I'm a student learning deep learning. You need to solve some problems and give the detailed interpretations. The first problem is:

Vector Calculus Review

Let x, c ∈ R

n

and A ∈ R

n×n

. For the following parts, before taking any derivatives, identify what the derivative looks like (is it a scalar, vector, or matrix?) and how we calculate each term in the derivative.

Then carefully solve for an arbitrary entry of the derivative, then stack/arrange all of them to get the final

result. Note that the convention we will use going forward is that vector derivatives of a scalar (with respect

to a column vector) are expressed as a row vector, i.e. ∂f

∂x = [∂f

∂x1

,

∂f

∂x2

, ...,

∂f

∂xn

] since a row acting on a

column gives a scalar. You may have seen alternative conventions before, but the important thing is that you

need to understand the types of objects and how they map to the shapes of the multidimensional arrays we

use to represent those types.

(a) Show ∂

∂x

(x

T c) = c

T

(b) Show ∂

∂x

||x||2

2 = 2x

T

(c) Show ∂

∂x

(Ax) = A

(d) Show ∂

∂x

(x

T Ax) = x

T

(A + AT

)

(e) Under what condition is the previous derivative equal to 2x

T A?

Let's dive into this vector calculus review problem with enthusiasm, tackling each part systematically. We're tasked with computing derivatives of various functions with respect to a vector $\mathbf{x} \in \mathbb{R}^n$, where $\mathbf{c} \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$. For each part, we'll first identify the type of derivative (scalar, vector, or matrix) and describe how to compute it, then derive an arbitrary entry of the derivative, and finally assemble the full result. The convention specifies that the derivative of a scalar with respect to a column vector $\mathbf{x}$ is a row vector, i.e., $\frac{\partial f}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right]$. Let's get started!

Part (a): Show $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T \mathbf{c}) = \mathbf{c}^T$

Step 1: Identify the derivative type and computation method

The function $f(\mathbf{x}) = \mathbf{x}^T \mathbf{c}$ is a scalar (since $\mathbf{x}^T \mathbf{c} = \sum_{i=1}^n x_i c_i$, a dot product). The derivative $\frac{\partial f}{\partial \mathbf{x}}$ is taken with respect to the column vector $\mathbf{x}$, so by the given convention, it's a row vector in $\mathbb{R}^{1 \times n}$. Each component of the derivative is $\frac{\partial f}{\partial x_i}$, where $f = \sum_{j=1}^n x_j c_j$. To compute the i-th entry, differentiate f with respect to x_i, treating all other x_j (for $j \neq i$) as constants.

Step 2: Compute an arbitrary entry

Write the function explicitly:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{c} = x_1 c_1 + x_2 c_2 + \cdots + x_n c_n.$$

The partial derivative with respect to x_i:

$$\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\sum_{j=1}^{n} x_j c_j \right).$$

Since only the term $x_i c_i$ involves x_i, and c_i is a constant, we get:

$$\frac{\partial f}{\partial x_i} = c_i.$$

Step 3: Assemble the derivative

The derivative is a row vector:

$$\frac{\partial f}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right] = [c_1, c_2, \ldots, c_n] = \mathbf{c}^T.$$

Thus, $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T \mathbf{c}) = \mathbf{c}^T$, as required.

Part (b): Show $\frac{\partial}{\partial \mathbf{x}} \|\mathbf{x}\|_2^2 = 2\mathbf{x}^T$

Step 1: Identify the derivative type and computation method

The function is $f(\mathbf{x}) = \|\mathbf{x}\|_2^2 = \mathbf{x}^T \mathbf{x} = \sum_{i=1}^{n} x_i^2$, a scalar. The derivative $\frac{\partial f}{\partial \mathbf{x}}$ is a row vector in $\mathbb{R}^{1 \times n}$. To find the i-th component, compute $\frac{\partial f}{\partial x_i}$, differentiating $f = x_1^2 + x_2^2 + \cdots + x_n^2$ with respect to x_i.

Step 2: Compute an arbitrary entry

The function is:

$$f(\mathbf{x}) = \sum_{j=1}^{n} x_j^2.$$

Differentiate with respect to x_i:

$$\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i} \left(x_1^2 + x_2^2 + \cdots + x_i^2 + \cdots + x_n^2 \right).$$

Only the term x_i^2 depends on x_i, so:

$$\frac{\partial f}{\partial x_i} = \frac{\partial}{\partial x_i}(x_i^2) = 2x_i.$$

Step 3: Assemble the derivative

The derivative is:

$$\frac{\partial f}{\partial \mathbf{x}} = [2x_1, 2x_2, \ldots, 2x_n] = 2[x_1, x_2, \ldots, x_n] = 2\mathbf{x}^T.$$

Thus, $\frac{\partial}{\partial \mathbf{x}} \|\mathbf{x}\|_2^2 = 2\mathbf{x}^T$.

Part (c): Show $\frac{\partial}{\partial \mathbf{x}}(A\mathbf{x}) = A$

Step 1: Identify the derivative type and computation method

Here, $f(\mathbf{x}) = A\mathbf{x}$, where $A \in \mathbb{R}^{n \times n}$ and $\mathbf{x} \in \mathbb{R}^n$, so $A\mathbf{x} \in \mathbb{R}^n$, a column vector. The derivative $\frac{\partial (A\mathbf{x})}{\partial \mathbf{x}}$ is the Jacobian, a matrix in $\mathbb{R}^{n \times n}$, where the (i, j)-th entry is $\frac{\partial (A\mathbf{x})_i}{\partial x_j}$. The function's i-th component is:

$$(A\mathbf{x})_i = \sum_{k=1}^{n} a_{ik} x_k.$$

We compute the partial derivative of the i-th component with respect to x_j.

Step 2: Compute an arbitrary entry

For the i-th component of $A\mathbf{x}$:

$$(A\mathbf{x})_i = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n.$$

Differentiate with respect to x_j:

$$\frac{\partial (A\mathbf{x})_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\sum_{k=1}^{n} a_{ik} x_k\right).$$

Only the term where $k = j$ involves x_j, and a_{ij} is constant, so:

$$\frac{\partial (A\mathbf{x})_i}{\partial x_j} = a_{ij}.$$

Step 3: Assemble the derivative

The Jacobian matrix is:

$$\frac{\partial (A\mathbf{x})}{\partial \mathbf{x}} = \left[\frac{\partial (A\mathbf{x})_i}{\partial x_j}\right]_{i,j} = [a_{ij}] = A.$$

Thus, $\frac{\partial}{\partial \mathbf{x}}(A\mathbf{x}) = A$.

Part (d): Show $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T A\mathbf{x}) = \mathbf{x}^T(A + A^T)$

Step 1: Identify the derivative type and computation method

The function $f(\mathbf{x}) = \mathbf{x}^T A\mathbf{x}$ is a scalar (a quadratic form). The derivative $\frac{\partial f}{\partial \mathbf{x}}$ is a row vector in $\mathbb{R}^{1\times n}$. Write:

$$f(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i a_{ij} x_j.$$

The k-th component of the derivative is $\frac{\partial f}{\partial x_k}$. We'll differentiate the quadratic form, considering terms involving x_k.

