CS 189 / 289A Introduction to Machine Learning
Fall 2024 Jennifer Listgarten, Saeed Saremi HW1

Due 09/11/24 11:59 PM PT

- Homework 1 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.
- **Start early. This is a long assignment. Most of the material is prerequisite material not covered in lecture; you are responsible for finding resources to understand it.**

Deliverables:

1. Submit a PDF of your homework to the Gradescope assignment entitled "HW 1 Write-Up". **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 "Homework 1 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 Linear Algebra Review

1. **First isomorphism theorem.** The isomorphism theorems are an important class of results with versions for various algebraic structures. Here, we are concerned about the first isomorphism theorem for vector spaces–one of the most fundamental results in linear algebra.

   Theorem. *Let V, W be vector spaces, and let $T : V \to W$ be a linear map. Then the following are true:*

   (a) $\ker T$ *is a subspace of* V.

   (b) $\operatorname{Im} T$ *is a subspace of* W.

   (c) $\operatorname{Im} T$ *is isomorphic to* $V/\ker T$.

   Prove **parts (a) and (b)** of the theorem. (The interesting result is part (c), so, if you're inclined, try it out! We promise it's a very rewarding proof :) If you are interested but unfamiliar with the language, try looking up "isomorphism" and "quotient space.")

   Solution: To prove that U, where $U \subset U'$ for some vector space U', is a subspace, we must show that U contains the zero vector of U', U is closed under vector addition, and U is closed under scalar multiplication.

   (a) By definition, we know that $\ker T \subset V$, and $\mathbf{0} \in \ker T$ is trivial. Let $\mathbf{x}, \mathbf{y} \in \ker T$. Then $T(\mathbf{x} + \mathbf{y}) = T(\mathbf{x}) + T(\mathbf{y}) = \mathbf{0} \implies \mathbf{x} + \mathbf{y} \in \ker T$. Hence, $\ker T$ is closed under vector addition. Now let $\mathbf{z} \in V$ and α some scalar. Then $T(\alpha\mathbf{z}) = \alpha T(\mathbf{z}) = 0 \implies \alpha\mathbf{z} \in \ker T$. Hence, $\ker T$ is closed under scalar multiplication. This concludes our proof of the first statement.

   (b) Since $T(\mathbf{0}) = \mathbf{0}$, we have $\mathbf{0} \in \operatorname{Im} T$. Let $\mathbf{x}, \mathbf{y} \in \operatorname{Im} T$. Then $\exists \mathbf{a}, \mathbf{b} \in V$ such that $T(\mathbf{a}) = \mathbf{x}$ and $T(\mathbf{b}) = \mathbf{y}$. Then $\mathbf{x} + \mathbf{y} = T(\mathbf{a}) + T(\mathbf{b}) = T(\mathbf{a} + \mathbf{b}) \implies x + y \in \operatorname{Im} T$. Hence, $\operatorname{Im} T$ is closed under vector addition. Now let $\mathbf{z} \in \operatorname{Im} T$ and α some scalar. Then $\exists \mathbf{c} \in V$ such that $T(\mathbf{c}) = \mathbf{z}$. $\alpha\mathbf{z} = \alpha T(\mathbf{c}) = T(\alpha\mathbf{c}) \implies \alpha\mathbf{z} \in \operatorname{Im} T$. Hence, $\operatorname{Im} T$ is closed under scalar multiplication. This concludes our proof of the second statement.

   [grading note] For full credit, students should not assume anything about the sets of vectors or the field of scalars of V and W. (Referring to the field as some arbitrary $\mathbb{F}$ is fine.)

2. First we review some basic concepts of rank. Recall that elementary matrix operations do not change a matrix's rank. Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$. Let I_n denote the $n \times n$ identity matrix.

   (a) Perform elementary row and column operations[1] to transform $\begin{bmatrix} I_n & 0 \\ 0 & AB \end{bmatrix}$ to $\begin{bmatrix} B & I_n \\ 0 & A \end{bmatrix}$.

   (b) Let's find lower and upper bounds on $\operatorname{rank}(AB)$. Use part (a) to prove that $\operatorname{rank} A + \operatorname{rank} B - n \le \operatorname{rank}(AB)$. Then use what you know about the relationship between the column space (range) and/or row space of AB and the column/row spaces for A and B to argue that $\operatorname{rank}(AB) \le \min\{\operatorname{rank} A, \operatorname{rank} B\}$.

[1] If you're not familiar with these, https://stattrek.com/matrix-algebra/elementary-operations is a decent introduction.

(c) If a matrix A has rank r, then some $r \times r$ submatrix M of A has a nonzero determinant. Use this fact to show the standard facts that the dimension of A's column space is at least r, and the dimension of A's row space is at least r. (Note: You will not receive credit for other ways of showing those same things.)

(d) It is a fact that $\operatorname{rank}(A^\top A) = \operatorname{rank} A$; here's one way to see that. We've already seen in part (b) that $\operatorname{rank}(A^\top A) \le \operatorname{rank} A$. Suppose that $\operatorname{rank}(A^\top A)$ were strictly less than $\operatorname{rank} A$. What would that tell us about the relationship between the column space of A and the null space of $A^\top$? What standard fact about the fundamental subspaces of A says that relationship is impossible?

(e) Given a set of vectors $S \subseteq \mathbb{R}^n$, let $AS = \{Av : v \in s\}$ denote the subset of $\mathbb{R}^m$ found by applying A to every vector in S. In terms of the ideas of the column space (range) and row space of A: What is $A\mathbb{R}^n$, and why? (*Hint*: what are the definitions of column space and row space?) What is $A^\top A\mathbb{R}^n$, and why? (Your answer to the latter should be purely in terms of the fundamental subspaces of A itself, not in terms of the fundamental subspaces of $A^\top A$.)

Solution:

(a) The operations are as follows.

$$\begin{bmatrix} I_n & 0 \\ 0 & AB \end{bmatrix} \Rightarrow \begin{bmatrix} I_n & 0 \\ A & AB \end{bmatrix} \Rightarrow \begin{bmatrix} I_n & -B \\ A & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} B & I_n \\ 0 & A \end{bmatrix}$$

In order,

- Left multiply the first row by A and add it to the second row
- Right multiply the first column by B and subtract it from the second column
- Negate the right column and exchange it with the left one

(b) Using (a) and rearranging gives us the lower bound,

$$n + \operatorname{rank}(AB) = \operatorname{rank}\left(\begin{bmatrix} I_n & 0 \\ 0 & AB \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} B & I_n \\ 0 & A \end{bmatrix}\right) \ge \operatorname{rank} A + \operatorname{rank} B$$

Let $\mathcal{R}(M)$ denote the range (column space) of a matrix M. Since $\mathcal{R}(AB) \subseteq \mathcal{R}(A)$, we have $\operatorname{rank}(AB) \le \operatorname{rank} A$. Similarly, since $\mathcal{R}(B^\top A^\top) \subseteq \mathcal{R}(B^\top)$—that is, the row space of AB is a subset of the row space of B—we have $\operatorname{rank}(AB) \le \operatorname{rank} B$. Thus $\operatorname{rank}(AB) \le \min\{\operatorname{rank} A, \operatorname{rank} B\}$.

(c) As M has a nonzero determinant, its columns are linearly independent, and so are its rows; so its column space and row space are both $\mathbb{R}^r$.

Therefore, the r columns of A from which M is drawn are linearly independent—as we go from M to A, lengthening the columns can't break their linear independence. Likewise, the r rows of A from which M is drawn are linearly independent. The column space of A is the set of vectors in $\mathbb{R}^m$ spanned by the columns of A, so it must have dimension at least r. The row space of A is the set of vectors in $\mathbb{R}^n$ spanned by the rows of A, so it must have dimension at least r.

(d) If $\operatorname{rank}(A^\top A) < \operatorname{rank} A$, then the column space of A has dimension $\operatorname{rank} A$, but when we multiply $A^\top$ by that column space, we get a subspace of smaller dimension. Therefore, there is a nonzero vector v in the column space of A that is also in the null space of $A^\top$. (In other words, the intersection of the column space of A with the null space of $A^\top$ contains more than just the origin.) But the column space of A is the row space of $A^\top$, which is well known to be orthogonal to the null space of $A^\top$. This implies that v is orthogonal to itself but nonzero, a contradiction.

(e) $A\mathbb{R}^n$ is the column space of A, because if A_{*i} denotes the ith column of A, then $Av = \sum_{i=1}^n v_i A_{*i}$, and as v ranges over $\mathbb{R}^n$, the set $A\mathbb{R}^n$ contains all possible linear combinations of the columns of A.

$A^\top A\mathbb{R}^n$ is the row space of A, because $\mathcal{R}(A^\top A) \subseteq \mathcal{R}(A^\top)$, but it can't be a strict subset, because $\operatorname{rank}(A^\top A) = \operatorname{rank} A$. $\mathcal{R}(A^\top)$ is the row space of A.

3. Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Prove equivalence between these three different definitions of positive semidefiniteness (PSD). Note that when we talk about PSD matrices in this class, they are defined to be symmetric matrices. There are nonsymmetric matrices that exhibit PSD properties, like the first definition below, but not all three.

(a) For all $x \in \mathbb{R}^n$, $x^\top Ax \geq 0$.

(b) All the eigenvalues of A are nonnegative.

(c) There exists a matrix $U \in \mathbb{R}^{n\times n}$ such that $A = UU^\top$.

Positive semidefiniteness will be denoted as $A \succeq 0$.

Solution: Showing the cycle of implications (a) $\implies$ (b) $\implies$ (c) $\implies$ (a) will prove the equivalence between all 3 statements:

(a) $\Rightarrow$ (b): Let λ be an eigenvalue of A with corresponding eigenvector v. Then

$$v^\top Av = \lambda v^\top v = \lambda\|v\|^2.$$

By part (a), we know that $\lambda\|v\|^2 \geq 0$, so $\lambda \geq 0$.

(b) $\Rightarrow$ (c): Consider the eigendecomposition of A, $A = V\Lambda V^\top$, where Λ is a diagonal matrix with entries equal to the eigenvalues of A, $\lambda_1, \dots, \lambda_n$. Define $U := V\sqrt{\Lambda}$, where $\sqrt{\Lambda}$ is diagonal with entries equal to $\sqrt{\lambda_1}, \dots, \sqrt{\lambda_n}$; notice that this choice is justified because, by assumption, the eigenvalues are non-negative. Clearly, $A = UU^\top$.

