

Deep Learning and Linear Algebra Expert Assistance

<https://gemini.google.com/share/89b0a83f691b>  

Created with Thinking with 3 Pro 2 December 2025 at 05:55 pm • Published on 3 December 2025 at 03:18 pm



You are a deep learning and linear algebra expert. You will be given undergraduate level deep learning problems. Please solve them step by step and explain your reasoning.



I am ready. As an expert in both Deep Learning and Linear Algebra, I am well-equipped to guide you through undergraduate-level problems.

Whether you are dealing with the geometry of high-dimensional spaces, matrix calculus for backpropagation, or the specifics of model architectures, I will break down the solutions to ensure you understand both the **mathematical mechanics** and the **conceptual intuition**.

To give you the best assistance, I will structure my responses as follows:

- **Problem Statement:** Describes the problem or task at hand.
- **Mathematical Setup:** Provides context (e.g., $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{m \times n}$).
- **Step-by-Step Derivation:** Breaks down the solution into steps involving matrix multiplication and other operations.
- **Intuition:** Explaining why the steps work, often in the context of deep learning.

[Continue where they left off](#)

You can add this chat to your own account to continue to create something new

[Tell me more](#) [Dismiss](#)

...lly regarding matrix

[Continue this chat](#)

Gemini may display inaccurate info, including about people, so double-check its responses.

- **Forward Propagation:** Computing outputs using affine transformations and non-linearities (e.g., $\mathbf{a} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$).
- **Matrix Calculus & Gradients:** Deriving gradients for backpropagation (e.g., $\frac{\partial L}{\partial \mathbf{W}}$).
- **Loss Functions:** Analyzing MSE, Cross-Entropy, and their geometric properties.
- **Optimization:** Understanding SGD, momentum, and Hessian-based methods.
- **Dimensionality Reduction:** PCA, SVD, and their relationship to autoencoders.

Please paste the first problem or set of problems you would like us to solve.

2. Vector Calculus Review

Let $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$. 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: When calculating derivatives, remember that derivatives of vectors with respect to a column vector are expressed as a row vector, i.e. $\frac{\partial}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial}{\partial x_1} & \dots & \frac{\partial}{\partial x_n} \end{pmatrix}$ 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 these types.

Homework 0 © Faculty teaching EBCS 102, Fall 2025. All rights reserved. You may not publicly share without explicit permission.

Homework 0 © 2025-09-29 14:73 (01-07-0)

- Show $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top \mathbf{c}) = \mathbf{c}^\top$
- Show $\frac{\partial}{\partial \mathbf{x}}(\|\mathbf{x}\|) = 2\mathbf{x}^\top$
- Show $\frac{\partial}{\partial \mathbf{x}}(A\mathbf{x}) = A$
- Show $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^\top A\mathbf{x}) = \mathbf{x}^\top(A + A^\top)$
- Under what condition is the previous derivative equal to $2\mathbf{x}^\top A$?



This is a classic "Vector Calculus 101" assignment for Deep Learning. Mastering these derivatives is crucial because modern backpropagation in libraries like PyTorch or TensorFlow is essentially applying these rules (and the chain rule) repeatedly.

The prompt explicitly defines the **numerator layout convention** (often used in econometrics and some ML texts), where the derivative of a scalar f with respect to a column vector \mathbf{x} is a **row**.

vector:

$$\frac{\partial f}{\partial \mathbf{x}} = \left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right]$$

Let's solve each part step-by-step, ensuring we analyze dimensions first as requested.

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

1. Dimensionality Analysis

- **Input:** \mathbf{x} is a column vector $\in \mathbb{R}^{n \times 1}$.
- **Function:** $f(\mathbf{x}) = \mathbf{x}^T \mathbf{c}$. Since \mathbf{x}^T is $(1 \times n)$ and \mathbf{c} is $(n \times 1)$, the result is a **scalar** (1×1) .
- **Derivative Shape:** Since the output is a scalar and input is a column vector, the derivative must be a **row vector** $\in \mathbb{R}^{1 \times n}$.