Step 2: Compute an arbitrary entry

Express the function:

$$f(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i a_{ij} x_j.$$

Differentiate with respect to x_k:

$$\frac{\partial f}{\partial x_k} = \frac{\partial}{\partial x_k}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} x_i a_{ij} x_j\right).$$

Consider terms where x_k appears, i.e., when $i = k$ or $j = k$:

- When $i = k$: $\sum_{j=1}^{n} x_k a_{kj} x_j$. Differentiate:

  $$\frac{\partial}{\partial x_k}\left(\sum_{j=1}^{n} x_k a_{kj} x_j\right) = \sum_{j=1}^{n} a_{kj} x_j,$$

  since x_j is constant with respect to x_k.

- When $j = k$: $\sum_{i=1}^{n} x_i a_{ik} x_k$. Rewrite as $\sum_{i=1}^{n} x_i a_{ik} x_k$, and differentiate:

  $$\frac{\partial}{\partial x_k}\left(\sum_{i=1}^{n} x_i a_{ik} x_k\right) = \sum_{i=1}^{n} x_i a_{ik},$$

  since x_i and a_{ik} are constant.

- The term where $i = k$ and $j = k$, i.e., $x_k a_{kk} x_k$, is counted in both sums, so we must avoid double-counting. Let's compute the total:

  $$\frac{\partial f}{\partial x_k} = \sum_{j=1}^{n} a_{kj} x_j + \sum_{i=1}^{n} x_i a_{ik}.$$

Rewrite the second sum by changing the index $i \to j$:

$$\sum_{i=1}^{n} x_i a_{ik} = \sum_{j=1}^{n} x_j a_{jk}.$$

Thus:

$$\frac{\partial f}{\partial x_k} = \sum_{j=1}^{n} a_{kj} x_j + \sum_{j=1}^{n} x_j a_{jk} = \sum_{j=1}^{n} (a_{kj} + a_{jk}) x_j.$$

In matrix form, this is the k-th component of $\mathbf{x}^T(A + A^T)$.

Step 3: Assemble the derivative

The derivative is:

$$\frac{\partial f}{\partial \mathbf{x}} = \left[\sum_{j=1}^{n} (a_{1j} + a_{j1}) x_j, \ldots, \sum_{j=1}^{n} (a_{nj} + a_{jn}) x_j\right].$$

This matches $\mathbf{x}^T(A + A^T)$, since:

$$(A + A^T)_{jk} = a_{jk} + a_{kj},$$

and the k-th component is:

$$[\mathbf{x}^T(A + A^T)]_k = \sum_{j=1}^{n} x_j (a_{jk} + a_{kj}).$$

Thus, $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T A \mathbf{x}) = \mathbf{x}^T(A + A^T)$.

Part (e): Under what condition is the previous derivative equal to $2\mathbf{x}^T A$?

We need:

$$\mathbf{x}^T(A + A^T) = 2\mathbf{x}^T A.$$

Equate the matrices acting on $\mathbf{x}^T$:

$$A + A^T = 2A.$$

Subtract A from both sides:

$$A^T = A.$$

This means A must be symmetric, i.e., $A = A^T$. If A is symmetric, then:

$$A + A^T = A + A = 2A,$$

so:

$$\mathbf{x}^T(A + A^T) = \mathbf{x}^T(2A) = 2\mathbf{x}^T A.$$

Thus, the condition is that A is a symmetric matrix.

Summary of Results

- (a) $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T\mathbf{c}) = \mathbf{c}^T$
- (b) $\frac{\partial}{\partial \mathbf{x}}\|\mathbf{x}\|_2^2 = 2\mathbf{x}^T$
- (c) $\frac{\partial}{\partial \mathbf{x}}(A\mathbf{x}) = A$
- (d) $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T A\mathbf{x}) = \mathbf{x}^T(A + A^T)$
- (e) The derivative in (d) equals $2\mathbf{x}^T A$ when A is symmetric ($A = A^T$).

These results are fundamental in deep learning for optimizing loss functions involving linear and quadratic forms!

You've done a good job! Now let's look at the second question:

3. Least Squares and the Min-norm problem from the Perspective of SVD

Consider the equation Xw = y, where X ∈ R

m×n

is a non-square data matrix, w is a weight vector, and y

is vector of labels corresponding to the datapoints in each row of X.

Let’s say that X = UΣV

T

is the (full) SVD of X. Here, U and V are orthonormal square matrices, and

Σ

is an m × n matrix with non-zero singular values (σi) on the "diagonal".

For this problem, we define Σ

†

an n × m matrix with the reciprocals of the singular values (1

σi

) along the

"diagonal".

(a) First, consider the case where m > n, i.e. our data matrix X has more rows than columns (tall matrix)

and the system is overdetermined. How do we find the weights w that minimizes the error between

Xw and y? In other words, we want to solve minw ∥ Xw − y ∥
2
.
(b) Plug in the SVD X = UΣV
T and simplify. Be careful with dimensions!
(c) You'll notice that the least-squares solution is in the form w⋆ = Ay.
What happens if we leftmultiply X by our matrix A? This is why the matrix
A of the least-squares solution is called the
left-inverse.
(d) Now, let's consider the case where m < n, i.e. the data matrix X has
more columns than rows and
the system is underdetermined. There exist infinitely many solutions for
w, but we seek the minimumnorm solution, ie. we want to solve min ∥ w ∥
2
s.t.Xw = y. What is the minimum norm solution?
(e) Plug in the SVD X = UΣV
T and simplify. Be careful with dimensions!
(f) You'll notice that the min-norm solution is in the form w⋆ = By. What
happens if we right-multiply
X by our matrix B? This is why the matrix B of the min-norm solution is
called the right-inverse.

Let's tackle this problem on least squares and the minimum-norm solution using the Singular Value Decomposition (SVD) of the data matrix $X \in \mathbb{R}^{m\times n}$. The matrix X has the SVD $X = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthonormal (i.e., $U^TU = I_m, V^TV = I_n$), and $\Sigma \in \mathbb{R}^{m\times n}$ has non-zero singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$ (where $r = \min(m, n)$) along the "diagonal" and zeros elsewhere. The matrix $\Sigma^\dagger \in \mathbb{R}^{n\times m}$ is the pseudoinverse of Σ, with $\frac{1}{\sigma_i}$ on the "diagonal" corresponding to non-zero singular values. We'll solve each part systematically, ensuring clarity on dimensions and interpretations, as this is crucial for understanding the role of SVD in linear regression problems central to deep learning.