(c) $\Rightarrow$ (a): Let $x \in \mathbb{R}^n$. Then

$$x^\top Ax = x^\top UU^\top x = (U^\top x)^\top(U^\top x) = \|U^\top x\|^2 \geq 0.$$

4. The Frobenius inner product between two matrices of the same dimensions $A, B \in \mathbb{R}^{m\times n}$ is

$$\langle A, B\rangle = \operatorname{trace}(A^\top B) = \sum_{i=1}^m \sum_{j=1}^n A_{ij}B_{ij},$$

where trace M denotes the *trace* of M, which you should look up if you don't already know it. (The norm is sometimes written $\langle A, B\rangle_F$ to be clear.) The Frobenius norm of a matrix is

$$\|A\|_F = \sqrt{\langle A, A\rangle} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |A_{ij}|^2}.$$

Prove the following. The Cauchy–Schwarz inequality, the cyclic property of the trace, and the definitions in part 3 above may be helpful to you.

(a) $x^\top Ay = \langle A, xy^\top\rangle$ for all $x \in \mathbb{R}^m, y \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}$.

(b) If A and B are symmetric PSD matrices, then trace $(AB) \geq 0$.

(c) **[OPTIONAL]** If $A, B \in \mathbb{R}^{n\times n}$ are real symmetric matrices with $\lambda_{\max}(A) \geq 0$ and B being PSD, then $\langle A, B\rangle \leq \sqrt{n}\lambda_{\max}(A)\|B\|_F$.
Hint: Construct a PSD matrix using $\lambda_{\max}(A)$

Solution:

(a) Realize that $x^\top Ay = \text{trace}\,(x^\top Ay) = \text{trace}\,(yx^\top A) = \langle xy^\top, A\rangle = \langle A, xy^\top\rangle$. Alternatively, you can expand out both terms: $x^\top Ay = \sum_{i=1,\,j=1}^{m,n} x_i \cdot A_{ij} \cdot y_j = \langle A, xy^\top\rangle$.

(b) By the third definition of PSD, let $A = UU^\top$ and $B = VV^\top$. Then

$$\text{trace}\,(AB) = \text{trace}\,(UU^\top VV^\top) = \text{trace}\,(U^\top VV^\top U) = \text{trace}\,(U^\top V(U^\top V)^\top) \geq 0,$$

which follows because $M \overset{\text{def}}{=} U^\top V(U^\top V)^\top$ is PSD by the third definition, and trace $M \geq 0$, since trace is the sum of all eigenvalues.

(c) First realize that $\lambda_{\max}(A)I_n - A \geq 0$. This follows from the (a) definition of PSD matrices. Then by the identity proved above, trace $((\lambda_{\max}(A)I_n - A)B) \geq 0$, and rearranging gives trace $(AB) \leq \lambda_{\max}(A)$trace (I_n, B). By the second identity proved in this question we have trace $(I_n, B) \leq \sqrt{n}\|B\|_F$.

5. Let $A \in \mathbb{R}^{m\times n}$ be an arbitrary matrix. The maximum singular value of A is defined to be $\sigma_{\max}(A) = \sqrt{\lambda_{\max}(A^\top A)} = \sqrt{\lambda_{\max}(AA^\top)}$. Prove that

$$\sigma_{\max}(A) = \max_{\substack{u\in\mathbb{R}^m,\ v\in\mathbb{R}^n \\ \|u\|=1,\ \|v\|=1}} (u^\top Av).$$

Solution: In this solution, we abbreviate $\sigma_{\max}(A)$ as σ for notational convenience.

The proof will decompose into two steps. We will first show that σ^2 can be written as the supremum of a quadratic form over the unit sphere of $\mathbb{R}^n$. We will then reformulate this supremum to show the desired result.

Step 1: Observe that A^TA is a symmetric matrix, and that $\forall x \in \mathbb{R}^n$, we have $x^TA^TAx = (Ax)^T(Ax) = \|Ax\|^2 \geq 0$, that is, A^TA is positive semi-definite. We can, therefore, apply the spectral theorem to A^TA. There exists an orthogonal matrix $U \in \mathbb{R}^{n\times n}$, and a diagonal matrix $\Lambda \in \mathbb{R}^{n\times n}$ such that $A^TA = U\Lambda U^T$. Consequently,

$$\sup_{v\in\mathbb{R}^n,\ \|v\|=1} v^TA^TAv = \sup_{v\in\mathbb{R}^n,\ \|v\|=1} v^TU\Lambda U^Tv.$$

Consider the change of variable $w = U^Tv$. Since U^T is orthogonal, the image of the unit sphere $\{v \in \mathbb{R}^n, \|v\| = 1\}$ under the transformation $v \mapsto U^Tv$ is the unit sphere $\{w \in \mathbb{R}^n, \|w\| = 1\}$. Therefore,

$$\sup_{v\in\mathbb{R}^n,\ \|v\|=1} v^TA^TAv = \sup_{w\in\mathbb{R}^n,\ \|w\|=1} w^T\Lambda w.$$

Observe that $w^T\Lambda w = \sum_{i=1}^n \lambda_i w_i^2$, where λ_i is the i-th largest eigenvalue of A^TA. The supremum of this sum is achieved when w is the unit vector associated with the largest eigenvalue of A^TA, that is, $\lambda_{\max}(A^\top A) = \sigma^2$. This implies that $\sigma^2 = \sup_{v,\ \|v\|=1} v^TA^TAv$.

Step 2: In parallel,

$$\sup_{v\in\mathbb{R}^n,\ \|v\|=1} v^TA^TAv = \sup_{v\in\mathbb{R}^n,\ \|v\|=1} \|Av\|^2$$

The application of the Cauchy-Schwarz theorem yields that for any unit vector $u \in \mathbb{R}^m$, $u \cdot Av \leq \|Av\|\|u\| = \|Av\|$, and that this inequality is tight when u is a scalar multiple of Av. This implies that $\|Av\| = \sup_{u\in\mathbb{R}^m,\ \|u\|=1} u^TAv$, and therefore,

$$\sigma^2 = \sup_{v\in\mathbb{R}^n,\ \|v\|=1}\ \sup_{u\in\mathbb{R}^m,\ \|u\|=1} \left[u^TAv\right]^2,$$

which concludes the proof.

2 Matrix/Vector Calculus and Norms

1. Consider a 2×2 matrix A, written in full as $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, and two arbitrary 2-dimensional vectors x, y. Calculate the gradient of

$$\sin\left(A_{11}^2 + e^{A_{11}+A_{22}}\right) + x^\top A y$$

with respect to the matrix A.
Hint: The gradient has the same dimensions as A. Use the chain rule.

Solution: We will take derivatives of $\sin\left(A_{11}^2 + e^{A_{11}+A_{22}}\right)$ and $x^\top A y$ separately.
First consider the former term, and take the derivative with respect to A_{11}.

$$\frac{\partial}{\partial A_{11}} \sin\left(A_{11}^2 + e^{A_{11}+A_{22}}\right) = \cos\left(A_{11}^2 + e^{A_{11}+A_{22}}\right)[(2A_{11}) + e^{A_{11}+A_{22}}).$$

A similar computation can be done for A_{22}. Therefore, the derivative of $\sin\left(A_{11}^2 + e^{A_{11}+A_{22}}\right)$ with respect to A is

$$\begin{bmatrix} \cos\left(A_{11}^2 + e^{A_{11}+A_{22}}\right)[(2A_{11}) + e^{A_{11}+A_{22}})] & 0 \\ 0 & \cos\left(A_{11}^2 + e^{A_{11}+A_{22}}\right) \cdot e^{A_{11}+A_{22}} \end{bmatrix}.$$

Furthermore, $x^\top A y = \langle A, xy^\top \rangle$ by Q1.4(a). Therefore, its derivative with respect to A is $xy^\top$. Thus, combined together, the final derivative is

$$\begin{bmatrix} \cos\left(A_{11}^2 + e^{A_{11}+A_{22}}\right)[(2A_{11}) + e^{A_{11}+A_{22}})] + x_1 y_1 & x_1 y_2 \\ x_2 y_1 & \cos\left(A_{11}^2 + e^{A_{11}+A_{22}}\right) \cdot e^{A_{11}+A_{22}} + x_2 y_2 \end{bmatrix}.$$

2. Aside from norms on vectors, we can also impose norms on matrices. Besides the Frobenius norm, the most common kind of norm on matrices is called the induced norm. Induced norms are defined as

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

where the notation $\|\cdot\|_p$ on the right-hand side denotes the vector ℓ_p-norm. Please give the closed-form (or the most simple) expressions for the following induced norms of $A \in \mathbb{R}^{m \times n}$.

(a) $\|A\|_2$. *Hint*: use the singular value decomposition.

(b) $\|A\|_\infty$

Solution:

(a) Let $A = U\Sigma V^\top$ be an SVD of A. Then

$$\sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sup_{x \neq 0} \frac{\|U\Sigma V^\top x\|_2}{\|x\|_2}$$

$$
\begin{aligned}
&= \sup_{x\neq 0} \frac{||\Sigma V^\top x||_2}{||x||_2} \\
&= \sup_{y\neq 0} \frac{||\Sigma y||_2}{||Vy||_2} (\text{suppose } y = V^\top x) \\
&= \sup_{y\neq 0} \frac{||\Sigma y||_2}{||y||_2} \\
&= \sigma_1. \text{ (The largest singular value of } A \text{ in absolute value.)}
\end{aligned}
$$

(b) We have

$$
\begin{aligned}
\sup_{x\neq 0} \frac{||Ax||_\infty}{||x||_\infty} &= \sup_{x\neq 0} \frac{\max_i(|a_{i1}x_1 + \cdots + a_{in}x_n|)}{\max_i |x_i|} \\
&= \sup_{x\neq 0} \frac{\max_i(|a_{i1}x_1| + \cdots + |a_{in}x_n|)}{\max_i |x_i|} \\
&= \sup_{x\neq 0} \frac{\max_i(|a_{i1}|\,|x_1| + \cdots + |a_{in}|\,|x_n|)}{\max_i |x_i|} \\
&= \max_i \sum_{j=1}^{n} |a_{ij}| \text{ (the largest row sum)}
\end{aligned}
$$

3. (a) Let $\alpha = \sum_{i=1}^{n} y_i \ln(1 + e^{\beta_i})$ for $y, \beta \in \mathbb{R}^n$. What are the partial derivatives $\frac{\partial\alpha}{\partial\beta_i}$?

