CS189Hw1

Trustin Nguyen

October 23, 2024

Linear Algebra Review

Exercise 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 $T : V \longrightarrow W$ be a linear map. Then the following are true:

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

(b) $\Im T$ is a subspace of W.

(c) $\Im T$ 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 by unfamiliar with the language, try looking up "isomorphism" and "quotient space.")

Proof. (Part A) By definition, if $k \in \ker T$, then $T(k) = 0 \in W$. Then for definition of a subspace:

- $T(k_1) + T(k_1) = 0 = T(k_1 + k_2)$. Since T is linear. It follows that $k_1 + k_2$ is also in $\ker T$.
- $rT(k_1) = T(rk_1) = 0$ also by the linearity of T. So rk_1 is also in $\ker T$ if $k_1 \in \ker T$.

We must have 0 in $\ker T$ because T is linear. So we are done.

(Part B) Same process as the last one:

- $w_1 + w_2 = T(v_1) + T(v_2) = T(v_1 + v_2)$, so $w_1 + w_2 \in \Im T$.
- $rw_1 = rT(v_1) = T(rv_1)$ so $rw_1 \in \Im T$.

We must have 0 in the image also because 0 is mapped to 0 by the linearity of T.

(Part C) We have that $V/\ker T$ are elements of the form $v + \ker T$ which are the cosets that partition the set. Then the isomorphism given by the mapping:

$$v + \ker T \in V/\ker T \rightarrow T(v) \in \Im T$$

This is injective because if $v_0 + \ker T, v_1 + \ker T$ were two different cosets, we have:

$$T(v_0 + \ker T) = T(v_0), T(v_1 + \ker T) = T(v_1)$$

It follows that:

$$T(v_0) = T(v_1) \implies T(v_0 - v_1) = 0, v_0 - v_1 \in \ker T$$

But this is a contradiction because then

$$v_1 + \ker T = v_1 + (v_0 - v_1) + \ker T = v_0 + \ker T$$

To see surjectivity, suppose that $w \in \Im T$. Then

$$T(v) = w$$

for some $v \in V$. Since the kernel cosets partitions V, $v = v' + \ker T$. It follows that $T(v') = w$ for some $v' \in \ker T$. □

Exercise 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 to transform $\begin{bmatrix} I_n & 0 \\ 0 & AB \end{bmatrix}$ to $\begin{bmatrix} B & I_n \\ 0 & A \end{bmatrix}$.

Answer.

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

Answer. Row operations preserve rank. So we see that the rank of the second matrix is $\mathrm{rank}B + \mathrm{rank}A - n$ while for the first matrix, it is $n + \mathrm{rank}AB$. So it follows that:

$$\mathrm{rank}A + \mathrm{rank}B - n \leqslant \mathrm{rank}AB$$

It is clear that $\mathrm{rank}AB \leqslant \min(\mathrm{rank}A, \mathrm{rank}B)$, because by the isomorphism theorem, the dimension of the image is less than or equal to the dimension of the domain. Then it follows that $\mathrm{rank}AB \leqslant \mathrm{rank}B$. We also know that the rank of AB is less than or equal to the rank of A because the matrix A in AB operates on a smaller subspace, and therefore its image is a subspace of the image of A.

(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.

Answer. Since a submatrix of A has non-zero determinant, there is a way to row-reduce A such that a submatrix is the identity matrix of dimension r. This means that the row space and column space have dimension at least r.

(d) It is a fact that $\mathrm{rank}(A^T A) = \mathrm{rank}A$; here's one way to see that. We've already seen in part (b) that $\mathrm{rank}(A^T A) \leqslant \mathrm{rank}A$. Suppose that $\mathrm{rank}(A^T A)$ were strictly less than $\mathrm{rank}A$. What would that tell us about the relationship between the column space of A and the null space of A^T? What standard fact about the fundamental subspaces of A says that relationship is impossible?

Answer. We can check the rank of $A^T A$ by multiplying it with the unit vectors that span the subspace. Then it is clear that none of the column vectors of A are in the null space of A^T. Therefore, we cannot have that $\mathrm{rank}A^T A < \mathrm{rank}A$.

(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?

Answer. $A\mathbb{R}^n$ gives the span of the column vectors of A. This is because it is the subspace spanned by the action of A on the generators of $\mathbb{R}^n$.

Exercise 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 this class, they are defined to be symmetric matrices. There are non-symmetric matrices that exhibit PSD properties, like the first definition below, but not all three.

(a) For all $x \in \mathbb{R}^n$, $x^T A x \geqslant 0$.

(b) All the eigenvalues of A are non-negative.

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

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

Proof. We start with $(a) \implies (b)$. Let x be an eigenvector of A. Then

$$x^T A x = \lambda x^T x \geqslant 0$$

clearly, $\lambda \geqslant 0$.

Now for $(b) \implies (c)$. By the spectral theorem, we can write $A = PDP^T$ where P are orthogonal matrices and D is a diagonal matrix containing the eigenvalues of A. Then:

$$A = PD^{1/2}D^{1/2}P^T = (PD^{1/2})(PD^{1/2})^T$$

For $(c) \implies (a)$, we have:

$$x^T U U^T x = (x^T U)(x^T U)^T \geqslant 0$$

since it becomes a dot product of $(x^T U)^T$ with itself. □

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

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

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

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

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

Answer. We have that the trace of $x^T A y$ is the same as the trace of

$$(x^T A y)^T = y^T (x^T A)^T = y^T A^T x$$

Now note that $\langle A, xy^T \rangle = \mathrm{trace}(A^T x y^T)$. $A^T x$ is a vector in $\mathbb{R}^n$. We see that $y^T A^T x$ is a dot product between y and $A^T x$. On the other hand, the diagonal of $A^T x$ times y^T gives us the summands of the dot product previously mentioned. Then it is clear that they are equivalent.