Part (a): Overdetermined case ($m > n$), minimize $\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|_2^2$

Objective and Approach

For $X \in \mathbb{R}^{m\times n}$ with $m > n$ (a tall matrix), the system $X\mathbf{w} = \mathbf{y}$ is overdetermined, typically having no exact solution. We seek the weight vector $\mathbf{w} \in \mathbb{R}^n$ that minimizes the squared Euclidean norm of the residual:

$$\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|_2^2 = \min_{\mathbf{w}} (X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y}).$$

This is the least squares problem. To find the optimal $\mathbf{w}$, we take the derivative of the objective function with respect to $\mathbf{w}$ and set it to zero. The objective is:

$$f(\mathbf{w}) = \|X\mathbf{w} - \mathbf{y}\|_2^2 = (X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y}) = \mathbf{w}^T X^T X \mathbf{w} - 2\mathbf{w}^T X^T \mathbf{y} + \mathbf{y}^T \mathbf{y}.$$

Using vector calculus (from the previous problem):

- Derivative of $\mathbf{w}^T X^T X \mathbf{w}$: Since $X^T X$ is symmetric, $\frac{\partial}{\partial \mathbf{w}} (\mathbf{w}^T X^T X \mathbf{w}) = \mathbf{w}^T (X^T X + (X^T X)^T) = 2\mathbf{w}^T X^T X$.
- Derivative of $-2\mathbf{w}^T X^T \mathbf{y}$: $\frac{\partial}{\partial \mathbf{w}} (-2\mathbf{w}^T X^T \mathbf{y}) = -2(X^T \mathbf{y})^T = -2\mathbf{y}^T X$.
- Derivative of $\mathbf{y}^T \mathbf{y}$: Constant, so zero.

The gradient is:

$$\frac{\partial f}{\partial \mathbf{w}} = 2\mathbf{w}^T X^T X - 2\mathbf{y}^T X = 0.$$

Transposing (since it's a row vector):

$$2X^T X \mathbf{w} - 2X^T \mathbf{y} = 0 \implies X^T X \mathbf{w} = X^T \mathbf{y}.$$

These are the normal equations. Assuming X has full column rank ($\mathrm{rank}(X) = n$), $X^T X \in \mathbb{R}^{n \times n}$ is invertible, so:

$$\mathbf{w}^* = (X^T X)^{-1} X^T \mathbf{y}.$$

This $\mathbf{w}^*$ is the least squares solution. The matrix $(X^T X)^{-1} X^T \in \mathbb{R}^{n \times m}$ is the (Moore-Penrose) pseudoinverse of X, often denoted $X^\dagger$.

Part (b): Plug in SVD $X = U \Sigma V^T$ and simplify

SVD Structure

Given $X = U \Sigma V^T$, where:

- $U \in \mathbb{R}^{m \times m}, U^T U = I_m$,
- $V \in \mathbb{R}^{n \times n}, V^T V = I_n$,
- $\Sigma \in \mathbb{R}^{m \times n}$, with $m > n$, so $\Sigma = \begin{bmatrix} \mathrm{diag}(\sigma_1, \ldots, \sigma_n) \\ 0 \end{bmatrix}$ (top $n \times n$ diagonal, bottom $(m - n) \times n$ zeros),

- $\Sigma^\dagger \in \mathbb{R}^{n\times m}$, with $\Sigma^\dagger = \begin{bmatrix}\text{diag}(1/\sigma_1, \ldots, 1/\sigma_n) & 0\end{bmatrix}$.

Assume X has full rank ($\text{rank}(X) = n$), so all $\sigma_i > 0$. Substitute into the least squares solution:

$$\mathbf{w}^* = (X^T X)^{-1} X^T \mathbf{y}.$$

Compute:

- $X^T = (U\Sigma V^T)^T = V\Sigma^T U^T$,
- $X^T X = (V\Sigma^T U^T)(U\Sigma V^T) = V\Sigma^T \Sigma V^T$, since $U^T U = I_m$,
- $\Sigma^T \Sigma \in \mathbb{R}^{n\times n} = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2)$, since $\Sigma^T = \begin{bmatrix}\text{diag}(\sigma_1, \ldots, \sigma_n) & 0\end{bmatrix}$, and multiplying gives a diagonal matrix.

Thus:

$$X^T X = V\text{diag}(\sigma_1^2, \ldots, \sigma_n^2)V^T.$$

Since V is orthonormal, invert:

$$(X^T X)^{-1} = (V\text{diag}(\sigma_1^2, \ldots, \sigma_n^2)V^T)^{-1} = V\text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2)V^T.$$

Now compute:

$$X^T = V\Sigma^T U^T,$$

so:

$$(X^T X)^{-1} X^T = (V\text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2)V^T)(V\Sigma^T U^T) = V\text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2)\Sigma^T$$

Since $\text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2)\Sigma^T = \text{diag}(1/\sigma_1, \ldots, 1/\sigma_n) = \Sigma^\dagger$ (considering dimensions), we get:

$$(X^T X)^{-1} X^T = V\Sigma^\dagger U^T.$$

Thus:

$$\mathbf{w}^* = V\Sigma^\dagger U^T \mathbf{y}.$$

This is the least squares solution using SVD, where $X^\dagger = V\Sigma^\dagger U^T$ is the pseudoinverse.

Part (c): Left-multiply X by matrix A, where $\mathbf{w}^* = A\mathbf{y}$

From part (b), $A = X^\dagger = V\Sigma^\dagger U^T$. Compute:

$$AX = (V\Sigma^\dagger U^T)(U\Sigma V^T) = V\Sigma^\dagger \Sigma V^T.$$

Since $\Sigma^\dagger \Sigma = \begin{bmatrix} I_n & 0 \end{bmatrix} \begin{bmatrix} \Sigma_n \\ 0 \end{bmatrix} = I_n$, we have:

$$AX = VI_nV^T = VV^T = I_n.$$

Thus, $AX = I_n$, meaning $A = V\Sigma^\dagger U^T$ is a left-inverse of X, satisfying $AX = I_n$. This is why A is called the left-inverse: it inverts X from the left, projecting $\mathbf{y}$ onto the column space of X to find the optimal $\mathbf{w}^*$.

Part (d): Underdetermined case ($m < n$), minimize $\min \|\mathbf{w}\|_2^2$ subject to $X\mathbf{w} = \mathbf{y}$

For $X \in \mathbb{R}^{m \times n}$ with $m < n$ (a wide matrix), the system $X\mathbf{w} = \mathbf{y}$ is underdetermined, typically having infinitely many solutions. We seek the minimum-norm solution:

$$\min \|\mathbf{w}\|_2^2 \quad \text{subject to} \quad X\mathbf{w} = \mathbf{y}.$$

Use the method of Lagrange multipliers. Define the Lagrangian:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \boldsymbol{\lambda}^T(X\mathbf{w} - \mathbf{y}),$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the Lagrange multiplier. Take derivatives:

- With respect to $\mathbf{w}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w}^T + \boldsymbol{\lambda}^T X = 0 \implies \mathbf{w} = -X^T\boldsymbol{\lambda}$.
- With respect to $\boldsymbol{\lambda}$: $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = X\mathbf{w} - \mathbf{y} = 0 \implies X\mathbf{w} = \mathbf{y}$.

Substitute $\mathbf{w} = -X^T\boldsymbol{\lambda}$ into the constraint:

$$X(-X^T\boldsymbol{\lambda}) = \mathbf{y} \implies -XX^T\boldsymbol{\lambda} = \mathbf{y} \implies \boldsymbol{\lambda} = -(XX^T)^{-1}\mathbf{y},$$

assuming $XX^T \in \mathbb{R}^{m \times m}$ is invertible ($\operatorname{rank}(X) = m$). Then:

$$\mathbf{w} = X^T(XX^T)^{-1}\mathbf{y}.$$

This is the minimum-norm solution, as it selects the $\mathbf{w}$ in the row space of X with the smallest $\|\mathbf{w}\|_2$.