   (b) Given $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{mxn}$. Write the partial derivative $\frac{\partial(Ax)}{\partial x}$.

   (c) Given $z \in \mathbb{R}^m$. Write the gradient $\nabla_z(z^\top z)$.

   (d) Given $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$, and $z = g(x)$. Write the gradient $\nabla_x z^\top z$ in terms of $\frac{\partial z}{\partial x}$ and z.

   (e) Given $x \in \mathbb{R}^n$, y, $z \in \mathbb{R}^m$, $A \in \mathbb{R}^{mxn}$, and $z = Ax - y$. Write the gradient $\nabla_x(z^\top z)$.

Solution:

(a) $\frac{\partial\alpha}{\partial\beta_i} = \frac{\partial\alpha_i}{\partial\beta_i} = y_i \cdot \frac{\partial}{\partial\beta_i} \ln(1 + e^{\beta_i}) = y_i \cdot \frac{1}{1+e^{\beta_i}} \cdot e^{\beta_i} = \frac{y_i e^{\beta_i}}{1+e^{\beta_i}}$

(b) $\frac{\partial(Ax)_i}{\partial x_j} = \frac{\partial(\sum_{k=1}^{n} A_{ik})}{\partial x_j} = A_{ij}$, so $\frac{\partial(Ax)}{\partial x} = A$.

(c) Note that $z^\top z$ is a scalar that can be expressed element-wise as:

$$z^\top z = \sum_{j=1}^{m} z_j^2$$

Therefore the partial derivatives can be expressed as:

$$\frac{\partial z^\top z}{\partial z_i} = 2z_i$$

The gradient is therefore:

$$\nabla_z z^\top z = 2z$$

(d) Applying the chain rule:

$$\frac{d(z^\top z)}{dx_i} = \sum_{j=1}^{m} \left(\frac{dz^\top z}{dz_j}\right)\left(\frac{dz_j}{dx_i}\right)$$

Plugging in $\frac{\partial z^\top z}{\partial z} = (\nabla_z z^\top z)^\top$ from (c):

$$\frac{\partial z^\top z}{\partial x} = 2z^\top \left(\frac{\partial z}{\partial x}\right)$$

The gradient is therefore:

$$\nabla_x z^\top z = 2\left(\frac{\partial z}{\partial x}\right)^\top z$$

(e) Building on the previous parts, let $z = g(x) = Ax - y$ and $\frac{\partial(Ax-y)}{\partial x} = \frac{\partial(Ax)}{\partial x} = A$, since y is a constant.

$$\nabla_x z^\top z = 2A^\top(Ax - y)$$

3 Linear Neural Networks

Let's apply the multivariate chain rule to a "simple" type of neural network called a *linear neural network*. They're not very powerful, as they can learn only linear regression functions or decision functions, but they're a good stepping stone for understanding more complicated neural networks. We are given an $n \times d$ *design matrix* X. Each row of X is a training point, so X represents n training points with d features each. We are also given an $n \times k$ matrix Y. Each row of Y is a set of k labels for the corresponding training point in X. Our goal is to learn a $k \times d$ matrix W of weights[2] such that

$$Y \approx XW^\top.$$

If n is larger than d, typically there is no W that achieves equality, so we seek an approximate answer. We do that by finding the matrix W that minimizes the *cost function*

$$\text{RSS}(W) = \|XW^\top - Y\|_F^2. \tag{1}$$

This is a classic *least-squares linear regression* problem; most of you have seen those before. But we are solving k linear regression problems simultaneously, which is why Y and W are matrices instead of vectors.

Linear neural networks. Instead of optimizing W over the space of $k \times d$ matrices directly, we write the W we seek as a product of multiple matrices. This parameterization is called a *linear neural network*.

$$W = \mu(W_L, W_{L-1}, \ldots, W_2, W_1) = W_L W_{L-1} \cdots W_2 W_1.$$

Here, μ is called the *matrix multiplication map* (hence the Greek letter mu) and each W_j is a real-valued $d_j \times d_{j-1}$ matrix. Recall that W is a $k \times d$ matrix, so $d_L = k$ and $d_0 = d$. L is the number of *layers* of "connections" in the neural network. You can also think of the network as having $L + 1$ layers of units: $d_0 = d$ units in the *input layer*, d_1 units in the first *hidden layer*, d_{L-1} units in the last hidden layer, and $d_L = k$ units in the *output layer*.

We collect all the neural network's weights in a *weight vector* $\theta = (W_L, W_{L-1}, \ldots, W_1) \in \mathbb{R}^{d_\theta}$, where $d_\theta = d_L d_{L-1} + d_{L-1} d_{L-2} + \ldots + d_1 d_0$ is the total number of real-valued weights in the network. Thus we can write $\mu(\theta)$ to mean $\mu(W_L, W_{L-1}, \ldots, W_1)$. But you should imagine θ as a column vector: we take all the components of all the matrices $W_L, W_{L-1}, \ldots, W_1$ and just write them all in one very long column vector. Given a fixed weight vector θ, the linear neural network takes an *input vector* $x \in \mathbb{R}^{d_0}$ and returns an *output vector* $y = W_L W_{L-1} \cdots W_2 W_1 x = \mu(\theta) x \in \mathbb{R}^{d_L}$.

Now our goal is to find a weight vector θ that minimizes the composition $\text{RSS} \circ \mu$—that is, it minimizes the cost function

$$J(\theta) = \text{RSS}(\mu(\theta)).$$

We are substituting a linear neural network for W and optimizing the weights in θ instead of directly optimizing the components of W. This makes the optimization problem harder to solve, and you

[2]The reason for the transpose on $W^\top$ is because we think in terms of applying W to an individual training point. Indeed, if $X_i \in \mathbb{R}^d$ and $Y_i \in \mathbb{R}^k$ respectively denote the i-th rows of X and Y transposed to be column vectors, then we can write $Y_i \approx WX_i$. For historical reasons, most papers in the literature use design matrices whose rows are sample points, rather than columns.

would never solve least-squares linear regression problems this way in practice; but again, it is a good exercise to work toward understanding the behavior of "real" neural networks in which μ is *not* a linear function.

We would like to use a gradient descent algorithm to find θ, so we will derive $\nabla_\theta J$ as follows.

1. The gradient $G = \nabla_W \text{RSS}(W)$ is a $k \times d$ matrix whose entries are $G_{ij} = \partial \text{RSS}(W)/\partial W_{ij}$, where $\text{RSS}(W)$ is defined by Equation (1). Knowing that the simple formula for $\nabla_W \text{RSS}(W)$ in matrix notation can be written as the following:

$$\nabla_W \text{RSS}(W) = 2(WX^\top - Y^\top)X$$

prove this fact by deriving a formula for each G_{ij} using summations, simplified as much as possible. *Hint:* To break down $\text{RSS}(W)$ into its component summations, start with the relationship between the Frobenius norm and the trace of a matrix.

Solution: Observe that

$$\begin{aligned}
\text{RSS}(W) &= \text{trace}\left(WX^\top XW^\top - WX^\top Y - Y^\top XW^\top + Y^\top Y\right) \\
&= \sum_{p=1}^{k}(WX^\top XW^\top)_{pp} - \sum_{p=1}^{k}(WX^\top Y)_{pp} - \sum_{p=1}^{k}(Y^\top XW^\top)_{pp} + \text{trace}(Y^\top Y) \\
&= \sum_{p=1}^{k}(WX^\top XW^\top)_{pp} - 2\sum_{p=1}^{k}(Y^\top XW^\top)_{pp} + \text{trace}\left(Y^\top Y\right) \\
&= \sum_{p=1}^{k}\sum_{q=1}^{d}\sum_{r=1}^{n}\sum_{s=1}^{d} W_{pq}X_{rq}X_{rs}W_{ps} - 2\sum_{p=1}^{k}\sum_{q=1}^{n}\sum_{r=1}^{d} Y_{qp}X_{qr}W_{pr} + \text{trace}\left(Y^\top Y\right).
\end{aligned}$$

Therefore,

$$\begin{aligned}
G_{ij} = \frac{\partial}{\partial W_{ij}}\text{RSS}(W) &= \sum_{q=1}^{d}\sum_{r=1}^{n} W_{iq}X_{rq}X_{rj} + \sum_{r=1}^{n}\sum_{s=1}^{d} X_{rj}X_{rs}W_{is} - 2\sum_{q=1}^{n} Y_{qi}X_{qj} \\
&= 2\left(\sum_{q=1}^{d}\sum_{r=1}^{n} W_{iq}X_{rq}X_{rj} - \sum_{q=1}^{n} Y_{qi}X_{qj}\right)
\end{aligned}$$

We can write $G_{ij} = 2(WX^\top X - Y^\top X)_{ij}$, so

$$\nabla_W \text{RSS}(W) = 2(WX^\top - Y^\top)X.$$

2. Directional derivatives are closely related to gradients. The notation $\text{RSS}'_{\Delta W}(W)$ denotes the directional derivative of $\text{RSS}(W)$ in the direction ΔW, and the notation $\mu'_{\Delta\theta}(\theta)$ denotes the directional derivative of $\mu(\theta)$ in the direction $\Delta\theta$.[3] Informally speaking, the directional

[3] "ΔW" and "$\Delta\theta$" are just variable names that remind us to think of these as small displacements of W or θ; the Greek letter delta is not an operator nor a separate variable.

derivative $\text{RSS}'_{\Delta W}(W)$ tells us how much $\text{RSS}(W)$ changes if we increase W by an infinitesimal displacement $\Delta W \in \mathbb{R}^{k\times d}$. (However, any ΔW we can actually specify is not actually infinitesimal; $\text{RSS}'_{\Delta W}(W)$ is a local linearization of the relationship between W and $\text{RSS}(W)$ at W. To a physicist, $\text{RSS}'_{\Delta W}(W)$ tells us the initial velocity of change of $\text{RSS}(W)$ if we start changing W with velocity ΔW.)