(b) If A and B are symmetric PSD matrices, then $\mathrm{trace}(AB) \geqslant 0$.

Answer. This means that $A = UU^T$, $B = VV^T$. Then

$$\mathrm{tr}(AB) = \mathrm{tr}(UU^T VV^T) = \mathrm{tr}(V^T U U^T V) = \mathrm{tr}((U^T V)^T (U^T V)) \geqslant 0$$

(c) Optional: If $A, B \in \mathbb{R}^{n \times n}$ are real symmetric matrices with $\lambda_{\max}(A) \geqslant 0$ and B being PSD, the $\langle A, B \rangle \leqslant \sqrt{n}\lambda_{\max}(A)\|B\|_F$.

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

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

Answer. We see that that:

$$\max_{u \in \mathbb{R}^m, v \in \mathbb{R}^n, \|u\|=1, \|v\|=1} (u^T A v) = \sqrt{\max_{u \in \mathbb{R}^m, v \in \mathbb{R}^n, \|u\|=1, \|v\|=1} (u^T A v v^T A^T u)}$$

This simplifies to:

$$\sqrt{\max_{u \in \mathbb{R}^m, v \in \mathbb{R}^n, \|u\|=1, \|v\|=1} (u^T A A^T u)}$$

Now u can be written as a linear combination of the eigenbasis for $A^T A$. $A^T A$ acts linearly on u, so we see that the max is when u is an eigenvector corresponding to the maximum eigenvalue. This concludes the proof.

Matrix/Vector Calculus and Norms

Exercise 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

$$\sin A_{11}^2 + e^{A_{11}+A_{22}} + x^T A y$$

Answer. Applying dA to the whole thing:

$$\frac{d}{dA} \sin(A_{11}^2 + e^{A_{11}+A_{22}}) + \frac{d}{dA} x^T A y$$

We get:

$$\begin{bmatrix} \frac{d}{dA_{11}} \sin(A_{11}^2 + e^{A_{11}+A_{22}}) & \frac{d}{dA_{12}} \sin(A_{11}^2 + e^{A_{11}+A_{22}}) \\ \frac{d}{dA_{21}} \sin(A_{11}^2 + e^{A_{11}+A_{22}}) & \frac{d}{dA_{22}} \sin(A_{11}^2 + e^{A_{11}+A_{22}}) \end{bmatrix} + \frac{d}{dA} x^T A y$$

For the left term, we get:

$$\begin{bmatrix} (2A_{11} + e^{A_{11}}) \cos A_{11}^2 + e^{A_{11}+A_{22}} & 0 \\ 0 & e^{A_{22}} \cos(A_{11}^2 + e^{A_{11}+A_{22}}) \end{bmatrix}$$

Now for the

$$\frac{d}{dA} x^T A y$$

term:

$$\begin{aligned} \frac{d}{dA} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} &= \frac{d}{dA} \left(\begin{bmatrix} x_1 & x_2 \end{bmatrix} \right) \begin{bmatrix} y_1 A_{11} + y_2 A_{12} \\ y_1 A_{21} + y_2 A_{22} \end{bmatrix} \\ &= \frac{d}{dA} x_1 y_1 A_{11} + x_1 y_2 A_{12} + x_2 y_1 A_{21} + x_2 y_2 A_{22} \end{aligned}$$

So the term is:

$$\begin{bmatrix} x_1 y_1 & x_1 y_2 \\ x_2 y_1 & x_2 y_2 \end{bmatrix}$$

Then the derivative is given by:

$$\begin{bmatrix} (2A_{11} + e^{A_{11}}) \cos(A_{11}^2 + e^{A_{11}+A_{22}}) & 0 \\ 0 & e^{A_{22}} \cos(A_{11}^2 + e^{A_{11}+A_{22}}) \end{bmatrix} + \begin{bmatrix} x_1 y_1 & x_1 y_2 \\ x_2 y_1 & x_2 y_2 \end{bmatrix}$$

Exercise 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) $\|A\|_2$

Answer. We can write it as:

$$\|A\|_p = \frac{x^T A^T A x}{x^T x}$$

Now since $A^T A$ is symmetric, we can diagonalize it. The supremum is them given by the max eigenvalue. So it is $\lambda_{\max}(A^T A)$

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

Answer. This would be the maximum over the sum over each column of the rows of A.

Exercise 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 $\frac{\partial \alpha}{\partial \beta_i}$?

Answer. The partials are $\frac{\partial \alpha}{\partial \beta_i} = \frac{y_i e^{\beta_i}}{1+e^{\beta_i}}$.

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

Answer. The partial is A. Let a_i be the column vectors of A. Then

$$Ax = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$$

Then the partial of a direct sum of the $a_i x_i$ is the same as the direct sum of the partials. So it is a direct sum of $a_1, \ldots, a_n$, which gives the matrix A.

(c) Given $z \in \mathbb{R}^m$, write the gradient $\nabla_z (z^T z)$.