Part (e): Plug in SVD $X = U\Sigma V^T$ and simplify

For $m < n$:

- $\Sigma \in \mathbb{R}^{m \times n} = \begin{bmatrix} \operatorname{diag}(\sigma_1, \ldots, \sigma_m) & 0 \end{bmatrix}$,

- $\Sigma^\dagger \in \mathbb{R}^{n\times m} = \begin{bmatrix} \text{diag}(1/\sigma_1, \ldots, 1/\sigma_m) \\ 0 \end{bmatrix}$.

Compute:

- $X^T = (U\Sigma V^T)^T = V\Sigma^T U^T$,
- $XX^T = (U\Sigma V^T)(V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$,
- $\Sigma\Sigma^T = \text{diag}(\sigma_1^2, \ldots, \sigma_m^2) \in \mathbb{R}^{m\times m}$.

Thus:

$$XX^T = U\text{diag}(\sigma_1^2, \ldots, \sigma_m^2)U^T,$$

and:

$$(XX^T)^{-1} = U\text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_m^2)U^T.$$

Then:

$$\mathbf{w} = X^T(XX^T)^{-1}\mathbf{y} = (V\Sigma^T U^T)(U\text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_m^2)U^T)\mathbf{y} = V\Sigma^T\text{diag}(1/\sigma_1^2, ..$$

Since $\Sigma^T\text{diag}(1/\sigma_1^2, \ldots, 1/\sigma_m^2) = \text{diag}(1/\sigma_1, \ldots, 1/\sigma_m) = \Sigma^\dagger$, we get:

$$\mathbf{w}^* = V\Sigma^\dagger U^T\mathbf{y}.$$

This matches the pseudoinverse form, $\mathbf{w}^* = X^\dagger\mathbf{y}$.

Part (f): Right-multiply X by matrix B, where $\mathbf{w}^* = B\mathbf{y}$

Here, $B = X^\dagger = V\Sigma^\dagger U^T$. Compute:

$$XB = (U\Sigma V^T)(V\Sigma^\dagger U^T) = U\Sigma\Sigma^\dagger U^T.$$

Since $\Sigma\Sigma^\dagger = \begin{bmatrix} \text{diag}(\sigma_1, \ldots, \sigma_m) & 0 \end{bmatrix} \begin{bmatrix} \text{diag}(1/\sigma_1, \ldots, 1/\sigma_m) \\ 0 \end{bmatrix} = I_m$, we have:

$$XB = UI_mU^T = UU^T = I_m.$$

Thus, $XB = I_m$, meaning $B = V\Sigma^\dagger U^T$ is a right-inverse of X, satisfying $XB = I_m$. This ensures $X\mathbf{w}^* = X(X^\dagger\mathbf{y}) = \mathbf{y}$, and $\mathbf{w}^*$ has the minimum norm among all solutions.

Summary of Results

- **(a)** Least squares solution ($m > n$): $\mathbf{w}^* = (X^TX)^{-1}X^T\mathbf{y}$.

- **(b)** Using SVD: $\mathbf{w}^* = V\Sigma^\dagger U^T \mathbf{y}$.
- **(c)** $A = V\Sigma^\dagger U^T$ is a left-inverse, since $AX = I_n$.
- **(d)** Minimum-norm solution ($m < n$): $\mathbf{w}^* = X^T(XX^T)^{-1}\mathbf{y}$.
- **(e)** Using SVD: $\mathbf{w}^* = V\Sigma^\dagger U^T \mathbf{y}$.
- **(f)** $B = V\Sigma^\dagger U^T$ is a right-inverse, since $XB = I_m$.

These results highlight the power of SVD in solving linear systems, providing stable and interpretable solutions for both overdetermined and underdetermined cases, which are critical in deep learning for tasks like linear regression and optimization.

That's correct！ Now let's look at the next problem
4. The 5 Interpretations of Ridge Regression
(a) Perspective 1: Optimization Problem. Ridge regression can be
understood as the unconstrained optimization problem
argmin
w
∥ y − Xw ∥
2
2 + λ ∥ w ∥
2
2
, (1)
where X ∈ R
n×d
is a data matrix, and y ∈ R
n
is the target vector of measurement values. What’s new
compared to the simple OLS problem is the addition of the λ ∥ w ∥
2
term, which can be interpreted as a
"penalty" on the weights being too big.
Use vector calculus to expand the objective and solve this optimization
problem for w.
(b) Perspective 2: "Hack" of shifting the Singular Values. In the previous
part, you should have found the
optimal w is given by
w = (XT X + λI)

−1XT y

(If you didn’t get this, you should check your work for the previous part)

Let X = UΣV

T be the (full) SVD of the X. Recall that U and V are square orthonormal (normpreserving) matrices, and Σ is a n × d matrix with singular values σi along the "diagonal". Plug

this into the Ridge Regression solution and simplify. What happens to the singular values of

(XT X + λI)

−1XT when σi << λ? What about when σi >> λ?

(c) Perspective 3: Maximum A Posteriori (MAP) estimation. Ridge Regression can be viewed as finding

the MAP estimate when we apply a prior on the (now viewed as random parameters) W. In particular,

we can think of the prior for W as being N (0, I) and view the random Y as being generated using

Y = x

TW +

√

λN where the noise N is distributed iid (across training samples) as N (0, 1). At the

vector level, we have Y = XW +

√

λN. Note that the X matrix whose rows are the n different

training points are not random.

Show that (1) is the MAP estimate for W given an observation Y = y.

(d) Perspective 4: Fake Data. Another way to interpret “ridge regression” is as the ordinary least squares

for an augmented data set — i.e. adding a bunch of fake data points to our data. Consider the following

augmented measurement vector ˆy and data matrix Xˆ :

ˆy =

"

y

0d

#

Xˆ =

"
X
√
λId
#
,
where 0d is the zero vector in R
d
and Id ∈ R
d×d
is the identity matrix. Show that the classical OLS
optimization problem argminw ∥ ˆy − Xwˆ ∥
2
2 has the same minimizer as (1).
(e) Perspective 5: Fake Features. For this last interpretation, let's instead
construct an augmented design
matrix in the following way:
Xˇ = [X
√
λIn]
i.e. we stack X with √
λIn horizontally. Now our problem is underdetermined: the new dimension
d + n is larger than the number of points n. Therefore, there are infinitely
many values η ∈ R
d+n
for
which Xˇη = y. We are interested in the min-norm solution, ie. the
solution to
argmin
η
∥ η ∥
2
2
s.t. Xˇη = y. (2)
Show that this is yet another form of ridge regression and that the first d
coordinates of η
*

form

the minimizer of (1).

(f) We know that the Moore-Penrose pseudo-inverse for an underdetermined system (wide matrix) is

given by A† = AT

(AAT

)

−1

, which corresponds to the min-norm solution for Aη = z. That is, the optimization problem

argmin ∥ η ∥

2

s.t.Aη = z

is solved by η = A†z. Let ˆ w be the minimizer of (1).

Use the pseudo-inverse to show that solving to the optimization problem in (2) yields

ˆ w = XT

(XXT + λI)

−1y

Then, show that this is equivalent to the standard formula for Ridge Regression

ˆ w = (XT X + λI)

−1XT y

Hint: It may be helpful to review Kernel Ridge Form.