Show how to write $\text{RSS}'_{\Delta W}(W)$ as a Frobenius inner product of two matrices, one related to part 3.1.

Solution:

$$\text{RSS}'_{\Delta W}(W) = \langle G, \Delta W\rangle_F = \langle 2(WX^\top - Y^\top)X, \Delta W\rangle_F.$$

3. In principle, we could take the gradient $\nabla_\theta \mu(\theta)$, but we would need a 3D array to express it! As we don't know a nice way to write it, we'll jump directly to writing the directional derivative $\mu'_{\Delta\theta}(\theta)$. Here, $\Delta\theta \in \mathbb{R}^{d_\theta}$ is a weight vector whose matrices we will write $\Delta\theta = (\Delta W_L, \Delta W_{L-1}, \ldots, \Delta W_1)$. Show that

$$\mu'_{\Delta\theta}(\theta) = \sum_{j=1}^{L} W_{>j}\Delta W_j W_{<j}$$

where $W_{>j} = W_L W_{L-1}\cdots W_{j+1}$, $W_{<j} = W_{j-1}W_{j-2}\cdots W_1$, and we use the convention that $W_{>L}$ is the $d_L \times d_L$ identity matrix and $W_{<1}$ is the $d_0 \times d_0$ identity matrix.

Hint: although μ is not a linear function of θ, μ is linear in any *single* W_j; and any directional derivative of the form $\mu'_{\Delta\theta}(\theta)$ is linear in $\Delta\theta$ (for a fixed θ).

Solution: Observe that μ is linear in any single W_j, so

$$\begin{aligned}&\mu(W_L, \ldots, W_{j+1}, W_j + \Delta W_j, W_{j-1}, \ldots, W_1)\\ &\qquad = \mu(W_L, \ldots, W_{j+1}, W_j, W_{j-1}, \ldots, W_1) + W_{>j}\Delta W_j W_{<j}.\end{aligned}$$

This linearity implies that

$$\mu'_{\Delta W_j}(\theta) = W_{>j}\Delta W_j W_{<j}.$$

For the specific weight vector $\Delta\theta = (0, \ldots, 0, \Delta W_j, 0, \ldots, 0)$, we have $\mu'_{\Delta\theta}(\theta) = \mu'_{\Delta W_j}(\theta) = W_{>j}\Delta W_j W_{<j}$. But what about for a general weight vector? The directional derivative $\mu'_{\Delta\theta}(\theta)$ is linear in $\Delta\theta$ (for a fixed θ). By linearity, we can just add together the directional derivatives for each ΔW_j, yielding

$$\mu'_{\Delta\theta}(\theta) = \sum_{j=1}^{L}\mu'_{\Delta W_j}(\theta) = \sum_{j=1}^{L} W_{>j}\Delta W_j W_{<j}.$$

4. Recall the chain rule for scalar functions, $\frac{\mathrm{d}}{\mathrm{d}x}f(g(x))|_{x=x_0} = \frac{\mathrm{d}}{\mathrm{d}y}f(y)|_{y=g(x_0)} \cdot \frac{\mathrm{d}}{\mathrm{d}x}g(x)|_{x=x_0}$. There is a multivariate version of the chain rule, which we hope you remember from some class you've taken, and the multivariate chain rule can be used to chain directional derivatives. Write out the chain rule that expresses the directional derivative $J'_{\Delta\theta}(\theta)|_{\theta=\theta_0}$ by composing

your directional derivatives for RSS and μ, evaluated at a weight vector θ_0. (Just write the pure form of the chain rule without substituting the values of those directional derivatives; we'll substitute the values in the next part.)

Solution:

$$J'_{\Delta\theta}(\theta)|_{\theta=\theta_0} = \langle \nabla_W \mathrm{RSS}(\mu(\theta)), \mu'_{\Delta\theta}(\theta)\rangle_F|_{\theta=\theta_0}.$$

There are many acceptable variations; e.g.,

$$\begin{aligned} J'_{\Delta\theta}(\theta)|_{\theta=\theta_0} &= \langle \nabla_W \mathrm{RSS}(\mu(\theta_0)), \mu'_{\Delta\theta}(\theta_0)\rangle_F. \\ J'_{\Delta\theta}(\theta)|_{\theta=\theta_0} &= \langle \nabla_W \mathrm{RSS}(W)|_{W=\mu(\theta)}, \mu'_{\Delta\theta}(\theta)|_{\theta=\theta_0}\rangle_F. \\ J'_{\Delta\theta}(\theta)|_{\theta=\theta_0} &= \langle 2\,(\mu(\theta)\,X^\top - Y^\top)X, \mu'_{\Delta\theta}(\theta)\rangle_F|_{\theta=\theta_0}. \end{aligned}$$

Note: we will give part marks for $\langle G, \mu'_{\Delta\theta}(\theta)\rangle_F|_{\theta=\theta_0}$, but not full marks unless there is an indication of awareness that G depends on θ_0.

5. Now substitute the values you derived in parts 3.2 and 3.3 into your expression for $J'_{\Delta\theta}(\theta)$ and use it to show that

$$\begin{aligned} \nabla_\theta J(\theta) = & \ (2\,(\mu(\theta)\,X^\top - Y^\top)XW_{<L}^\top, \\ & \ \ldots, \\ & \ 2W_{>j}^\top\,(\mu(\theta)\,X^\top - Y^\top)XW_{<j}^\top, \\ & \ \ldots, \\ & \ 2W_{>1}^\top\,(\mu(\theta)\,X^\top - Y^\top)X). \end{aligned}$$

This gradient is a vector in $\mathbb{R}^{d_\theta}$ written in the same format as $(W_L, \ldots, W_j, \ldots, W_1)$. Note that the values $W_{>j}$ and $W_{<j}$ here depend on θ.

Hint: you might find the cyclic property of the trace handy.

Solution:

$$\begin{aligned} J'_{\Delta\theta}(\theta) &= \left\langle 2\,(\mu(\theta)\,X^\top - Y^\top)X, \sum_{j=1}^{L} W_{>j}\Delta W_j W_{<j} \right\rangle_F \\ &= \sum_{j=1}^{L} \left\langle 2\,(\mu(\theta)\,X^\top - Y^\top)X, W_{>j}\Delta W_j W_{<j} \right\rangle_F \\ &= \sum_{j=1}^{L} \mathrm{trace}\left(2\,(\mu(\theta)\,X^\top - Y^\top)XW_{<j}^\top \Delta W_j^\top W_{>j}^\top\right) \\ &= \sum_{j=1}^{L} \mathrm{trace}\left(2W_{>j}^\top\,(\mu(\theta)\,X^\top - Y^\top)XW_{<j}^\top \Delta W_j^\top\right) \\ &= \sum_{j=1}^{L} \left\langle 2W_{>j}^\top\,(\mu(\theta)\,X^\top - Y^\top)XW_{<j}^\top, \Delta W_j \right\rangle_F. \end{aligned}$$

As $J'_{\Delta\theta}(\theta) = \nabla_\theta J(\theta) \cdot \Delta\theta$ for all $\Delta\theta \in \mathbb{R}^{d_\theta}$, the gradient $\nabla_\theta J$ must have the value we claim.

4 Probability Potpourri

1. Recall the covariance of two scalar random variables X and Y is defined as $\text{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$. For a multivariate random variable $Z \in \mathbb{R}^n$, (i.e., Z is a column vector where each element Z_i is a scalar random variable), we define the covariance matrix Σ such that $\Sigma_{ij} = \text{Cov}(Z_i, Z_j)$. Concisely, $\Sigma = \mathbb{E}[(Z - \mu)(Z - \mu)^\top]$, where μ is the mean value of Z. Prove that the covariance matrix is always positive semidefinite (PSD).
   Hint: Use linearity of expectation.

   Solution: For any $v \in \mathbb{R}^n$, note that $v^\top \mathbb{E}[(X-\mu)(X-\mu)^\top]v = \Sigma_{i=1}^n \Sigma_{j=1}^n \mathbb{E}[(X_i-\mu_i)(X_j-\mu_j)]v_i v_j = \mathbb{E}[v^\top (X-\mu)(X-\mu)^\top v] = \mathbb{E}[((X-\mu)^\top v)^2] \geq 0$. Note the second identity comes from linearity of expectation.

2. Suppose a pharmaceutical company is developing a diagnostic test for a rare disease that has a prevalence of 1 in 1,000 in the population. Let x be the true positive rate of the test, and let y be the false positive rate. Determine the minimum value that x must have, expressed as a function of y, such that a patient who tests positive actually has the disease with probability greater than 0.5.

   Solution: Let T denote the test results for a patient, with T^+ indicating a positive test and T^- indicating a negative test. Similarly, let D denote whether a patient has a disease with D^+ indicating that they do and D^- indicating that they don't. Our goal is to find the minimum value of x such that $P(D^+ \mid T^+) > 0.5$. To determine this, we use Bayes rule

$$\begin{aligned}
P(D^+ \mid T^+) &> 0.5 \\
\frac{P(T^+ \mid D^+)P(D^+)}{P(T^+)} &> 0.5 \\
\frac{P(T^+ \mid D^+)P(D^+)}{P(T^+ \mid D^+)P(D^+) + P(T^+ \mid D^-)P(D^-)} &> 0.5 \\
\frac{x \cdot 1/1000}{x \cdot 1/1000 + y \cdot 999/1000} &> 0.5 \\
\frac{x}{x + 999y} &> 0.5 \\
x &> 0.5(x + 999y) \\
0.5x &> 0.5(999y) \\
x &> 999y
\end{aligned}$$

3. An archery target is made of 3 concentric circles of radii $1/\sqrt{3}$, 1 and $\sqrt{3}$ feet. Arrows striking within the inner circle are awarded 4 points, arrows within the middle ring are awarded 3 points, and arrows within the outer ring are awarded 2 points. Shots outside the target are awarded 0 points.

   Consider a random variable X, the distance of the strike from the center (in feet), and let the probability density function of X be

$$f(x) = \begin{cases} \frac{2}{\pi(1+x^2)} & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

What is the expected value of the score of a single strike?