Answer. Using the same reasoning as the last one, it would be $2z$

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

Answer. We would take $\frac{\partial}{\partial x_i} z^T z$ over i. This is equivalent to $\frac{\partial}{\partial z} z^T z \frac{\partial z}{\partial x_i} = 2z \frac{\partial z}{\partial x_i}$. Now we have $2z \frac{\partial z}{\partial x}$

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

Answer. We have:

$$\nabla_x(z^T z) = \nabla_x((Ax - y)^T (Ax - y)) = \nabla_x(x^T A^T A x - 2y^T A x + y^T y)$$

So we are reduced to

$$\nabla_x(x^T A^T A x) - 2A^T y$$

Now the general rule for $\nabla_x(x^T B x) = (B + B^T)x$, which can be shown by expanding and computation. Then

$$\nabla_x(x^T A^T A x) = (A^T A + A^T A)x = 2A^T A x$$

So

$$\nabla_x(z^T z) = 2A^T A x - 2Ay = 2A^T(Ax - y) = 2A^T z$$

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 such that

$$Y \approx XW^T$$

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*

$$\mathrm{RSS}(W) = \|XW^T - Y\|_F^2$$

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 $k \times d$ 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 $d_\theta = d_L d_{L-1} + d_{L-1} d_{L-2} + \cdots + d_1 d_0$ 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 image θ as a column vector: we take all the components of all the matrices $W_L, W_{L-1}, \ldots, W_1$ and write them all in one very long column vector. Given a fixed weight vector θ, the linear neural network takes an *input vector* $x \in \mathbb{R}^{d_\theta}$ 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 $\mathrm{RSS} \circ \mu$ - that is, it minimizes the cost function

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

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

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

   prove this fact by deriving a formula for each G_{ij} using summations, simplified as much as possible.

Answer. We know that the Frobenius norm is also given by the square root of the trace of the matrix times the transpose:

$$\mathrm{RSS}(W) = \mathrm{tr}((WX^T - Y^T)(XW^T - Y))$$

Now we break it down:

$$X = \begin{bmatrix} | & | & & | \\ x_0 & x_1 & \cdots & x_d \\ | & | & & | \end{bmatrix}$$

$$Y = \begin{bmatrix} | & | & & | \\ y_0 & y_1 & \cdots & y_k \\ | & | & & | \end{bmatrix}$$

We see that a given column of $XW^T - Y$ is given by

$$\sum_{i\geqslant 0} x_i w_{j,i} - y_j$$

for the j-th column. The trace gives us a sum of dot products:

$$\mathrm{RSS}(W) = \sum_j \left(\left(\sum_{i\geqslant 0} x_i w_{j,i} - y_j \right)^T \left(\sum_{i\geqslant 0} x_i w_{j,i} - y_j \right) \right)$$

Now if we only care about the partial wrt $w_{i,j}$, we only care about the term with $w_{j,i}$. I used $w_{j,i}$ to denote the element $w_{i,j}$ in the transpose matrix. So this reduces to

$$(x_i w_{j,i} - y_j)^T (x_i w_{j,i} - y_j) = \sum_{n\geqslant 0} (x_{n,i} w_{j,i} - y_{n,j})^2$$

Taking the partial wrt to $w_{i,j}$, we get:

$$\sum_{n\geqslant 0} 2x_{n,i}(x_{n,i} w_{j,i} - y_{n,i}) = 2(x_i w_{j,i} - y_i)^T x_i$$

So then the gradient is given by

$$2(XW^T - Y)^T X = 2(X^T W - Y^T)X$$

2. Directional derivatives are closely related to gradients. The notation $\mathrm{RSS}'_{\Delta\theta}(W)$ denotes the directional derivative of $\mathrm{RSS}(W)$ in the direction ΔW, and the notation $\mu'_{\Delta\theta}(\theta)$ denotes the directional derivative of $\mu(\theta)$ in the direction $\Delta\theta$. Informally speaking, the directional derivative $\mathrm{RSS}'_{\Delta W}(W)$ tells us how much $\mathrm{RSS}(W)$ 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; $\mathrm{RSS}'_{\Delta}(W)$ is a local linearization of the relationship between W and $\mathrm{RSS}(W)$ at W. To a physicist, $\mathrm{RSS}'_{\Delta W}(W)$ tells us the initial velocity of change of $\mathrm{RSS}(W)$ if we start changing W with velocity ΔW.)

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

   Answer. To get the directional derivative, we can take the dot product of the vectors in W with the direction of change ΔW. So the Frobenius inner product is

$$\langle 2(WX^T - Y^T)X, \Delta W \rangle_F$$

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.