(g) We know that the solution to ridge regression (1) is given by ˆ wr =

(X⊤X + λI)

−1X⊤y. What

happens when λ → ∞? It is for this reason that sometimes ridge regularization is referred to as

"shrinkage."

(h) What happens to the solution of ridge regression when you take the limit λ → 0? Consider both

the cases when X is wide (underdetermined system) and X is tall (overdetermined system).

Let's dive into the five interpretations of ridge regression, a cornerstone of regularized linear regression in machine learning. We'll solve each part systematically, using vector calculus, SVD, probabilistic interpretations, and matrix manipulations, while ensuring clarity on dimensions and providing detailed interpretations relevant to deep learning. The problem revolves around the ridge regression optimization:

$$\arg\min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2,$$

where $X \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{w} \in \mathbb{R}^d$, and $\lambda > 0$ is the regularization parameter. Let's proceed step-by-step through each part.

Part (a): Optimization Problem

We need to solve the ridge regression optimization problem:

$$\arg\min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2.$$

The objective function is:

$$f(\mathbf{w}) = (\mathbf{y} - X\mathbf{w})^T(\mathbf{y} - X\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}.$$

Expand the first term:

$$\|\mathbf{y} - X\mathbf{w}\|_2^2 = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T X\mathbf{w} + \mathbf{w}^T X^T X\mathbf{w}.$$

Thus:

$$f(\mathbf{w}) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T X\mathbf{w} + \mathbf{w}^T X^T X\mathbf{w} + \lambda\mathbf{w}^T\mathbf{w}.$$

Take the derivative with respect to $\mathbf{w}$ (using vector calculus rules, noting the convention that $\frac{\partial f}{\partial \mathbf{w}}$ is a row vector):

- Derivative of $\mathbf{y}^T\mathbf{y}$: Constant, so zero.
- Derivative of $-2\mathbf{y}^T X\mathbf{w}$: $\frac{\partial}{\partial \mathbf{w}}(-2\mathbf{w}^T X^T\mathbf{y}) = -2(X^T\mathbf{y})^T = -2\mathbf{y}^T X$.
- Derivative of $\mathbf{w}^T X^T X\mathbf{w}$: Since $X^T X$ is symmetric, $\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T X^T X\mathbf{w}) = \mathbf{w}^T(X^T X + (X^T X)^T) = 2\mathbf{w}^T X^T X$.
- Derivative of $\lambda\mathbf{w}^T\mathbf{w}$: $\frac{\partial}{\partial \mathbf{w}}(\lambda\mathbf{w}^T\mathbf{w}) = 2\lambda\mathbf{w}^T$.

The gradient is:

$$\frac{\partial f}{\partial \mathbf{w}} = -2\mathbf{y}^T X + 2\mathbf{w}^T X^T X + 2\lambda\mathbf{w}^T = 0.$$

Transposing and solving:

$$2X^T X\mathbf{w} + 2\lambda\mathbf{w} - 2X^T\mathbf{y} = 0 \implies (X^T X + \lambda I)\mathbf{w} = X^T\mathbf{y}.$$

Since $\lambda > 0$, $X^T X + \lambda I$ is positive definite (hence invertible), so:

$$\mathbf{w}^* = (X^T X + \lambda I)^{-1} X^T \mathbf{y}.$$

This is the ridge regression solution, where the $\lambda \|\mathbf{w}\|_2^2$ term penalizes large weights, promoting smaller, more stable solutions compared to ordinary least squares (OLS).

Part (b): Shifting the Singular Values

Given the solution $\mathbf{w}^* = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$ and $X = U \Sigma V^T$, where $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{d \times d}$, and $\Sigma \in \mathbb{R}^{n \times d}$ with singular values σ_i on the "diagonal," we plug in the SVD and analyze the effect on singular values.

Compute:

- $X^T X = (U \Sigma V^T)^T (U \Sigma V^T) = V \Sigma^T \Sigma V^T$.
- For $n \geq d$, $\Sigma^T \Sigma = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_d^2) \in \mathbb{R}^{d \times d}$.
- Thus, $X^T X + \lambda I = V(\Sigma^T \Sigma + \lambda I_d) V^T$, where $\Sigma^T \Sigma + \lambda I_d = \operatorname{diag}(\sigma_1^2 + \lambda, \ldots, \sigma_d^2 + \lambda)$.
- Invert: $(X^T X + \lambda I)^{-1} = V(\Sigma^T \Sigma + \lambda I_d)^{-1} V^T = V \operatorname{diag}\left(\frac{1}{\sigma_1^2 + \lambda}, \ldots, \frac{1}{\sigma_d^2 + \lambda}\right) V^T$.
- Then, $X^T = V \Sigma^T U^T$, so:

$$(X^T X + \lambda I)^{-1} X^T = V \operatorname{diag}\left(\frac{1}{\sigma_1^2 + \lambda}, \ldots, \frac{1}{\sigma_d^2 + \lambda}\right) V^T \cdot V \Sigma^T U^T = V \operatorname{diag}\left(\frac{\sigma_1}{\sigma_1^2 + \lambda}\right.$$

Thus:

$$\mathbf{w}^* = V \operatorname{diag}\left(\frac{\sigma_1}{\sigma_1^2 + \lambda}, \ldots, \frac{\sigma_d}{\sigma_d^2 + \lambda}\right) U^T \mathbf{y}.$$

Compare to OLS ($\mathbf{w}_{\text{OLS}} = V \Sigma^\dagger U^T \mathbf{y}$), where $\Sigma^\dagger$ has $\frac{1}{\sigma_i}$ for non-zero σ_i. The ridge solution modifies the singular values from $\frac{1}{\sigma_i}$ to $\frac{\sigma_i}{\sigma_i^2 + \lambda}$.

Effect on singular values:

- If $\sigma_i \ll \lambda$, then $\sigma_i^2 \ll \lambda$, so $\frac{\sigma_i}{\sigma_i^2 + \lambda} \approx \frac{\sigma_i}{\lambda}$, which is much smaller than $\frac{1}{\sigma_i}$. This dampens the contribution of directions with small singular values, reducing sensitivity to noise.
- If $\sigma_i \gg \lambda$, then $\sigma_i^2 \gg \lambda$, so $\frac{\sigma_i}{\sigma_i^2 + \lambda} \approx \frac{\sigma_i}{\sigma_i^2} = \frac{1}{\sigma_i}$, resembling the OLS solution for dominant directions.

This "shrinking" of singular values stabilizes the solution, especially for ill-conditioned matrices.