2. Step-by-Step Derivation First, let's write out the dot product in summation notation. Let vectors \mathbf{x} and \mathbf{c} have components x_i and c_i :

$$f(\mathbf{x}) = \sum_{i=1}^n x_i c_i = x_1 c_1 + x_2 c_2 + \dots + x_n c_n$$

Now, we calculate the partial derivative with respect to an arbitrary k -th entry, x_k :

$$\frac{\partial f}{\partial x_k} = \frac{\partial}{\partial x_k} (x_1 c_1 + \dots + x_k c_k + \dots + x_n c_n)$$

Since x_i and x_k are independent when $i \neq k$, all terms except the k -th term vanish:

$$\frac{\partial f}{\partial x_k} = c_k$$

3. Stacking the Result We arrange these partials into a row vector:

$$\frac{\partial f}{\partial \mathbf{x}} = [c_1, c_2, \dots, c_n] = \mathbf{c}^T$$

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

1. Dimensionality Analysis

- **Input:** $\mathbf{x} \in \mathbb{R}^{n \times 1}$.
- **Function:** $f(\mathbf{x}) = \|\mathbf{x}\|_2^2$ (squared L2 norm). This is a **scalar**.
- **Derivative Shape:** **Row vector** $\in \mathbb{R}^{1 \times n}$.

2. Step-by-Step Derivation The squared L2 norm is the dot product of the vector with itself:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{x} = \sum_{i=1}^n x_i^2 = x_1^2 + x_2^2 + \dots + x_n^2$$

Take the partial derivative with respect to an arbitrary entry x_k :

$$\frac{\partial f}{\partial x_k} = \frac{\partial}{\partial x_k} (x_1^2 + \dots + x_k^2 + \dots + x_n^2)$$

$$\frac{\partial f}{\partial x_k} = \frac{d}{dx_k} (x_k^2) = 2x_k$$

3. Stacking the Result

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

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

1. Dimensionality Analysis

- **Input:** $\mathbf{x} \in \mathbb{R}^{n \times 1}$.
- **Function:** $\mathbf{y} = \mathbf{A}\mathbf{x}$. Since $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{x} \in \mathbb{R}^{n \times 1}$, the output \mathbf{y} is a **column vector** $\in \mathbb{R}^{n \times 1}$.
- **Derivative Shape:** We are differentiating a vector w.r.t. a vector. The result is the **Jacobian Matrix**. Based on the prompt's row-vector convention for scalars, the i -th row of the Jacobian corresponds to the derivative of the i -th scalar output y_i .
 - Shape: (Output dim) \times (Input dim) = $n \times n$.

2. Step-by-Step Derivation

Let's look at the i -th component of the output vector \mathbf{y} , denoted as y_i :

$$y_i = (\mathbf{A}\mathbf{x})_i = \sum_{j=1}^n A_{ij}x_j$$

This y_i is a scalar. Its derivative w.r.t \mathbf{x} is a row vector (as established in problem a). Let's find the partial derivative of y_i with respect to x_k :

$$\frac{\partial y_i}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{j=1}^n A_{ij}x_j$$

Only the term where $j = k$ survives:

$$\frac{\partial y_i}{\partial x_k} = A_{ik}$$

3. Stacking the Result The entry at row i and column k of our derivative matrix is A_{ik} .

$$\left(\frac{\partial(\mathbf{A}\mathbf{x})}{\partial \mathbf{x}} \right)_{ik} = A_{ik}$$

Therefore, the full matrix is simply \mathbf{A} .

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

1. Dimensionality Analysis

- **Input:** $\mathbf{x} \in \mathbb{R}^{n \times 1}$.
- **Function:** $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$. This is a **scalar** (known as a Quadratic Form).
 - $(1 \times n) \times (n \times n) \times (n \times 1) \rightarrow (1 \times 1)$.
- **Derivative Shape:** **Row vector** $\in \mathbb{R}^{1 \times n}$.

2. Step-by-Step Derivation

Let's expand the quadratic form into a double summation:

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

We need the partial derivative w.r.t x_k . Note that x_k appears in this sum in two ways:

1. When $i = k$ (row index is k)
2. When $j = k$ (column index is k)

Using the product rule on the term $x_i A_{ij} x_j$:

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

The derivative $\frac{\partial}{\partial x_k}(x_i x_j)$ is non-zero only if $i = k$ or $j = k$:

- **Case 1** ($i = k, j \neq k$): Derivative is x_j . Term is $A_{kj} x_j$.
- **Case 2** ($i \neq k, j = k$): Derivative is x_i . Term is $x_i A_{ik}$.