Answer. We want to compute

$$\mu'_{\Delta\theta}(\theta) = \langle \nabla_\theta \mu(\theta), \Delta\theta \rangle_F$$

Now we can break this down component-wise:

$$\sum_{j \geqslant 0}^{L} \langle \nabla_{W_j} \mu(\theta), \Delta\theta \rangle_F$$

then:

$$\nabla_{W_j, \mu(\theta)} = \frac{\partial}{\partial W_j}(W_L W_{L-1} \cdots W_2 W_1) = W_L W_{L-1} \cdots W_{j+1} \nabla W_j W_{j-1} \cdots W_1$$

We can see this by taking multiple product rules:

$$\frac{\partial}{\partial W_j} W_L W_{L-1} \cdots W_1 = \frac{\partial W_L}{\partial W_j} W_{L-1} W_{L-2} \cdots W_1 + W_L \cdot \frac{\partial}{\partial W_j} W_{L-2} \cdots W_1$$

and once we get to W_j:

$$W_L \cdots W_{j+1} \frac{\partial}{\partial W_j} W_j \cdots W_1 = W_L \cdots W_{j+1} (\nabla W_j W_{j-1} \cdots W_1 + \frac{\partial}{\partial W_j} W_{j-1} \cdots W_1)$$

So the answer is given by

$$\sum_{j \geqslant 0} \langle W_{>j} \nabla W_j W_{<j}, \Delta\theta \rangle_F = \sum_{j \geqslant 0}^{L} W_{>j} \Delta W_j W_{<j}$$

4. Recall the chain rule for scalar functions, $\frac{d}{dx} f(g(x))\big|_{x=x_0} = \frac{d}{dy} f(y)\big|_{y=g(x)} \cdot \frac{d}{dx} g(x)\big|_{x=x_0}$. 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 $J'_{\Delta\theta}(\theta)\big|_{\theta=\theta_0}$ 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 the those directional derivatives; we'll substitute the values in the next part.)

Answer. Using the chain rule:

$$J'_{\Delta\theta}(\theta)\bigg|_{\theta=\theta_0} = \mathsf{RSS}'_{\Delta\mu}(\mu(\theta))\bigg|_{\mu=\mu(\theta_0)} \cdot \mu'_{\Delta\theta}(\theta)\bigg|_{\theta=\theta_0}$$

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

$$\begin{aligned}\nabla_\theta J(\theta) = \; & (2(\mu(\theta)X^T - Y^T)XW^T_{<L}, \\ & \dots, \\ & 2W^T_{>j}(\mu(\theta)X^T - Y^T)XW^T_{<j}, \\ & \dots, \\ & 2W^T_{>1}(\mu(\theta)X^T - Y^T)X).\end{aligned}$$

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

Answer. Plugging in our results:

$$J'_{\Delta\theta}(\theta)\bigg|_{\theta=\theta_0} = \langle 2(WX^T - Y^T)X, \Delta W\rangle_F\bigg|_{W=\mu(\theta_0)} \cdot \sum_{j\geqslant 0}^{L} W_{>j}\Delta W_j W_{<j}$$

And

$$J'_{\Delta\theta}(\theta) = \langle 2(\mu(\theta)X^T - Y^T)X, \Delta\mu(\theta)\rangle_F \cdot \sum_{j\geqslant 0}^{L} W_{>j}\Delta W_j W_{<j}$$

Now to get the gradient:

$$J'_{\Delta\theta}(\theta) = \langle \nabla_\theta J(\theta), \Delta\theta\rangle$$

So:

$$\langle \nabla_\theta J(\theta), \Delta\theta\rangle = \langle 2(\mu(\theta)X^T - Y^T)X, \Delta\mu(\theta)\rangle_F \cdot \sum_{j\geqslant 0}^{L} W_{>j}\Delta W_j W_{<j}$$

Probability Potpourri

Exercise 1: Recall the covariance of two scalar random variables X and Y is defined as $Cov(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$. 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 $\Sigma_{ij} = Cov(Z_i, Z_j)$. Concisely, $\Sigma = \mathbb{E}[(Z - \mu)(Z - \mu)^T]$, where μ is the mean value of Z. Prove that the covariance matrix is always positive semi-definite.

Answer. We have that if x is a column vector:

$$x^T \Sigma x = x^T \mathbb{E}[(Z - \mu)(Z - \mu)^T]x$$

This gives:

$$\sum_i \sum_j \mathbb{E}[x_i(Z_i - \mu_i)]\mathbb{E}[(Z_j - \mu_j)x_j] = \sum_i \mathbb{E}[x_i(Z_i - \mu_i)] \sum_j \mathbb{E}[x_j(Z_j - \mu_j)]$$

Let $Y = \sum_j x_j Z_j$ so $\mathbb{E}[Y] = \sum_j x_j \mu_j$ and $Y - \mathbb{E}[Y] = \sum_j x_j Z_j - x_j \mu_j$. Then the equation up top tells us that $x^T \Sigma x = (Y - \mathbb{E}[Y])^2 = \sigma_Y^2$ or the variance of Y. This number is positive, so the covariance is semi-definite.

Exercise 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.

Answer. Let T_p denote probability of testing positive and $D, 1 - D$ denote probability of having the disease and probability of not having the disease. Then we have:

$$P(T_p \mid D) = x, P(T_p \mid 1 - D) = y$$

And we want to find:

$$P(D \mid T_p) = \frac{D \cap T_p}{T_p} > 0.5$$

We have:

$$D \cdot P(T_p \mid D) + (1 - D) \cdot P(T_p \mid 1 - D) = T_p = Dx + (1 - D)y$$

Also,

$$x = P(T_p \mid D) = \frac{T_p \cap D}{D} \implies xD = T_p \cap D$$

So we have in terms of x, y:

$$\frac{Dx}{Dx + (1 - D)y} > 0.5$$

Simplifying yields:

$$x > \frac{(1 - D)y}{D}$$

where $D = 1/1000$

Exercise 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

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

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

Answer. Let Y be the random variable for the score such that:

$$Y = g(X) = \begin{cases} 2 & \text{if } 0 \leqslant x \leqslant \dfrac{1}{\sqrt{3}} \\ 3 & \text{if } \dfrac{1}{\sqrt{3}} \leqslant x \leqslant 1 \\ 4 & \text{if } 1 \leqslant x \leqslant \sqrt{3} \\ 0 & \text{if } otherwise \end{cases}$$