Part (c): Maximum A Posteriori (MAP) Estimation

Ridge regression is the MAP estimate for $\mathbf{W} \sim N(0, I_d)$, with $\mathbf{Y} = X\mathbf{W} + \sqrt{\lambda}\mathbf{N}$, where $\mathbf{N} \sim N(0, I_n)$. The likelihood is:

$$p(\mathbf{y}|\mathbf{w}) = N(\mathbf{y}; X\mathbf{w}, \lambda I_n) = \frac{1}{(2\pi\lambda)^{n/2}} \exp\left(-\frac{1}{2\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2\right).$$

The prior is:

$$p(\mathbf{w}) = N(\mathbf{w}; 0, I_d) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\mathbf{w}^T\mathbf{w}\right).$$

The MAP estimate maximizes the posterior $p(\mathbf{w}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{w})p(\mathbf{w})$. Take the log-posterior:

$$\log p(\mathbf{w}|\mathbf{y}) = \log p(\mathbf{y}|\mathbf{w}) + \log p(\mathbf{w}) + \text{const} = -\frac{1}{2\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2 - \frac{1}{2}\mathbf{w}^T\mathbf{w} + \text{const}.$$

Maximize by minimizing:

$$\frac{1}{2\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2 + \frac{1}{2}\mathbf{w}^T\mathbf{w}.$$

Multiply through by 2λ:

$$\|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\mathbf{w}^T\mathbf{w} = \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2.$$

This matches the ridge regression objective (1), so the MAP estimate is:

$$\mathbf{w}_{\text{MAP}} = (X^TX + \lambda I)^{-1}X^T\mathbf{y}.$$

Part (d): Fake Data

Consider the augmented system:

$$\hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix}, \quad \hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda}I_d \end{bmatrix}.$$

Solve the OLS problem:

$$\arg\min_{\mathbf{w}} \|\hat{\mathbf{y}} - \hat{X}\mathbf{w}\|_2^2.$$

The objective is:

$$\|\hat{\mathbf{y}} - \hat{X}\mathbf{w}\|_2^2 = \left\| \begin{bmatrix} \mathbf{y} - X\mathbf{w} \\ -\sqrt{\lambda} I_d \mathbf{w} \end{bmatrix} \right\|_2^2 = \|\mathbf{y} - X\mathbf{w}\|_2^2 + \| - \sqrt{\lambda}\mathbf{w}\|_2^2 = \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|$$

This is identical to the ridge regression objective (1). Thus, the minimizer is:

$$\mathbf{w}^* = (X^T X + \lambda I)^{-1} X^T \mathbf{y}.$$

The “fake data” $\sqrt{\lambda} I_d$ adds penalty terms, mimicking the regularization effect.

Part (e): Fake Features

Consider the augmented matrix $\check{X} = [X \quad \sqrt{\lambda} I_n] \in \mathbb{R}^{n \times (d+n)}$. Solve the minimum-norm problem:

$$\arg\min_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \check{X}\boldsymbol{\eta} = \mathbf{y}.$$

The minimum-norm solution is:

$$\boldsymbol{\eta}^* = \check{X}^\dagger \mathbf{y} = \check{X}^T (\check{X}\check{X}^T)^{-1} \mathbf{y}.$$

Compute:

- $\check{X}\check{X}^T = [X \quad \sqrt{\lambda} I_n][X^T \quad \sqrt{\lambda} I_n] = XX^T + \lambda I_n.$
- Thus, $(\check{X}\check{X}^T)^{-1} = (XX^T + \lambda I_n)^{-1}$.
- Then, $\check{X}^T = \begin{bmatrix} X^T \\ \sqrt{\lambda} I_n \end{bmatrix}$, so:

$$\boldsymbol{\eta}^* = \begin{bmatrix} X^T \\ \sqrt{\lambda} I_n \end{bmatrix} (XX^T + \lambda I_n)^{-1} \mathbf{y} = \begin{bmatrix} X^T (XX^T + \lambda I_n)^{-1} \mathbf{y} \\ \sqrt{\lambda} (XX^T + \lambda I_n)^{-1} \mathbf{y} \end{bmatrix}.$$

The first d coordinates are:

$$\mathbf{w}^* = X^T (XX^T + \lambda I_n)^{-1} \mathbf{y}.$$

We need to show this equals the ridge solution (part (f)).

Part (f): Equivalence to Standard Ridge Regression

Using the pseudoinverse, we have:

$$\mathbf{w}^* = X^T (XX^T + \lambda I_n)^{-1} \mathbf{y}.$$

Show this equals $\hat{\mathbf{w}} = (X^T X + \lambda I_d)^{-1} X^T \mathbf{y}$. This is the kernel ridge regression form. Use the Woodbury matrix identity or direct manipulation. Consider:

$$(XX^T + \lambda I_n)^{-1} X = X(X^T X + \lambda I_d)^{-1}.$$

This is a known identity (can be verified via SVD or matrix inversion lemma). Thus:

$$\mathbf{w}^* = X^T (XX^T + \lambda I_n)^{-1} \mathbf{y} = X^T X(X^T X + \lambda I_d)^{-1} \mathbf{y} = (X^T X + \lambda I_d - \lambda I_d)(X^T X$$

Thus, the first d coordinates of $\boldsymbol{\eta}^*$ match the ridge regression solution.

Part (g): Limit as $\lambda \to \infty$

As $\lambda \to \infty$, consider:

$$\hat{\mathbf{w}} = (X^T X + \lambda I_d)^{-1} X^T \mathbf{y}.$$

For large λ, λI_d dominates $X^T X$, so:

$$X^T X + \lambda I_d \approx \lambda I_d \implies (X^T X + \lambda I_d)^{-1} \approx \frac{1}{\lambda} I_d.$$

Thus:

$$\hat{\mathbf{w}} \approx \frac{1}{\lambda} I_d X^T \mathbf{y} \to \mathbf{0}_d.$$

This is why ridge regression is called “shrinkage”: as λ increases, the weights shrink toward zero, prioritizing small $\|\mathbf{w}\|_2$.

Part (h): Limit as $\lambda \to 0$

As $\lambda \to 0$:

$$\hat{\mathbf{w}} = (X^T X + \lambda I_d)^{-1} X^T \mathbf{y} \to (X^T X)^{-1} X^T \mathbf{y}.$$

- **Tall matrix ($n > d$)**: If X has full column rank, $X^T X$ is invertible, and the solution approaches the OLS solution, $\mathbf{w}_{\text{OLS}} = (X^T X)^{-1} X^T \mathbf{y}$.
- **Wide matrix ($n < d$)**: $X^T X$ is singular (rank $\leq n$), so the pseudoinverse is used: $\mathbf{w}^* = X^\dagger \mathbf{y} = X^T (XX^T)^{-1} \mathbf{y}$, the minimum-norm OLS solution.

Summary

- **(a)** $\mathbf{w}^* = (X^T X + \lambda I_d)^{-1} X^T \mathbf{y}$.

- **(b)** Singular values shift to $\frac{\sigma_i}{\sigma_i^2+\lambda}$, dampening small σ_i.
- **(c)** Ridge regression is the MAP estimate under Gaussian prior and noise.
- **(d)** Equivalent to OLS on augmented data $\hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I_d \end{bmatrix}$.
- **(e, f)** Min-norm solution on $\check{X} = [X \quad \sqrt{\lambda} I_n]$ yields ridge solution.
- **(g)** As $\lambda \to \infty$, $\mathbf{w} \to \mathbf{0}$ (shrinkage).
- **(h)** As $\lambda \to 0$, solution approaches OLS (full rank) or minimum-norm OLS (underdetermined).

These interpretations connect ridge regression to optimization, linear algebra, probability, and data augmentation, making it a versatile tool in deep learning for regularization and stability.