Solution: The expected value is

$$\begin{aligned}
&\int_0^{1/\sqrt{3}} 4\frac{2}{\pi(1+x^2)}\,\mathrm{d}x + \int_{1/\sqrt{3}}^{1} 3\frac{2}{\pi(1+x^2)}\,\mathrm{d}x + \int_1^{\sqrt{3}} 2\frac{2}{\pi(1+x^2)}\,\mathrm{d}x \\
=\ & \frac{2}{\pi}\left[4\left(\arctan\frac{1}{\sqrt{3}} - \arctan 0\right) + 3\left(\arctan 1 - \arctan\frac{1}{\sqrt{3}}\right) + 2\left(\arctan\sqrt{3} - \arctan 1\right)\right] \\
=\ & \frac{13}{6}
\end{aligned}$$

4. Let $X \sim \text{Pois}(\lambda)$, $Y \sim \text{Pois}(\mu)$. Given that $X \perp\!\!\!\perp Y$, derive an expression for $\mathbb{P}(X = k \,|\, X + Y = n)$ where $k = 0, \ldots, n$. What well-known probability distribution is this? What are its parameters?

   Solution: To derive this conditional distribution, we can write

$$P(X = k | X + Y = n) = \frac{P(X = k \cap X + Y = n)}{P(X + Y = n)}$$

using the definition of conditional probability. The event $X = k \cap X + Y = n$ can equivalently be expressed as $X = k \cap Y = n - k$ and we can express this using that $X \perp\!\!\!\perp Y$, i.e.,

$$\begin{aligned}
P(X = k \cap Y = n - k) &= \frac{e^{-\lambda}\lambda^k}{k!}\frac{e^{-\mu}\mu^{n-k}}{(n-k)!} \\
&= \frac{1}{n!}e^{-(\lambda+\mu)}\binom{n}{k}\lambda^k\mu^{n-k}
\end{aligned}$$

Note that due to the additivity property of independent Poisson random variables, $X + Y \sim \text{Pois}(\lambda + \mu)$. Now, we note that we can use the law of total probability with the above to get an expression for the denominator

$$\begin{aligned}
P(X + Y = n) &= \sum_{k=0}^{n} P(X = k \cap Y = n - k) \\
&= \sum_{k=0}^{n} \frac{1}{n!}e^{-(\lambda+\mu)}\binom{n}{k}\lambda^k\mu^{n-k}
\end{aligned}$$

$$= \frac{1}{n!}e^{-(\lambda+\mu)}\sum_{k=0}^{n}\binom{n}{k}\lambda^k\mu^{n-k}$$
$$= \frac{1}{n!}e^{-(\lambda+\mu)}(\lambda+\mu)^n$$

where the last equality comes from binomial expansion. Lastly, we plug these in to get

$$P(X=k|X+Y=n) = \binom{n}{k}\frac{\lambda^k\mu^{n-k}}{(\lambda+\mu)^n}$$

This is exactly the PMF for a binomial distribution with parameters n and $p = \frac{\lambda}{\lambda+\mu}$.

5. Consider a coin that may be biased, where the probability of the coin landing heads on any single flip is θ. If the coin is flipped n times and heads is observed k times, what is the maximum likelihood estimate (MLE) of θ?

Solution: The likelihood function for observing k heads in n independent flips is given by the binomial distribution:

$$L(\theta) = \binom{n}{k}\theta^k(1-\theta)^{n-k} \tag{2}$$

The log likelihood is accordingly

$$\log L(\theta) = \log\binom{n}{k} + k\log\theta + (n-k)\log(1-\theta)$$

To find the MLE estimate of θ, we take the derivative of $L(\theta)$ with respect to θ and set it equal to 0:

$$\begin{aligned}
\frac{\partial \log L(\theta)}{\partial\theta} &= 0\\
\frac{k}{\theta} - \frac{n-k}{1-\theta} &= 0\\
\frac{k}{\theta} &= \frac{n-k}{1-\theta}\\
k(1-\theta) &= \theta(n-k)\\
k - k\theta &= n\theta - k\theta\\
k &= n\theta\\
\theta &= \frac{k}{n}
\end{aligned}$$

To ensure that this is a maximum, we can check that the second derivative of the log-likelihood function is concave:

$$\frac{\partial^2 \log L(\theta)}{\partial\theta^2} = -\frac{k}{\theta^2} - \frac{n-k}{(1-\theta)^2}$$

Because $k \geq 0$ and $n-k \geq 0$, for all $\theta \in [0,1]$, all terms are negative and therefore the second derivative is negative. Therefore, setting the first derivative equal to 0 did give us the maximum as desired.

6. Consider a family of distributions parameterized by $\theta \in \mathbb{R}$ with the following probability density function:

$$f_\theta(x) = \begin{cases} e^{\theta - x} & \text{when } x \geq \theta \\ 0 & \text{when } x < \theta \end{cases}$$

(a) Prove that f is a valid probability density function by showing that it integrates to 1 for all θ.

Solution:

$$\begin{aligned} \int_{-\infty}^{\infty} f_\theta(x)\,dx &= \int_{-\infty}^{\theta} f_\theta(x)\,dx + \int_{\theta}^{\infty} f_\theta(x)\,dx \\ &= \int_{-\infty}^{\theta} 0\,dx + \int_{\theta}^{\infty} e^{\theta-x}\,dx \\ &= -e^{\theta-x}\Big|_{\theta}^{\infty} \\ &= 0 - (-1) \\ &= 1 \end{aligned}$$

(b) Suppose that you observe n samples distributed according to f: $x_1, x_2, \ldots, x_n$. Find the maximum likelihood estimate of θ.

Solution: We can find the MLE estimate of θ by maximizing the likelihood function:

$$\max_\theta L(\theta \mid x_1, \ldots, x_n) = \max_\theta \prod_{i=1}^{n} 1\{x_i \geq \theta\} e^{\theta - x_i}$$

Recall that if $x_i < \theta$ then $f_\theta(x) = 0$. Therefore, in order for the likelihood to be greater than 0, we require that $x_i \geq \theta$ for all i, or more succinctly $\theta \leq \min(x_1, \ldots, x_n)$. Therefore, we can equivalently write the optimization problem as

$$\max_{\theta \leq \min(x_1,\ldots,x_n)} \prod_{i=1}^{n} e^{\theta - x_i} \tag{3}$$

We can now maximize the log of the above expression instead to get

$$\max_{\theta \leq \min(x_1,\ldots,x_n)} \sum_{i=1}^{n} \theta - x_i = \max_{\theta \leq \min(x_1,\ldots,x_n)} n\theta - \sum_{i=1}^{n} x_i$$

Since $n\theta - \sum_{i=1}^{n} x_i$ is an increasing function with respect to θ, it will be maximized when θ is as large as possible with respect to the optimization constraints. Accordingly, the optimal value of θ (and therefore the MLE estimate) is $\min(x_1, \ldots, x_n)$.

5 The Multivariate Normal Distribution

The multivariate normal distribution with mean $\mu \in \mathbb{R}^d$ and positive definite covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, denoted $\mathcal{N}(\mu, \Sigma)$, has the probability density function

$$f(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right).$$

Here $|\Sigma|$ denotes the determinant of Σ. You may use the following facts without proof.

- The volume under the normal PDF is 1.

$$\int_{\mathbb{R}^d} f(x)\,\mathrm{d}x = \int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right\} \mathrm{d}x = 1.$$

- The change-of-variables formula for integrals: let f be a smooth function from $\mathbb{R}^d \to \mathbb{R}$, let $A \in \mathbb{R}^{d\times d}$ be an invertible matrix, and let $b \in \mathbb{R}^d$ be a vector. Then, performing the change of variables $x \mapsto z = Ax + b$,

$$\int_{\mathbb{R}^d} f(x)\,\mathrm{d}x = \int_{\mathbb{R}^d} f(A^{-1}z - A^{-1}b)\,|A^{-1}|\,\mathrm{d}z.$$

All throughout this question, we take $X \sim \mathcal{N}(\mu, \Sigma)$.

1. Use a suitable change of variables to show that $\mathbb{E}[X] = \mu$. You must utilize the definition of expectation.

   Solution: With the change of variables $x \mapsto z = x - \mu$,

$$\begin{aligned}\mathbb{E}[X] &= \int_{\mathbb{R}^d} \frac{x}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right\} \mathrm{d}x \\ &= \int_{\mathbb{R}^d} \frac{\mu + z}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}z^\top\Sigma^{-1}z\right\} \mathrm{d}z \\ &= \mu \int_{\mathbb{R}^d} f(z;0,\Sigma)\,\mathrm{d}z + \int_{\mathbb{R}^d} z f(z;0,\Sigma)\,\mathrm{d}z = \mu + 0 = \mu.\end{aligned}$$

   The left integral contains the normal PDF, whose integral is 1. The right integral integrates to 0 because $zf(z;0,\Sigma)$ is an odd function: the contributions of z and $-z$ cancel each other out.

2. Use a suitable change of variables to show that $\mathrm{Var}(X) = \Sigma$, where the variance of a vector-valued random variable X is

$$\mathrm{Var}(X) = \mathrm{Cov}(X, X) = \mathbb{E}[(X-\mu)\,(X-\mu)^\top] = \mathbb{E}[XX^\top] - \mu\,\mu^\top.$$

   Hints: Every symmetric, positive definite matrix Σ has a symmetric, positive definite square root $\Sigma^{1/2}$ such that $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$. Note that Σ and $\Sigma^{1/2}$ are invertible. After the change of variables, you will have to find another variance Var(Z); if you've chosen the right change of variables, you can solve that by solving the integral for each diagonal component of Var(Z) and a second integral for each off-diagonal component. The diagonal components will require integration by parts. You **cannot** assume anything about Var(Z)–you must compute it via integration.