Then

$$\mathbb{E}(Y) = \int p(g^{-1}(Y)) \cdot Y \, dY = \int_{-\infty}^{\infty} p(x) \cdot g(x) \, dx$$

This gives:

$$\begin{aligned}\mathbb{E}(Y) &= \int_0^{\frac{1}{\sqrt{3}}} \frac{2}{\pi(1+x^2)} g(x) \, dx + \int_{\frac{1}{\sqrt{3}}}^{1} \frac{2}{\pi(1+x^2)} g(x) \, dx + \int_1^{\sqrt{3}} \frac{2}{\pi(1+x^2)} g(x) \, dx \\ &= 4 \int_0^{\frac{1}{\sqrt{3}}} \frac{2}{\pi(1+x^2)} \, dx + 3 \int_{\frac{1}{\sqrt{3}}}^{1} \frac{2}{\pi(1+x^2)} \, dx + 2 \int_1^{\sqrt{3}} \frac{2}{\pi(1+x^2)} \, dx\end{aligned}$$

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

Answer. We see that:

$$\mathbb{P}(X = k \mid X + Y = n) = \frac{\mathbb{P}(X = k, X + Y = n)}{P(X + Y = n)} = \frac{\mathbb{P}(X = k)\mathbb{P}(Y = n - k)}{\mathbb{P}(X + Y = n)}$$

Solve for the denominator first:

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

Now for the numerator:

$$\mathbb{P}(X = k)\mathbb{P}(Y = n - k) = \frac{\lambda^k \mu^{n-k} e^{-(\lambda+\mu)}}{k!(n-k)!}$$

Putting this all together:

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

Then this is a binomial distribution with parameters $n, k, p = \frac{\lambda}{\mu+\lambda}$.

Exercise 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?

Answer. We have that:

$$\theta_{\text{MLE}} = \underset{\theta}{\operatorname{argmax}}\, p(D \mid \theta) = \underset{\theta}{\operatorname{argmax}} \prod p(x_i \mid \theta)$$

Since we rolled k heads, this is:

$$\underset{\theta}{\operatorname{argmax}}\, p(H \mid \theta)^k p(T \mid \theta)^{n-k} = \underset{\theta}{\operatorname{argmax}}\, \theta^k (1-\theta)^{n-k}$$

To find the maximum, we take the derivative with respect to θ:

$$\frac{d}{d\theta}\theta^k(1-\theta)^{n-k} = k\theta^{k-1}(1-\theta)^{n-k} - (n-k)\theta^k(1-\theta)^{n-k-1} = 0$$

Now simplify:

$$\begin{aligned}
0 &= k\theta^{k-1}(1-\theta)^{n-k} - (n-k)\theta^k(1-\theta)^{n-k-1} \\
0 &= \theta^{k-1}(1-\theta)^{n-k-1}(k(1-\theta) - (n-k)\theta) \\
0 &= k(1-\theta) - (n-k)\theta \\
0 &= k - k\theta - n\theta + k\theta \\
n\theta &= k \\
\theta &= \frac{k}{n}
\end{aligned}$$

So we conclude that the MLE of θ is $\frac{k}{n}$

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

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

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

Answer. Consider:

$$\int_{-\infty}^{\infty} f_\theta x \; dx$$

Then we can break it up as:

$$\int_{-\infty}^{\theta} f_\theta(x) \; dx + \int_{\theta}^{\infty} f_\theta(x) \; dx$$

Notice that $f_\theta(x) = 0$ when $x < \theta$, so the left integral is 0. So we just have:

$$\int_\theta^\infty e^{\theta - x}\, dx = \left(-e^{\theta-x}\right)\Big|_\theta^\infty = -e^{\theta-\infty} - (-e^{\theta-\theta}) = 0 + e^0 = 1$$

(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.

Answer. We have that:

$$\underset{\theta}{MLE} = \underset{\theta}{argmax}\, p(D, \theta) = \underset{\theta}{argmax} \prod_i p(x_i, \theta) = \underset{\theta}{argmax} \prod_i f_\theta(x_i)$$

Note that if any of $x_1, \ldots, x_n$ are 0, the expression evaluates to 0. So assume that none are 0. We are reduced to

$$\frac{d}{d\theta} \prod_i e^{\theta - x_i}$$

To make this easier, we can take the log MLE instead which is:

$$\frac{d}{d\theta} \sum_i \theta - x_i = n\theta - \sum_i x_i = n$$

We see that the derivative is positive. This means that if we wanted to minimize our original function, we want θ to be as minimal as possible. It follows that $\theta = \min(x_1, \ldots, x_n)$.

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

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

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

- The volume under the normal PDF is 1.

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

- 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 $x \mapsto z = Ax + b$,

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

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.