- **Case 3** ($i = k, j = k$): Derivative of x_k^2 is $2x_k$. (This is covered by adding Case 1 and 2 together).

So, we sum over the indices where x_k appears:

$$\frac{\partial f}{\partial x_k} = \sum_{j=1}^n A_{kj}x_j \quad (\text{from } i = k) \quad + \quad \sum_{i=1}^n x_i A_{ik} \quad (\text{from } j = k)$$

3. Stacking the Result Let's analyze the two resulting sums:

1. $\sum_{j=1}^n A_{kj}x_j$ is the dot product of the k -th **row** of A with \mathbf{x} . This corresponds to the k -th element of the vector $(A\mathbf{x})$.
2. $\sum_{i=1}^n x_i A_{ik}$ is the dot product of \mathbf{x} with the k -th **column** of A . This corresponds to the k -th element of the vector $(\mathbf{x}^T A)$ or $(A^T \mathbf{x})^T$.

Since we need a row vector output:

- First term collected for all k : $(A\mathbf{x})^T = \mathbf{x}^T A^T$.
- Second term collected for all k : $\mathbf{x}^T A$.

$$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{x}^T A^T + \mathbf{x}^T A = \mathbf{x}^T (A^T + A)$$

Note: Addition is commutative, so $\mathbf{x}^T (A + A^T)$ is the same.

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

We derived the derivative as $\mathbf{x}^T (A + A^T)$. We want to know when:

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

$$\mathbf{x}^T A + \mathbf{x}^T A^T = \mathbf{x}^T A + \mathbf{x}^T A$$

Subtract $\mathbf{x}^T A$ from both sides:

$$\mathbf{x}^T A^T = \mathbf{x}^T A$$

This implies that A^T must equal A .

Answer: The condition is that A must be a **Symmetric Matrix** ($A = A^T$).

Intuition: This is very common in Deep Learning. For example, the Hessian matrix (matrix of second derivatives) is always symmetric, so the derivative of the quadratic form involving the Hessian simplifies to $2\mathbf{x}^T H$.

Summary of Results

- (a) \mathbf{c}^T (b) $2\mathbf{x}^T$ (c) A (d) $\mathbf{x}^T (A + A^T)$ (e) A is Symmetric ($A = A^T$)

Would you like to walk through how these apply to a specific Deep Learning concept, like deriving the gradients for a single Linear Layer?

3. Least Squares and the Min-norm problem from the Perspective of SVD
 Consider the equation $\mathbf{X}\mathbf{w} = \mathbf{y}$, where $\mathbf{X} \in \mathbb{R}^{m \times n}$ is a non-square data matrix, \mathbf{w} is a weight vector, and \mathbf{y} is vector of labels corresponding to the datapoints in each row of \mathbf{X} .
 Let's say that $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ is the (full) SVD of \mathbf{X} . Here, \mathbf{U} and \mathbf{V} are orthonormal square matrices, and \mathbf{S} is an $m \times n$ matrix with non-zero singular values (s_i) on the "diagonal".
 For this problem, we define Σ' an $n \times m$ matrix with the reciprocals of the singular values ($\frac{1}{s_i}$) along the "diagonal".

- (a) First, consider the case where $m > n$, i.e. our data matrix \mathbf{X} has more rows than columns (full matrix) and the system is overdetermined. How do we find the weights \mathbf{w} that minimizes the error between $\mathbf{X}\mathbf{w}$ and \mathbf{y} ? In other words, we want to solve $\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$.



solve question 3 part a



- This problem asks us to solve the classic **Least Squares** problem for an overdetermined system (where $m > n$, meaning more equations than unknowns).

Since the problem title mentions "from the Perspective of SVD," we will first derive the standard solution using calculus (the Normal Equations) and then substitute the SVD components to express the final answer in terms of U , Σ , and V .

Problem Statement

We want to find the vector \mathbf{w} that minimizes the squared Euclidean norm of the error:

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

where $\mathbf{X} \in \mathbb{R}^{m \times n}$ has rank n (implied by non-zero singular values).