Solution: With the change of variables $x \mapsto z = \Sigma^{-1/2}(x-\mu)$,

$$\begin{aligned}
\mathrm{Var}(X) &= \int_{\mathbb{R}^d} (x-\mu)(x-\mu)^\top \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right\} \mathrm{d}x \\
&= \int_{\mathbb{R}^d} \Sigma^{1/2} z z^\top \Sigma^{1/2} \frac{1}{\sqrt{(2\pi)^d}} \exp\left\{\frac{1}{2} z^\top z\right\} \mathrm{d}z \\
&= \mathbb{E}_{Z\sim\mathcal{N}(0,I)}[\Sigma^{1/2} Z Z^\top \Sigma^{1/2}] = \Sigma^{1/2}\mathbb{E}_{Z\sim\mathcal{N}(0,I)}[ZZ^\top]\Sigma^{1/2} \\
&= \Sigma^{1/2}\mathrm{Var}(Z)\Sigma^{1/2}.
\end{aligned}$$

Thus, if we can show that $\mathrm{Var}(Z) = I$ where $Z \sim \mathcal{N}(0, I)$, we are done. To that end, we can compute the diagonal and off-diagonal components separately:

- For $i \neq j$, we have
$$\int_{\mathbb{R}^d} z_i z_j \frac{1}{\sqrt{(2\pi)^d}} e^{-\frac{1}{2}z^\top z} \mathrm{d}z = \frac{1}{2\pi}\int_{-\infty}^{\infty} z_i e^{-\frac{1}{2}z_i^2}\,\mathrm{d}z_i \int_{-\infty}^{\infty} z_j e^{-\frac{1}{2}z_j^2}\mathrm{d}z_j = 0.$$
- For $i = j$, we have
$$\int_{\mathbb{R}^d} z_i^2 \frac{1}{\sqrt{(2\pi)^d}} e^{-\frac{1}{2}z^\top z} \mathrm{d}z = \int_{-\infty}^{\infty} z_i^2 \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_i^2} \mathrm{d}z_i = \left[-z_i \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_i^2}\right]_{-\infty}^{\infty} + \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z_i^2}\,\mathrm{d}z_i = 1$$
where we used integration by parts in the second equality.

This suffices to prove that $\mathrm{Var}(Z) = I$. Therefore, $\mathrm{Var}(X) = \Sigma$.

3. Compute the moment generating function (MGF) of X: $M_X(\lambda) = \mathbb{E}[e^{\lambda^\top X}]$, where $\lambda \in \mathbb{R}^d$. Note: moment generating functions have several interesting and useful properties, one being that M_X characterizes the distribution of X: if $M_X = M_Y$, then X and Y have the same distribution.

   Hints:

   - You should try "completing the square" in the exponent of the Gaussian PDF.
   - You should arrive at
   $$M_X(\lambda) = \exp\left\{\lambda^\top\mu + \frac{1}{2}\lambda^\top\Sigma\lambda\right\}.$$

   Solution: Notice that by completing the square,
   $$-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu) + \lambda^\top x = -\frac{1}{2}(x-\mu-\Sigma\lambda)^\top\Sigma^{-1}(x-\mu-\Sigma\lambda) + \mu^\top\lambda + \frac{1}{2}\lambda^\top\Sigma\lambda.$$

Therefore,

$$\begin{aligned}
\mathbb{E}\left[\exp\left\{\lambda^\top X\right\}\right] &= \frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\int_{\mathbb{R}^d}\exp\left\{\lambda^\top x\right\}\exp\left\{-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right\}\,\mathrm{d}x \\
&= \frac{\exp\left\{\mu^\top\lambda+\frac{1}{2}\lambda^\top\Sigma\lambda\right\}}{\sqrt{(2\pi)^d|\Sigma|}}\int_{\mathbb{R}^d}\exp\left\{-\frac{1}{2}(x-\mu-\Sigma\lambda)^\top\Sigma^{-1}(x-\mu-\Sigma\lambda)\right\}\,\mathrm{d}x \\
&= \exp\left\{\lambda^\top\mu+\frac{1}{2}\lambda^\top\Sigma\lambda\right\}.
\end{aligned}$$

4. Using the fact that MGFs determine distributions, given $A \in \mathbb{R}^{k\times d}$ and $b \in \mathbb{R}^k$ identify the distribution of $AX + b$ (don't worry about covariance matrices being invertible).

Solution: We compute the MGF of $AX + b$:

$$\mathbb{E}e^{\lambda^\top(AX+b)} = e^{\lambda^\top b}\,\mathbb{E}e^{(A^\top\lambda)^\top X} = \exp\left\{\lambda^\top b+\lambda^\top A\mu+\frac{1}{2}\lambda^\top A\Sigma A^\top\lambda\right\}.$$

Thus, $AX + b \sim \mathcal{N}(A\mu + b, A\Sigma A^\top)$. In other words, an affine transformation of a multivariate Gaussian returns another Gaussian.

5. Show that there exists an affine transformation of X that is distributed as the standard multivariate Gaussian, $\mathcal{N}(0, I_d)$. (Assume Σ is invertible.)

Solution: Consider $Z = \Sigma^{-1/2}(X-\mu)$. Then, by the above part, $Z \sim \mathcal{N}(0, I_d)$.

6 Real Analysis

1. **Limit of a Sequence.** A sequence $\{x_n\}$ is said to converge to a limit L if, for every measure of closeness $\epsilon \in \mathbb{R}$, the sequence's terms $n \in \mathbb{N}$ after a point $n_0 \in \mathbb{N}$ converge upon that limit. More formally, if $\lim_{n\to\infty} x_n = L$ then $\forall \epsilon > 0, \exists n_0 \in \mathbb{Z}^+$ such that $\forall n \geq n_0$:

$$|x_n - L| < \epsilon$$

   (a) Consider the sequence $\{x_n\}$ defined by the recurrence relation $x_{n+1} = \frac{1}{2}x_n$. Treat x_0 as some constant that is the first element of the sequence. Prove that $\{x_n\}$ converges by evaluating $\lim_{n\to\infty} x_n$. **You must use the formal definition of the limit of a sequence.**

   Solution: We can re-write the sequence as $x_n = \frac{x_0}{2^n}$. Seeing the exponential decay, it makes sense to guess that $L = \lim_{n\to\infty} x_n = 0$.
   Let $\epsilon > 0$. We now need to find n_0 such that $\forall n \geq n_0$,

$$\left|\frac{x_0}{2^n} - 0\right| < \epsilon$$

   We find $n > \log_2 \frac{x_0}{\epsilon}$, so we set $n_0 = \max\left\{\left\lceil \log_2 \frac{x_0}{\epsilon}\right\rceil, 0\right\}$. Since we have found a n_0 for each ϵ such that the bound is met for all $n \geq n_0$, we have proven that $\lim_{n\to\infty} x_n = 0$ (and hence the sequence converges).
   (Note that there are many correct choices of n_0. The one that we have provided is the most "conservative," but any alternate solution n_0^* is valid as long as $\forall \epsilon > 0, n_0^* \geq n_0$.

   (b) **[OPTIONAL]** Consider a sequence $\{x_n\}$ of non-zero real numbers and suppose that $L = \lim_{n\to\infty} n\left(1 - \frac{|x_{n+1}|}{|x_n|}\right)$ exists. Prove that $\{|x_n|\}$ converges when $L > 1$ by evaluating $\lim_{n\to\infty} |x_n|$.
   Hint: Use the Bernoulli Inequality $1 - \frac{q}{n} < \exp\left(-\frac{q}{n}\right)$ and the approximation for the Harmonic Series $\sum_{k=1}^{n} \frac{1}{k} \approx \ln(n)$ for sufficiently large n.

   Solution: Consider a real number $1 < q < L$. Since $L > q$, $\lim_{n\to\infty} n\left(1 - \frac{|x_{n+1}|}{|x_n|}\right) > q$. For sufficiently large n:

$$n\left(1 - \frac{|x_{n+1}|}{|x_n|}\right) > q$$
$$1 - \frac{|x_{n+1}|}{|x_n|} > \frac{q}{n}$$
$$\frac{|x_{n+1}|}{|x_n|} < 1 - \frac{q}{n}$$

   Using the Bernoulli Inequality:

$$\frac{|x_{n+1}|}{|x_n|} < \exp\left(-\frac{q}{n}\right)$$

   We can get the cumulative product of $\frac{|x_{n+1}|}{|x_n|}$ starting from the starting index n_0, where for all $n \geq n_0$ the distance to our limit is bounded by ϵ:

$$\frac{|x_{n+1}|}{|x_n|} \cdot \frac{|x_n|}{|x_{n-1}|} \cdot \ldots \cdot \frac{|x_{n_0+1}|}{|x_{n_0}|} < \exp\left(-\frac{q}{n}\right) \cdot \exp\left(-\frac{q}{n-1}\right) \cdot \ldots \cdot \exp\left(-\frac{q}{n_0}\right).$$

$$\frac{|x_{n+1}|}{|x_{n_0}|} < \exp\left(-q\sum_{k=n_0}^{n}\frac{1}{k}\right)$$
$$|x_{n+1}| < |x_{n_0}|\exp\left(-q\sum_{k=n_0}^{n}\frac{1}{k}\right)$$

Using the approximation for the Harmonic Series:

$$|x_{n+1}| < |x_{n_0}|\exp\left(-q\ln(n)\right)$$
$$|x_{n+1}| < |x_{n_0}|\frac{1}{n^q}$$

We can treat the starting term $|x_{n_0}|$ as a constant C, where the upper bound of the following terms change according to the preceding inequality. Since $q > 1$, this upper bound decays polynomially, meaning that

$$\lim_{n\to\infty}|x_n| = \frac{1}{n^q} = 0$$

2. **Taylor Series.** Taylor series expansions are a method of approximating a function near a point using polynomial terms. The Taylor expansion for a function $f(x)$ at point a is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

This can also be rewritten as $f(x) = \sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n$.

(a) Calculate the first three terms of the Taylor series for $f(x) = \ln(1+x)$ centered at $a = 0$.

Solution: The first three terms $f^{(n)}(x)$ are:

$$f(x) = \ln(1+x)$$
$$f'(x) = \frac{1}{1+x}$$
$$f''(x) = -\frac{1}{(1+x)^2}$$

The Taylor series approximation is therefore:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2$$
$$f(x) \approx \ln(1+a) + \frac{1}{(1+a)}(x-a) - \frac{1}{2}\frac{1}{(1+a)^2}(x-a)^2$$

At the point $a = 0$:

$$f(x) \approx x - \frac{1}{2}x^2$$

(b) **[OPTIONAL]** The gamma function is defined as

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt.$$

Calculate and use a first-order Taylor expansion of the gamma function centered at 1 to approximate $\Gamma(1.1)$. You should express your answer in terms of the Euler-Mascheroni constant γ.