That's correct. Now let's look at the last problem
ReLU Elbow Update under SGD
In this question we will explore the behavior of the ReLU nonlinearity with Stochastic Gradient Descent
(SGD) updates. The hope is that this problem should help you build a more intuitive understanding for how
SGD works and how it iteratively adjusts the learned function.
We want to model a 1D function y = f(x) using a 1-hidden layer network with ReLU activations and no
biases in the linear output layer. Mathematically, our network is
ˆf(x) = W(2)Φ

W(1)x + b

where x, y ∈ R, b ∈ R
d
, W(1) ∈ R
d×1
, and W(2) ∈ R
1×d
. We define our loss function to be the squared
error,
ℓ

x, y,W(1)
, b,W(2)
=
1
2

ˆ f(x) − y

2
2
.
For the purposes of this problem, we define the gradient of a ReLU at 0 to be 0.
(a) Let’s start by examining the behavior of a single ReLU with a linear function of x as the input,
ϕ(x) = (
wx + b, wx + b > 0
0, else
.
Notice that the slope of ϕ(x) is w in the non-zero domain.
We define a loss function ℓ(x, y, ϕ) = 1
2
∥ ϕ(x) − y ∥
2
2
. Find the following:
(i) The location of the ‘elbow’ e of the function, where it transitions from 0

to something else.

(ii) The derivative of the loss w.r.t. $\phi(x)$, namely $d\ell$

$d\phi$

(iii) The partial derivative of the loss w.r.t. w, namely $\partial\ell$

∂w

(iv) The partial derivative of the loss w.r.t. b, namely $\partial\ell$

∂b

(b) Now suppose we have some training point (x, y) such that $\phi(x)-y = 1$.

In other words, the prediction

$\phi(x)$ is 1 unit above the target y — we are too high and are trying to pull the function downward.

Describe what happpens to the slope and elbow of $\phi(x)$ when we perform gradient descent in the

following cases:

(i) $\phi(x) = 0$.

(ii) $w > 0$, $x > 0$, and $\phi(x) > 0$. It is fine to check the behavior of the elbow numerically in this

case.

(iii) $w > 0$, $x < 0$, and $\phi(x) > 0$.

(iv) $w < 0$, $x > 0$, and $\phi(x) > 0$. It is fine to check the behavior of the elbow numerically in this

case.

Additionally, draw and label $\phi(x)$, the elbow, and the qualitative changes to the slope and elbow

after a gradient update to w and b. You should label the elbow location and a candidate (x, y)

pair. Remember that the update for some parameter vector p and loss ℓ under SGD is

p

$' = p - \lambda \nabla p(\ell)$, $\lambda > 0$.

(c) Now we return to the full network function $\hat{}f(x)$. Derive the location ei of the elbow of the i'th

elementwise ReLU activation.

(d) Derive the new elbow location e

′

i

of the i'th elementwise ReLU activation after one stochastic gradient update with learning rate λ.

Let's dive into this problem about the behavior of a 1D neural network with ReLU activations under Stochastic Gradient Descent (SGD). We're modeling a function $\hat{f}(x) = W^{(2)}\Phi(W^{(1)}x + \mathbf{b})$, where $x, y \in \mathbb{R}$, $\mathbf{b} \in \mathbb{R}^d$, $W^{(1)} \in \mathbb{R}^{d\times 1}$, $W^{(2)} \in \mathbb{R}^{1\times d}$, and Φ is the elementwise ReLU activation, defined as $\Phi(z) = \max(0, z)$, with the gradient at $z = 0$ defined as 0. The loss function is the squared error:

$$\ell(x, y, W^{(1)}, \mathbf{b}, W^{(2)}) = \frac{1}{2}(\hat{f}(x) - y)^2.$$

We'll solve each part systematically, focusing on the ReLU's "elbow" (where the function transitions from 0 to non-zero) and how SGD updates affect it, providing insights into how neural networks adjust during training.

Part (a): Single ReLU Analysis

Consider a single ReLU function:

$$\phi(x) = \begin{cases} wx + b & \text{if } wx + b > 0, \\ 0 & \text{otherwise,} \end{cases}$$

with loss function:

$$\ell(x, y, \phi) = \frac{1}{2}(\phi(x) - y)^2.$$

(i) Location of the elbow

The elbow is where $\phi(x)$ transitions from 0 to non-zero, i.e., where:

$$wx + b = 0 \implies x = -\frac{b}{w}, \quad (w \neq 0).$$

Thus, the elbow is at:

$$e = -\frac{b}{w}.$$

If $w = 0$, the function is constant ($\phi(x) = 0$ if $b \leq 0$, or $\phi(x) = b$ if $b > 0$), so no elbow exists unless $b = 0$, but we assume $w \neq 0$ for a well-defined elbow.

(ii) Derivative of the loss w.r.t. $\phi(x)$

Let $z = \phi(x)$. The loss is:

$$\ell = \frac{1}{2}(z - y)^2.$$

Differentiate with respect to z:

$$\frac{d\ell}{dz} = z - y = \phi(x) - y.$$

(iii) Partial derivative of the loss w.r.t. w

Use the chain rule:

$$\frac{\partial \ell}{\partial w} = \frac{d\ell}{dz} \cdot \frac{\partial z}{\partial w}, \quad z = \phi(x).$$

From (ii), $\frac{d\ell}{dz} = \phi(x) - y$. Compute $\frac{\partial z}{\partial w}$:

$$z = \phi(x) = \max(0, wx + b).$$

- If $wx + b > 0$, then $z = wx + b$, and $\frac{\partial z}{\partial w} = x$.
- If $wx + b \leq 0$, then $z = 0$, and $\frac{\partial z}{\partial w} = 0$ (since the gradient of ReLU at 0 is 0).

Using the indicator function $1(wx + b > 0)$, which is 1 if $wx + b > 0$ and 0 otherwise:

$$\frac{\partial z}{\partial w} = x \cdot 1(wx + b > 0).$$

Thus:

$$\frac{\partial \ell}{\partial w} = (\phi(x) - y) \cdot x \cdot 1(wx + b > 0).$$

(iv) Partial derivative of the loss w.r.t. b

Similarly:

$$\frac{\partial \ell}{\partial b} = \frac{d\ell}{dz} \cdot \frac{\partial z}{\partial b}.$$

Compute $\frac{\partial z}{\partial b}$:

- If $wx + b > 0$, then $z = wx + b$, and $\frac{\partial z}{\partial b} = 1$.
- If $wx + b \leq 0$, then $z = 0$, and $\frac{\partial z}{\partial b} = 0$.

Thus:

$$\frac{\partial z}{\partial b} = 1(wx + b > 0).$$

So:

$$\frac{\partial \ell}{\partial b} = (\phi(x) - y) \cdot 1(wx + b > 0).$$

Part (b): SGD Updates for a Single ReLU

Given a training point (x, y) such that $\phi(x) - y = 1$, the prediction is too high, and we perform SGD updates:

$$w' = w - \lambda \frac{\partial \ell}{\partial w}, \quad b' = b - \lambda \frac{\partial \ell}{\partial b}, \quad \lambda > 0.$$

We analyze the slope (w) and elbow ($e = -\frac{b}{w}$) in four cases, and describe qualitative changes with sketches.

