**
2. **Declare and sign the following statement**:

   "I certify that all solutions in this document are entirely my own and that I have not looked at anyone else's solution. I have given credit to all external sources I consulted."

   Signature : _____

   While discussions are encouraged, *everything* in your solution must be your (and only your) creation. Furthermore, all external material (i.e., *anything* outside lectures and assigned readings, including figures and pictures) should be cited properly. We wish to remind you that the consequences of academic misconduct are *particularly severe*!