You may use the fact that $\Gamma(x+1)$ interpolates the factorial function without proof.

Solution:

We first calculate $\Gamma(1) = 0! = 1$.

Now we proceed to compute $\Gamma'(1)$.

$$\Gamma'(z) = \frac{d}{dz}\int_0^\infty t^{z-1}e^{-t}\,dt = \int_0^\infty \frac{d}{dz} t^{z-1}e^{-t}\,dt = \int_0^\infty t^{z-1}e^{-t}\log t\,dt$$

Plugging in $z = 1$, we find $\Gamma'(1) = -\gamma$. Now, using our linear approximation, we obtain

$$\Gamma'(1.1) \approx \Gamma(1) + \Gamma'(1)\cdot 0.1 = 1 - \frac{\gamma}{10}$$

3. Consider a twice continuously differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$. Suppose this function admits a unique global optimum $x^* \in \mathbb{R}^n$. Suppose that for some spherical region $\mathcal{X} = \{x \mid \|x - x^*\|^2 \le D\}$ around x^* for some constant D, the Hessian matrix H of the function $f(x)$ is PSD and its maximum eigenvalue is 1. Prove that

$$f(x) - f(x^*) \le \frac{D}{2}$$

for every $x \in \mathcal{X}$. *Hint*: Look up Taylor's Theorem with Remainder. Use Mean Value Theorem on the second order term instead of the first order term, which is what is usually done.

Solution: Start by doing a Taylor expansion at x^* and use the mean value theorem:

$$f(x) = f(x^*) + \nabla f(x^*)^\top (x - x^*) + \frac{1}{2}(x - x^*)^\top \nabla^2 f(tx + (1-t)x^*)(x - x^*),$$

where $t \in [0,1]$ is a constant by mean value theorem. We denote $\hat{x} \triangleq tx + (1-t)x^*$, and we know that $\hat{x} \in \mathcal{X}$ since $\hat{x}$ is on the line segment connecting x and x^*, making it also in $\mathcal{X}$ since the region is spherical (In more mathematical language, we say that any convex combination between points in a convex set must lie within that convex set). Therefore we know that $\lambda_{\max}(\nabla^2 f(tx + (1-t)x^*)) = 1$. Thus,

$$(x - x^*)^\top \nabla^2 f(tx + (1-t)x^*)(x - x^*) \le \|x - x^*\|^2.$$

Combining with the fact that $\nabla f(x^*) = 0$ gets:

$$f(x) - f(x^*) \le \frac{1}{2}\|x - x^*\|^2 \le \frac{D}{2}.$$

7 Hands-on with data

In the following problem, you will use two simple datasets to walk through the steps of a standard machine learning workflow: inspecting your data, choosing a model, implementing it, and verifying its accuracy. We have provided two datasets in the form of numpy arrays: `dataset_1.npy` and `dataset_2.npy`. You can load each using NumPy's `np.load` method[4]. You can plot figures using Matplotlib's `plt.plot` method[5].

Each dataset is a two-column array with the first column consisting of n scalar inputs $X \in \mathbb{R}^{n\times 1}$ and the second column consisting of n scalar labels $Y \in \mathbb{R}^{n\times 1}$. We denote each entry of X and Y with subscripts:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

and assume that y_i is a (potentially stochastic) function of x_i.

(a) It is often useful to visually inspect your data and calculate simple statistics; this can detect dataset corruptions or inform your method. For both datasets:

  (i) Plot the data as a scatter plot.

  (ii) Calculate the correlation coefficient between X and Y:

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

  in which $\mathrm{Cov}(X, Y)$ is the covariance between X and Y and σ_X is the standard deviation of X.

  Your solution may make use of the NumPy library only for arithmetic operations, matrix-vector or matrix-matrix multiplications, matrix inversion, and elementwise exponentiation. It may not make use of library calls for calculating means, standard deviations, or the correlation coefficient itself directly.

(b) We would like to design a function that can predict y_i given x_i and then apply it to new inputs. This is a recurring theme in machine learning, and you will soon learn about a general-purpose framework for thinking about such problems. As a preview, we will now explore one of the simplest instantiations of this idea using the class of linear functions:

$$\hat{Y} = Xw. \tag{4}$$

  The parameters of our function are denoted by $w \in \mathbb{R}$. It is common to denote predicted variants of quantities with a hat, so $\hat{Y}$ is a predicted label whereas Y is a ground truth label.

[4]https://numpy.org/doc

[5]https://matplotlib.org/stable/users/explain/quick_start.html

We would like to find a w^* that minimizes the **squared error** $\mathcal{J}_{\text{SE}}$ between predictions and labels:

$$w^* = \underset{w}{\operatorname{argmin}}\ \mathcal{J}_{\text{SE}}(w) = \underset{w}{\operatorname{argmin}}\ \|Xw - Y\|_2^2.$$

Derive $\nabla_w \mathcal{J}_{\text{SE}}(w)$ and set it equal to 0 to solve for w^*. (Note that this procedure for finding an optimum relies on the convexity of $\mathcal{J}_{\text{SE}}$. You do not need to show convexity here, but it is a useful exercise to convince yourself this is valid.)

(c) Your solution w^* should be a function of X and Y. Implement it and report its **mean squared error** (MSE) for **dataset 1**. The mean squared error is the objective $\mathcal{J}_{\text{SE}}$ from part (b) divided by the number of datapoints:

$$\mathcal{J}_{\text{MSE}}(w) = \frac{1}{n}\|Xw - Y\|_2^2.$$

Also visually inspect the model's quality by plotting a line plot of predicted $\hat{y}$ for uniformly-spaced $x \in [0, 10]$. Keep the scatter plot from part (a) in the background so that you can compare the raw data to your linear function. Does the function provide a good fit of the data? Why or why not?

(d) We are now going to experiment with constructing new *features* for our model. That is, instead of considering models that are linear in the inputs, we will now consider models that are linear in some (potentially nonlinear) transformation of the data:

$$\hat{Y} = \Phi w = \begin{bmatrix} \phi(x_1)^\top \\ \phi(x_2)^\top \\ \vdots \\ \phi(x_n)^\top \end{bmatrix} w,$$

where $\phi(x_i), w \in \mathbb{R}^m$. Repeat part (c), providing both the mean squared error of your predictor and a plot of its predictions, for the following features on **dataset 1**:

$$\phi(x_i) = \begin{bmatrix} x_i \\ 1 \end{bmatrix}.$$

How do the plotted function and mean squared error compare? (A single sentence will suffice.)

Hint: the general form of your solution for w^* is still valid, but you will now need to use features Φ where you once used raw inputs X.

(e) Now consider the quadratic features:

$$\phi(x_i) = \begin{bmatrix} x_i^2 \\ x_i \\ 1 \end{bmatrix}.$$

Repeat part (c) with these features on **dataset 1**, once again providing short commentary on any changes.

(f) Repeat parts (c)-(e) with **dataset 2**.

(g) Finally, we would like to understand which features Φ provide us with the best model. To that end, you will implement a method known as k-fold cross validation. The following are instructions for this method; deliverables for part (g) are at the end.

(i) Split **dataset 2** randomly into $k = 4$ equal sized subsets. Group the dataset into 4 distinct training / validation splits by denoting each subset as the validation set and the remaining subsets as the training set for that split.

(ii) On each of the 4 training / validation splits, fit linear models using the following 5 polynomial feature sets:

$$\phi_1(x_i) = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \quad \phi_2(x_i) = \begin{bmatrix} x_i^2 \\ x_i \\ 1 \end{bmatrix} \quad \phi_3(x_i) = \begin{bmatrix} x_i^3 \\ x_i^2 \\ x_i \\ 1 \end{bmatrix} \quad \phi_4(x_i) = \begin{bmatrix} x_i^4 \\ x_i^3 \\ x_i^2 \\ x_i \\ 1 \end{bmatrix} \quad \phi_5(x_i) = \begin{bmatrix} x_i^5 \\ x_i^4 \\ x_i^3 \\ x_i^2 \\ x_i \\ 1 \end{bmatrix}$$

This step will produce 20 distinct w^* vectors: one for each dataset split and featurization ϕ_j.

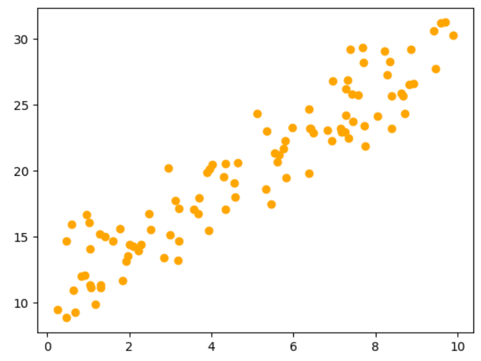
(iii) For each feature set ϕ_j, average the training and validation mean squared errors over all training splits.

It is worth thinking about what this extra effort has bought us: by splitting the dataset into subsets, we were able to use all available datapoints for model fitting while still having held-out datapoints for evaluation for any particular model.

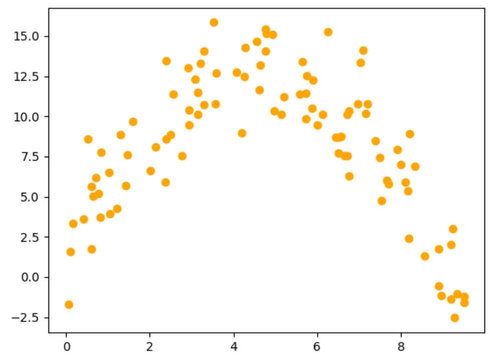
Deliverables for part (g): Plot the training mean squared error and the validation mean squared error on the same plot as a function of the largest exponent in the feature set. Use a log scale for the y-axis. Which model does the training mean squared error suggest is best? Which model does the validation mean squared error suggest is best?

Solution:

(a) Correlation coefficient of dataset 1: 0.939; dataset 2: -0.179.