   Answer. By the definition of expectation,

$$\mathbb{E}[X] = \int_{\mathbb{R}^d} \frac{x}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} dx$$

   Using the change of variables $y = x - \mu$,

$$\mathbb{E}[X] = \int_{\mathbb{R}^d} \frac{y}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}y^T\Sigma^{-1}y\right\} dy + \mu\int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}y^T\Sigma^{-1}y\right\} dy$$

   So this is

$$\mathbb{E}[X] = \int_{\mathbb{R}^d} \frac{y}{\sqrt{(2\pi)^d|\Sigma|}} \exp\{-\frac{1}{2}y^T\Sigma^{-1}y\}\,dy + \mu$$

   Notice that

$$-\frac{1}{2}y^T\Sigma^{-1}y = -\frac{1}{2}(-y)^T\Sigma^{-1}(-y)$$

   This means that the integral:

$$\int_{\mathbb{R}^d} \frac{y}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}y^T\Sigma^{-1}y\right\} dy = 0$$

   So $\mathbb{E}[X] = \mu$.

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

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

Answer. Using the same change of variables, we have:

$$\mathrm{Var}(X) = \int_{\mathbb{R}^d} (y+\mu)(y+\mu)^T \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}y^T\Sigma^{-1}y\right\} dy$$

expanding this gives:

$$\mathrm{Var}(X) = \int_{\mathbb{R}^d} \frac{yy^T}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}y^T\Sigma^{-1}y\right\} dy + \mu\mu^T$$

3. Compute the moment generating function (MGF) of $X : M_X(\lambda) = \mathbb{E}[e^{\lambda^T X}]$, 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

   Answer. By definition, the MGF is given by

$$\int_{\mathbb{R}^d} e^{\lambda^T x} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} dx$$

   and from a change of variables:

$$\int_{\mathbb{R}^d} e^{\lambda^T(y+\mu)} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}y^T\Sigma^{-1}y\right\} dy$$

   We can take out the constant factor $e^{\lambda^T\mu}$ and move the rest into the larger exponential:

$$e^{\lambda^T\mu} \cdot \int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(y^T\Sigma^{-1/2}\Sigma^{-1/2}y + 2\lambda^T y)\right\} dy$$

   Now we complete the square in the exponent:

$$y^T\Sigma^{-1/2}\Sigma^{-1/2}y + y^T\lambda + \lambda^T y + \lambda^T\Sigma\lambda - \lambda^T\Sigma\lambda = (y^T\Sigma^{-1/2} + \lambda^T\Sigma^{1/2})(\Sigma^{-1/2}y + \Sigma^{1/2}\lambda) - \lambda^T\Sigma\lambda$$

   Now consider only $(y^T\Sigma^{-1/2} + \lambda^T\Sigma^{1/2})(\Sigma^{-1/2}y + \Sigma^{1/2}\lambda)$:

$$\begin{aligned}(y^T\Sigma^{-1/2} + \lambda^T\Sigma^{1/2})(\Sigma^{-1/2}y + \Sigma^{1/2}\lambda) &= (y^T + \lambda^T\Sigma)\Sigma^{-1/2}\Sigma^{-1/2}(y + \Sigma\lambda) \\ &= (y+\Sigma\lambda)^T\Sigma^{-1}(y+\Sigma\lambda)\end{aligned}$$

   Then the integral becomes:

$$\begin{aligned}\mathbb{E}[e^{\lambda^T X}] &= e^{\lambda^T\mu}\int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(y+\Sigma\lambda)^T\Sigma^{-1}(y+\Sigma\lambda) - \lambda^T\Sigma\lambda\right\} dy \\ &= \exp\left\{\lambda^T\mu + \frac{1}{2}\lambda^T\Sigma\lambda\right\}\int_{\mathbb{R}^d} \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left\{-\frac{1}{2}(y+\Sigma\lambda)^T\Sigma^{-1}(y+\Sigma\lambda)\right\} dy \\ &= \exp\left\{\lambda^T\mu + \frac{1}{2}\lambda^T\Sigma\lambda\right\}\end{aligned}$$

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).

   Answer. We have that the mean of the new distribution is

$$\mathbb{E}[AX+b] = A\mathbb{E}[X] + b = A\mu + b$$

and the variance is

$$\begin{aligned} Var(AX+b) = Var(AX) &= \mathbb{E}[AXX^TA^T] - A\mathbb{E}[X]\mathbb{E}[X^T]A^T \\ &= A\mathbb{E}[XX^T]A^T - A\mathbb{E}[X]\mathbb{E}[X^T]A^T \\ &= A(\mathbb{E}[XX^T] - \mathbb{E}[X]\mathbb{E}[X^T])A^T \\ &= A\Sigma A^T \end{aligned}$$

Then we see that this has the same distribution as a multivariate gaussian with mean $A\mu + b$ and variance $A\Sigma A^T$.

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.)

   Answer. By the previous question, we want an affine transformation of $X \to AX + b$ such that the new distribution has mean 0 and variance I_d. So:

   $$A\Sigma A^T = I_d$$

   We can write $\Sigma = \Sigma^{1/2}\Sigma^{1/2}$. Then $A = (\Sigma^{1/2})^{-1}$. Now we want a mean of 0. So

   $$A\mu + b = 0$$

   or

   $$b = -(\Sigma^{1/2})^{-1}\mu$$

   So the affine transformation is

   $$X \longrightarrow (\Sigma^{1/2})^{-1}X - (\Sigma^{1/2})^{-1}b = (\Sigma^{1/2})^{-1}(X - b)$$

Real Analysis

Exercise 1: Limit of a Sequence. A sequence $\{x_n\}$ is said to converge to a limit L if, for every measure of closeness $\varepsilon \in \mathbb{R}$, 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 $\forall \varepsilon > 0, \exists n_0 \in \mathbb{Z}^+$ such that $\forall n \geqslant n_0$:

$$|x_n - L| < \varepsilon$$

(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.**

Answer. The claim is that $L = 0$. So we need to show that $\forall \varepsilon > 0$, there is an N such that $\forall n > N$,

$$|x_n| < \varepsilon$$

We see that $x_n = \frac{1}{2^n}x_0$. Now fix ε. Then we want:

$$\frac{1}{2^n}|x_0| < \varepsilon$$

or

$$\frac{|x_0|}{\varepsilon} < 2^n$$

Pick $N > \log_2 \frac{|x_0|}{\varepsilon}$. Then we see that for all $n > N$, $|x_n| < \varepsilon$. So the sequence converges.