Step 1: The Calculus Approach (Normal Equations)

First, let's derive the solution algebraically. We define the loss function $J(\mathbf{w})$:

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

Expanding this term (recall from the vector calculus review that $(A - B)^T = A^T - B^T$):

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

To minimize, we take the gradient with respect to \mathbf{w} and set it to zero. Using the identities $\nabla_{\mathbf{w}}(\mathbf{w}^T \mathbf{A} \mathbf{w}) = 2\mathbf{A}\mathbf{w}$ (for symmetric \mathbf{A}) and $\nabla_{\mathbf{w}}(\mathbf{w}^T \mathbf{c}) = \mathbf{c}$:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 2\mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{X}^T \mathbf{y} = 0$$

Rearranging gives us the famous **Normal Equations**:

$$\mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{y}$$

Assuming \mathbf{X} has full column rank (which implies $\mathbf{X}^T \mathbf{X}$ is invertible), we solve for \mathbf{w} :

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

Step 2: The SVD Perspective

Now, we substitute the SVD definition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ into our derived solution.

- \mathbf{U} and \mathbf{V} are orthogonal/unitary, so $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T \mathbf{V} = \mathbf{I}$.
- Σ is an $m \times n$ diagonal matrix.

Substitute into $\mathbf{X}^T \mathbf{X}$:

$$\mathbf{X}^T \mathbf{X} = (\mathbf{U}\Sigma\mathbf{V}^T)^T (\mathbf{U}\Sigma\mathbf{V}^T) = \mathbf{V}\Sigma^T \mathbf{U}^T \mathbf{U}\Sigma\mathbf{V}^T$$

Since $\mathbf{U}^T \mathbf{U} = \mathbf{I}$:

$$\mathbf{X}^T \mathbf{X} = \mathbf{V}(\Sigma^T \Sigma)\mathbf{V}^T$$

Invert $(\mathbf{X}^T \mathbf{X})^{-1}$:

$$(\mathbf{X}^T \mathbf{X})^{-1} = (\mathbf{V}(\Sigma^T \Sigma)\mathbf{V}^T)^{-1} = \mathbf{V}(\Sigma^T \Sigma)^{-1}\mathbf{V}^T$$

(Note: Since \mathbf{V} is square and orthogonal, $\mathbf{V}^{-1} = \mathbf{V}^T$.)

Substitute back into the full expression for \mathbf{w} :

$$\mathbf{w} = \underbrace{\mathbf{V}(\Sigma^T \Sigma)^{-1} \mathbf{V}^T}_{(\mathbf{X}^T \mathbf{X})^{-1}} \underbrace{\mathbf{V}\Sigma^T \mathbf{U}^T}_{\mathbf{X}^T} \mathbf{y}$$

$$\mathbf{w} = \mathbf{V}(\Sigma^T \Sigma)^{-1} (\mathbf{V}^T \mathbf{V}) \Sigma^T \mathbf{U}^T \mathbf{y}$$

Since $\mathbf{V}^T \mathbf{V} = \mathbf{I}$:

$$\mathbf{w} = V \underbrace{(\Sigma^T \Sigma)^{-1} \Sigma^T}_{\text{Simplify this}} U^T \mathbf{y}$$

Simplifying the middle term: Σ is an $m \times n$ matrix with singular values σ_i on the diagonal. The product $(\Sigma^T \Sigma)^{-1} \Sigma^T$ results in an $n \times m$ matrix with $\frac{1}{\sigma_i}$ on the diagonal. This matches the problem's definition of Σ^\dagger .

$$(\Sigma^T \Sigma)^{-1} \Sigma^T = \Sigma^\dagger$$

Final Answer

The weights \mathbf{w} that minimize the error are given by:

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

This specific matrix product $X^\dagger = V \Sigma^\dagger U^T$ is known as the **Moore-Penrose Pseudoinverse** of X .

Intuitive Summary:

1. $U^T \mathbf{y}$: Rotates the labels \mathbf{y} into the basis of the "data space".
2. Σ^\dagger : Scales each coordinate by $1/\sigma_i$ (inverting the stretching effect of the data matrix X).
3. V : Rotates the result back into the "weight space" to give the final coefficients \mathbf{w} .

(b) Plug in the SVD $X = U \Sigma V^T$ and simplify. Be careful with dimensions!

- ◆ Based on the images provided, Question 3(a) asks for the derivation of the weights \mathbf{w} that minimize the squared error for an overdetermined system (Least Squares). Since part (b) explicitly asks to "Plug in the SVD," the expected answer for part (a) is the standard algebraic derivation using matrix calculus to arrive at the **Normal Equations**.