(a)



(b)

Figure 1: Dataset plots

(b)

$$\begin{aligned}
\nabla_w \|\hat{Y} - Y\|_2^2 &= \nabla_w \frac{1}{n} \|Xw - Y\|_2^2 \\
&= \nabla_w (Xw - Y)^\top (Xw - Y) \\
&= \nabla_w w^\top X^\top Xw - 2w^\top X^\top Y + Y^\top Y \\
&= 2\left(X^\top Xw - X^\top Y\right).
\end{aligned}$$

Setting the gradient equal to zero and solving for w^* gives:

$$\begin{aligned}
2\left(X^\top Xw^* - X^\top Y\right) &= 0 \\
\Rightarrow X^\top Xw^* &= X^\top Y \\
\Rightarrow w^* &= (X^\top X)^{-1} X^\top Y.
\end{aligned}$$

(c) MSE: 32.027. The fit is poor because our parameterization does not have an offset, so the function must pass through the origin even though the data do not.

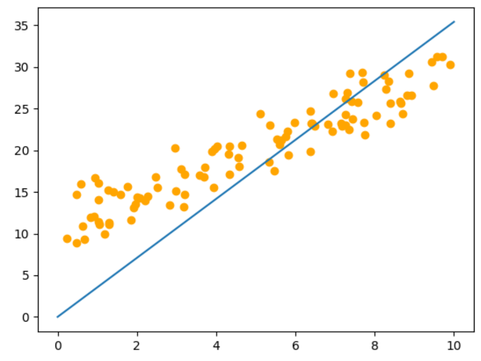


Figure 2: Dataset 1; raw features

(d) MSE: 4.020. The fit is now much better (both quantitatively and visually) because the function has an offset.

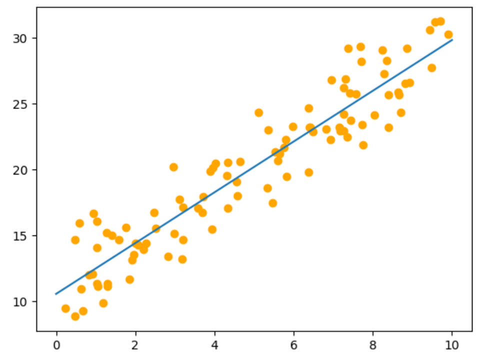


Figure 3: Dataset 1; offset features

(e) MSE: 4.009. There are no new meaningful changes because the function class is already expressive enough to capture the deterministic parts of our the data. (**Also acceptable:** we are now adding unneeded features that increase the expressivity of the function class, so our function is beginning to overfit slightly.)

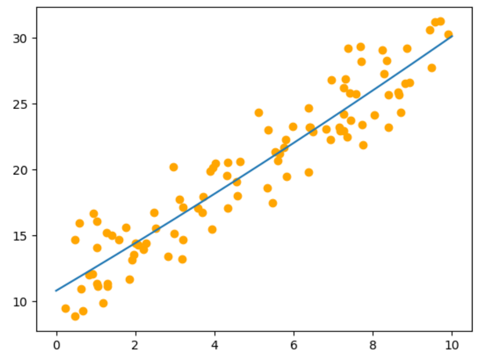


Figure 4: Dataset 1; quadratic features

(f) MSE for raw features: 42.556; for offset features: 19.807; for quadratic features: 4.287. Because the underlying data is quadratic, we now need the second-order features to fit the data well.

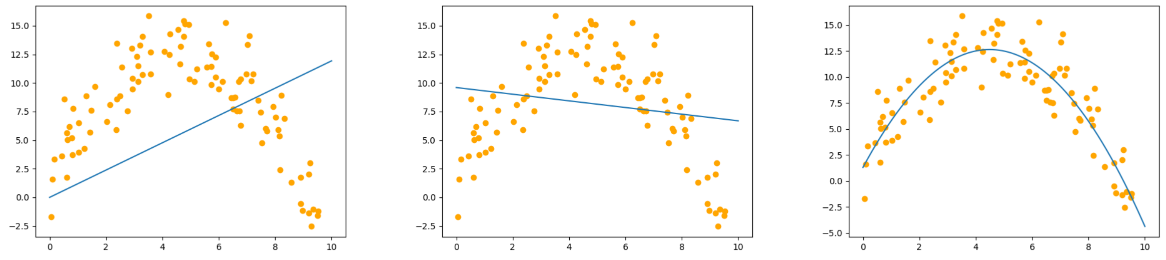


Figure 5: Dataset 2; raw, offset, and quadratic features

(g) The training MSE decreases with more features, so suggests that the quintic features are best. The validation MSE is lowest when the model class matches the underlying data-generating process, so suggests that the quadratic features are best. The validation MSE trend suggests that higher-order features lead to overfitting.

Grading note: There is more stochasticity in this solution because of the randomness introduced by dataset splitting. As long as the answer mentions training loss improving more than validation loss with more features, the answer should be counted as correct.

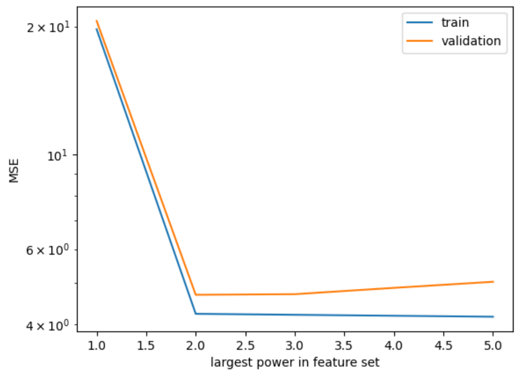


Figure 6: 4-fold cross-validation

```
import numpy as np
import functools
import matplotlib.pyplot as plt
import pdb

def plot(X, Y, X_linspace=None, Y_linspace=None, savepath=''):
    plt.clf()
    plt.plot()
    plt.scatter(X, Y, c='orange')
    if X_linspace is not None and Y_linspace is not None:
        plt.plot(X_linspace, Y_linspace)
    plt.savefig(savepath)


def cov_fn(X, Y):
    assert len(X) == len(Y)
    X_mean = X.sum() / len(X)
    Y_mean = Y.sum() / len(Y)
    cov = ((X - X_mean) * (Y - Y_mean)).sum() / (len(X) - 1)
    return cov

def corrcoef(X, Y, validate=True):
    X, Y = X.squeeze(), Y.squeeze()
    numer = cov_fn(X, Y)
    denom = (cov_fn(X, X) * cov_fn(Y, Y))**.5
    corr = numer / denom

    if validate:
        corr_ = np.corrcoef(X, Y)[0,1]
        assert np.isclose(corr, corr_)
    return corr

def lstsq(A, b, validate=True):
    w = np.linalg.inv(A.T @ A) @ A.T @ b
    if validate:
        w_, *_ = np.linalg.lstsq(A, b, rcond=None)
        assert np.allclose(w, w_)
    return w

def mse_fn(preds, targets, validate=True):
    mse = ((preds - targets)**2).sum() / len(preds)
    if validate:
        mse_ = np.linalg.norm(preds - targets, 2)**2 / len(preds)
        assert np.isclose(mse, mse_)
    return mse

def make_poly_features(X, p=2):
    phi = np.concatenate([
        X**power for power in range(p+1)
    ], axis=-1)
    return phi

def k_fold(X, Y, k):
    assert len(X) == len(Y)
    split_size = len(X) // k
    indices = np.arange(len(X))
    ## shuffle in place
    np.random.shuffle(indices)
    for i in range(k):
        val_inds = indices[np.arange(split_size * i, split_size * (i+1))]
        train_inds = np.setdiff1d(np.arange(len(X)), val_inds)
        yield X[train_inds], Y[train_inds], X[val_inds], Y[val_inds]


featurizers = [
    lambda x: x,  # raw features
    functools.partial(make_poly_features, p=1),  # offset
    functools.partial(make_poly_features, p=2),  # quadratic
]

X_linspace = np.linspace(0, 10, 1000)[:, None]

for i in range(1, 3):

    with open(f'dataset_{i}.npy', 'rb') as f:
        Z = np.load(f)
    X, Y = Z[:,0:1], Z[:,1:2]

    ## plot raw data
    plot(X, Y, savepath=f'{i}_data.png')

    ## quick statistics
    corr = corrcoef(X, Y)
    print(f'Dataset {i} | Correlation coefficient: {corr:4f}')

    for j, featurizer in enumerate(featurizers):
        phi = featurizer(X)
        w = lstsq(phi, Y)

        preds = phi @ w
        mse = mse_fn(preds, Y)
        print(f'Dataset {i} | featurizer {j} | MSE: {mse:.4f}')

        ## use linspace for plotting
        phi_linspace = featurizer(X_linspace)
        preds_linspace = phi_linspace @ w
        plot(X, Y, X_linspace, preds_linspace, f'{i}_preds_{j}.png')

    print()

## cross validation on dataset 2

featurizers = [functools.partial(make_poly_features, p=p) for p in range(1, 6)]

k = 4
plot_train = []
plot_val = []
for j, featurizer in enumerate(featurizers):
    mse_trains = []
    mse_vals = []
    for X_train, Y_train, X_val, Y_val in k_fold(X, Y, k=k):
        phi_train = featurizer(X_train)
        phi_val = featurizer(X_val)
        w = lstsq(phi_train, Y_train)

        preds_train = phi_train @ w
        preds_val = phi_val @ w

        mse_train = mse_fn(preds_train, Y_train)
        mse_val = mse_fn(preds_val, Y_val)

        mse_trains.append(mse_train)
        mse_vals.append(mse_val)

    mse_train_mean = np.mean(mse_trains)
    mse_val_mean = np.mean(mse_vals)
    plot_train.append(mse_train_mean)
    plot_val.append(mse_val_mean)

    print(f'featurizer {j} | split {k} | MSE train: {mse_train_mean:.4f} | MSE val: {mse_val_mean:.4f}')


plt.clf()
plt.plot(np.arange(1, 6), plot_train, label='train')
plt.plot(np.arange(1, 6), plot_val, label='validation')

plt.yscale('log')
plt.legend()
plt.ylabel('MSE')
plt.xlabel('largest power in feature set')
plt.savefig('cross_validation.png')
```

8 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*!