(b) Optional. Consider a sequence $\{x_n\}$ of non-zero real numbers and suppose that $L = \lim_{n\to\infty} n(1 - \frac{|x_{n+1}|}{|x_n|})$ exists. Prove that $\{|x_n|\}$ converges when $L > 1$ by evaluating $\lim_{n\to\infty} |x_n|$.

Exercise 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:

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

his 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 $a = 0$.

Answer. Here are the necessary derivatives:

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

Then

$$f(0) = 0$$
$$f'(0) = 1$$
$$f''(0) = -1$$

Then the first three terms are:

$$p(x) = 0 + x - \frac{1}{2}x^2$$

(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.

Exercise 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 $\chi = \{x \mid \|x - x^*\|^2 \leqslant D\}$ around x^* for some constant D, the Hessian matrix H of a function $f(x)$ is PSD and its maximum eigenvalue is 1. Prove that

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

for every $x \in \chi$.

Answer. The multivariate taylor expansion of degree 2 is:

$$T(x) = f(x^*) + (x - x^*)^T \nabla f(x^*) + \frac{1}{2}(x - x^*)^T H_f(x^*)(x - x^*)$$

then let $R_2(x) = \varepsilon$ where ε is the error of the first degree taylor expansion centered at x^*. Notice that

$$f(x) - f(x^*) = \varepsilon$$

The error bound is given by the next degree term

$$R_2(\mathcal{E}) = \frac{1}{2}(\mathcal{E} - x^*)^T H_f(x^*)(\mathcal{E} - x^*)$$

By the taylor remainder theorem, the maximum error is given by the max over the condition $\|\mathcal{E} - x^*\| \leqslant D$, as we know that $\|x - x^*\| \leqslant D$. Recall that for $\|x - x^*\| \leqslant 1$, the maximum of $x^T Ax$ for a PSD matrix is given by the eigenvector corresponding the max eigenvalue. Since the max eigenvalue is 1, we know that

$$\frac{1}{2}(\mathcal{E} - x^*)^T H_f(x^*)(\mathcal{E} - x) \leqslant \frac{1}{2}(\mathcal{E} - x^*)^T(\mathcal{E} - x) \leqslant \frac{D}{2}$$

So

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

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. You can plot figures using Matplotlib's `plt.plot` method.

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 element-wise 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:

$$\hat{Y} = Xw$$

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.

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

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

Derive $\nabla_w \mathcal{J}_{SE}(w)$ and set it equal to 0 to solve for w^*. (Note that this procedure for finding an optimum relies on the convexity of $\mathcal{J}_{SE}$. 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 $\mathcal{J}_{SE}$ from part (b)j divided by the number of data points:

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

Also visually inspect the model's quality by plotting a line plot of predicted $\hat{y}$ for uniformly space $x \in [0, 10]$. 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:

$$\hat{Y} = \Psi w = \begin{bmatrix} \varphi(x_1)^T \\ \varphi(x_2)^T \\ \vdots \\ \varphi(x_n)^T \end{bmatrix} w,$$

where $\varphi(x_i), w \in \mathbb{R}^m$. 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**:

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

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

(e) Now consider the quadratic features:

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

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 Ψ 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:

$$\varphi_1(x_i) = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \quad \varphi_2(x_i) = \begin{bmatrix} x_i^2 \\ x_i \\ 1 \end{bmatrix} \quad \varphi_3(x_i) = \begin{bmatrix} x_i^3 \\ x_i^2 \\ x_i \\ 1 \end{bmatrix} \quad \varphi_4(x_i) = \begin{bmatrix} x_i^4 \\ x_i^3 \\ x_i^2 \\ x_i \\ 1 \end{bmatrix} \quad \varphi_5(x_i) = \begin{bmatrix} x_i^5 \\ x_i^4 \\ x_i^3 \\ x_i^2 \\ x_i \\ 1 \end{bmatrix}$$

This step will product 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 data points for model fitting while still having held out data points 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?

Answer. For dataset 1, the training mean squared error suggested that a degree 5 feature was the best fit, but in contrast, the validation mean squared error suggested that a degree 1 feature was the best fit.

For dataset 2, both the training and validation error showed that a degree 2 feature was the best fit as it gave the lowest average error.