(i) Case: $\phi(x) = 0$

If $\phi(x) = 0$, then $wx + b \leq 0$, and:

$$\frac{\partial \ell}{\partial w} = (0 - y) \cdot x \cdot 1(wx + b > 0) = 0, \quad \frac{\partial \ell}{\partial b} = (0 - y) \cdot 1(wx + b > 0) = 0.$$

Updates:

$$w' = w, \quad b' = b.$$

The slope and elbow remain unchanged. Since $\phi(x) = 0$, but $\phi(x) - y = 1 \implies y = -1$, the point $(x, -1)$ lies below the zero region of the ReLU. No update occurs because the ReLU is inactive.

Sketch: Plot $\phi(x) = 0$ for $x \leq -\frac{b}{w}$, and $\phi(x) = wx + b$ for $x > -\frac{b}{w}$. Mark the elbow at $x = e = -\frac{b}{w}$, and point $(x, -1)$. The function doesn't change post-update.

(ii) Case: $w > 0, x > 0, \phi(x) > 0$

Since $\phi(x) > 0, wx + b > 0$, and:

$$\frac{\partial \ell}{\partial w} = (\phi(x) - y) \cdot x \cdot 1 = 1 \cdot x = x, \quad \frac{\partial \ell}{\partial b} = (\phi(x) - y) \cdot 1 = 1.$$

Updates:

$$w' = w - \lambda x, \quad b' = b - \lambda.$$

Since $x > 0$, $\lambda x > 0$, so $w' < w$. If $w' > 0$, the slope decreases but remains positive. If $w' \leq 0$, the slope becomes non-positive, but let's assume a small λ keeps $w' > 0$. The bias decreases ($b' < b$).

New elbow:

$$e' = -\frac{b'}{w'} = -\frac{b-\lambda}{w-\lambda x}.$$

Numerically, if $w = 1, b = 1, x = 1, \lambda = 0.1$:

$$w' = 1 - 0.1 \cdot 1 = 0.9, \quad b' = 1 - 0.1 = 0.9, \quad e' = -\frac{0.9}{0.9} = -1.$$

Original elbow: $e = -\frac{b}{w} = -\frac{1}{1} = -1$. The elbow may shift depending on x, λ. Generally, the function shifts downward, reducing $\phi(x) = wx + b$.

Sketch: Plot $\phi(x)$ with slope $w > 0$, elbow at $e = -\frac{b}{w}$, and point (x, y) where $\phi(x) = wx + b = y + 1$. Post-update, the slope decreases, and the elbow may shift right if $\frac{b}{w} < \frac{b'}{w'}$.

(iii) Case: $w > 0, x < 0, \phi(x) > 0$

Since $\phi(x) > 0, wx + b > 0$, and:

$$\frac{\partial \ell}{\partial w} = 1 \cdot x = x, \quad \frac{\partial \ell}{\partial b} = 1.$$

Since $x < 0$:

$$w' = w - \lambda x = w + \lambda |x|, \quad b' = b - \lambda.$$

The slope increases ($w' > w$), and the bias decreases. New elbow:

$$e' = -\frac{b-\lambda}{w + \lambda |x|}.$$

Since the denominator increases and numerator decreases, $|e'| < |e|$, shifting the elbow toward zero.

Sketch: Slope increases, elbow shifts left (toward zero), function lowers at x.

(iv) Case: $w < 0, x > 0, \phi(x) > 0$

Since $\phi(x) > 0, wx + b > 0$. Gradients:

$$\frac{\partial \ell}{\partial w} = x, \quad \frac{\partial \ell}{\partial b} = 1.$$

Updates:

$$w' = w - \lambda x, \quad b' = b - \lambda.$$

Since $x > 0, w' < w$ (more negative), increasing the magnitude of the negative slope. New elbow:

$$e' = -\frac{b - \lambda}{w - \lambda x}.$$

Numerically, if $w = -1, b = 2, x = 1, \lambda = 0.1$:

$$w' = -1 - 0.1 \cdot 1 = -1.1, \quad b' = 2 - 0.1 = 1.9, \quad e' = -\frac{1.9}{-1.1} \approx 1.727, \quad e = -\frac{2}{-1}$$

The elbow shifts left.

Sketch: Slope becomes steeper (more negative), elbow shifts left, function lowers at x
.

Part (c): Elbow Location for the Full Network

The network is:

$$\hat{f}(x) = W^{(2)} \Phi(W^{(1)} x + \mathbf{b}),$$

where $W^{(1)} = [w_1, \ldots, w_d]^T$, $\mathbf{b} = [b_1, \ldots, b_d]^T$, $W^{(2)} = [u_1, \ldots, u_d]$. The i-th ReLU activation is:

$$\Phi_i = \max(0, w_i x + b_i).$$

The elbow e_i is where:

$$w_i x + b_i = 0 \implies e_i = -\frac{b_i}{w_i}, \quad (w_i \neq 0).$$

Part (d): New Elbow Location After SGD Update

The loss is:

$$\ell = \frac{1}{2}(\hat{f}(x) - y)^2, \quad \hat{f}(x) = \sum_{i=1}^{d} u_i \max(0, w_i x + b_i).$$

Let $z_i = w_i x + b_i$, so $\Phi_i = \max(0, z_i)$, and:

$$\hat{f}(x) = \sum_{i=1}^{d} u_i \Phi_i.$$

Gradients:

- $\frac{\partial \ell}{\partial u_i} = (\hat{f}(x) - y) \cdot \Phi_i$,
- $\frac{\partial \ell}{\partial w_i} = (\hat{f}(x) - y) \cdot u_i \cdot x \cdot 1(z_i > 0)$,

- $\frac{\partial \ell}{\partial b_i} = (\hat{f}(x) - y) \cdot u_i \cdot 1(z_i > 0)$.

Updates:

$$w_i' = w_i - \lambda(\hat{f}(x) - y)u_i x 1(z_i > 0), \quad b_i' = b_i - \lambda(\hat{f}(x) - y)u_i 1(z_i > 0).$$

New elbow:

$$e_i' = -\frac{b_i'}{w_i'} = -\frac{b_i - \lambda(\hat{f}(x) - y)u_i 1(z_i > 0)}{w_i - \lambda(\hat{f}(x) - y)u_i x 1(z_i > 0)}.$$

If $z_i \leq 0$, then $1(z_i > 0) = 0$, so $w_i' = w_i$, $b_i' = b_i$, and $e_i' = e_i$. If $z_i > 0$, the elbow shifts based on $x, u_i, \lambda, \hat{f}(x) - y$.

Summary

- **(a)** Elbow: $e = -\frac{b}{w}$; derivatives: $\frac{d\ell}{dz} = \phi(x) - y$, $\frac{\partial \ell}{\partial w} = (\phi(x) - y)x1(wx + b > 0)$, $\frac{\partial \ell}{\partial b} = (\phi(x) - y)1(wx + b > 0)$.
- **(b)** SGD updates adjust slope and elbow based on w, x signs, shifting the function downward.
- **(c)** Elbow for i-th ReLU: $e_i = -\frac{b_i}{w_i}$.
- **(d)** New elbow after SGD: Depends on whether the ReLU is active, shifting only for active units.

These results illustrate how SGD adjusts ReLU-based networks, critical for understanding optimization in deep learning.

↳ Numerical elbow shift examples

↳ ReLU vs Leaky ReLU