Here is my code:

```python
import matplotlib.pyplot as plt
import numpy as np

d1 = np.load("./data/dataset_1.npy")
d2 = np.load("./data/dataset_2.npy")


def split_variable(data):
    return data[:, 0], data[:, 1]


def get_distribution(data):
    mean = sum(data).astype(np.float16) / len(data)
    variance = sum(np.square(data - mean)) / (len(data) - 1)
    return mean, variance


def get_covariance(x, y, mu_x, mu_y):
    cov = (x - mu_x) * (y - mu_y)
    return sum(cov) / len(cov)


def get_correlation(cov, var_x, var_y):
    return cov / np.sqrt(var_x * var_y)


d1_X, d1_Y = split_variable(d1)
d2_X, d2_Y = split_variable(d2)

mu1_X, var1_X = get_distribution(d1_X)
mu1_Y, var1_Y = get_distribution(d1_Y)

mu2_X, var2_X = get_distribution(d2_X)
mu2_Y, var2_Y = get_distribution(d2_Y)

cov1_XY = get_covariance(d1_X, d1_Y, mu1_X, mu1_Y)
cov2_XY = get_covariance(d2_X, d2_Y, mu2_X, mu2_Y)
```

```
corr1_XY = get_correlation(cov1_XY, var1_X, var1_Y)
corr2_XY = get_correlation(cov2_XY, var2_X, var2_Y)

# plt.scatter(d1_X, d1_Y, color="red")
# plt.scatter(d2_X, d2_Y, color="blue")


def least_square(x, y, feature='nonlinear'):
    if feature == 'linear':
        return np.dot(x, y) / np.dot(x, x)
    elif feature == 'nonlinear':
        xTx = x.T @ x
        a = np.linalg.inv(xTx)
        lhs = (x.T @ y)
        return a @ lhs


def get_mse(x, y, w):
    estimate = np.dot(x, w)
    error = estimate - y
    mean_squared_error = np.dot(error, error) / len(estimate)
    return mean_squared_error


def plot_fit(data, fit, shape):
    # data is an array of data
    # fit is an array of fits that fit each data piece in data
    rows, cols = shape
    f, axes = plt.subplots(rows, cols)
    if axes.shape[0] == 1:
        for i in range(len(data)):
            curr_data = data[i]
            x, y = curr_data[:, 0], curr_data[:, 1]
            axes[i].scatter(x, y, color="red")

            curr_fit = fit[i]
            x, y = curr_fit[:, 0], curr_fit[:, 1]
            axes[i].scatter(x, y, color="green")
    else:
        for i in range(len(data)):
            curr_data = data[i]
            x, y = curr_data[:, 0], curr_data[:, 1]
            axes[i // cols][i % cols].scatter(x, y, color="red")

            curr_fit = fit[i]
            x, y = curr_fit[:, 0], curr_fit[:, 1]
            axes[i // cols][i % cols].scatter(x, y, color="green")

    plt.show()


def get_feature(x, degree):
    if degree <= 0:
        return x
    else:
        res = [np.ones_like(x)]
        for i in range(degree):
            res.insert(0, x ** (i + 1))
        return np.array(res).T


def fit_data(data, degree):
    x, y = data[:, 0], data[:, 1]
    x = get_feature(x, degree)
    w = least_square(x, y, feature=('linear' if degree == 0 else 'nonlinear'))

    pred_x = np.linspace(0, 10, len(data))
    pred_y = np.dot(get_feature(pred_x, degree), w)
    fit = np.array([pred_x, pred_y]).T

    return data, fit, w


def kfold_cross_validate(data, k, max_degree=5):
    np.random.shuffle(data)
    subset_sz = len(data) // k
    training_sets = [np.concatenate((data[:i*subset_sz], data[(i + 1)*subset_sz:])) for
    validation_sets = [data[i*subset_sz:(i+1)*subset_sz] for i in range(k)]

    training_error = np.zeros(max_degree)
    validation_error = np.zeros(max_degree)

    for i in range(len(training_sets)):
        t_set = training_sets[i]
        v_set = validation_sets[i]
        for j in range(1, max_degree + 1):
            data, fit, w = fit_data(t_set, j)

            training_error[j - 1] += get_mse(get_feature(t_set[:, 0], j), t_set[:, 1],
            validation_error[j - 1] += get_mse(get_feature(v_set[:, 0], j), v_set[:, 1]

    training_error /= k
    validation_error /= k

    return training_error, validation_error


data = []
fits = []

################################################################################
# Dataset 1 linear feature                                                     #
################################################################################

d, fit, w = fit_data(d1, 0)
data.append(d), fits.append(fit)

################################################################################
# Dataset 1 affine feature                                                     #
################################################################################

d, fit, w = fit_data(d1, 1)
```

```
data.append(d), fits.append(fit)

################################################################################
# Dataset 1 quadratic features                                                 #
################################################################################

d, fit, w = fit_data(d1, 2)
data.append(d), fits.append(fit)

################################################################################
# Dataset 2 linear features                                                    #
################################################################################

d, fit, w = fit_data(d2, 0)
data.append(d), fits.append(fit)

################################################################################
# Dataset 2 affine features                                                    #
################################################################################

d, fit, w = fit_data(d2, 1)
data.append(d), fits.append(fit)

################################################################################
# Dataset 2 quadratic features                                                 #
################################################################################

d, fit, w = fit_data(d2, 2)
data.append(d), fits.append(fit)


plot_fit(data, fits, (2, 3))

train_error, val_error = kfold_cross_validate(d1, k=5, max_degree=5)
plt.scatter(np.arange(1, 6, 1), train_error, color="red")
plt.scatter(np.arange(1, 6, 1), val_error, color="blue")

plt.yscale("log")
plt.show()
print(train_error, val_error)



print("done")
```