Here is the step-by-step solution for **Question 3(a)**.

Problem Statement

We want to find the vector \mathbf{w} that minimizes the objective function:

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

where $X \in \mathbb{R}^{m \times n}$ (with $m > n$) and $\mathbf{y} \in \mathbb{R}^m$.

Step 1: Expand the Objective Function

Let $L(\mathbf{w})$ be the loss function (the squared L_2 norm). Recall that for any vector \mathbf{v} , $\|\mathbf{v}\|_2^2 = \mathbf{v}^T \mathbf{v}$.

$$L(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^T (X\mathbf{w} - \mathbf{y})$$

We distribute the transpose term $(X\mathbf{w} - \mathbf{y})^T = \mathbf{w}^T X^T - \mathbf{y}^T$:

$$L(\mathbf{w}) = (\mathbf{w}^T X^T - \mathbf{y}^T)(X\mathbf{w} - \mathbf{y})$$

Now, expand the multiplication:

$$L(\mathbf{w}) = \mathbf{w}^T X^T X \mathbf{w} - \mathbf{w}^T X^T \mathbf{y} - \mathbf{y}^T X \mathbf{w} + \mathbf{y}^T \mathbf{y}$$

Simplification Note: The terms $\mathbf{w}^T X^T \mathbf{y}$ and $\mathbf{y}^T X \mathbf{w}$ are scalars (results of dot products) and are transposes of each other. Since the transpose of a scalar is itself, these two terms are identical.

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

Step 2: Compute the Gradient

To find the minimum, we take the gradient of $L(\mathbf{w})$ with respect to \mathbf{w} and set it to zero. We use the matrix calculus identities derived in the "Vector Calculus Review" section of your homework:

1. **Quadratic Term:** $\frac{\partial}{\partial \mathbf{w}} (\mathbf{w}^T A \mathbf{w}) = 2\mathbf{w}^T A$ (if A is symmetric).
 - Here, $A = X^T X$, which is symmetric.
 - Derivative: $2\mathbf{w}^T X^T X$.
2. **Linear Term:** $\frac{\partial}{\partial \mathbf{w}} (-2\mathbf{w}^T \mathbf{b}) = -2\mathbf{b}^T$.
 - Here, $\mathbf{b} = X^T \mathbf{y}$.
 - Derivative: $-2(X^T \mathbf{y})^T = -2\mathbf{y}^T X$.
3. **Constant Term:** $\frac{\partial}{\partial \mathbf{w}} (\mathbf{y}^T \mathbf{y}) = 0$.

Putting it all together (and keeping the row-vector convention from the assignment):

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{w}^T X^T X - 2\mathbf{y}^T X$$

Step 3: Solve for \mathbf{w} (The Normal Equations)

Set the gradient to zero to find the critical point. (Transposing both sides to work with column vectors is standard practice here):

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

$$X^T X \mathbf{w} = X^T \mathbf{y}$$

This equation is known as the **Normal Equation**. To solve for \mathbf{w} , we multiply by the inverse of $(X^T X)$. Since X has full rank (implied by non-zero singular values in the problem description), $(X^T X)$ is invertible.

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

Final Answer for 3(a)

The weights that minimize the error are:

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

(Note: In part (b), you will plug the SVD components into this equation to derive the Pseudo-inverse form.)

(c) You'll notice that the least-squares solution is in the form $\mathbf{w}^* = A\mathbf{y}$. What happens if we left-multiply X by our matrix A ? This is why the matrix A of the least-squares solution is called the left-inverse.

- ◆ Based on the images you provided, Question 3(a) asks for the derivation of the optimal weights \mathbf{w} for the **Least Squares** problem where the system is overdetermined ($m > n$, i.e., a "tall" matrix).

Since Question 3(b) explicitly asks to "Plug in the SVD," the goal of 3(a) is to perform the standard matrix calculus derivation to arrive at the **Normal Equations**.

Here is the step-by-step solution consistent with the notation and conventions established in your "Vector Calculus Review" image.

3(a) Derivation of the Least Squares Solution

1. **Define the Objective Function** We want to minimize the squared Euclidean norm of the error:

$$J(\mathbf{w}) = \|X\mathbf{w} - \mathbf{y}\|^2$$

2. **Expand the Squared Norm** Recall that $\|\mathbf{v}\|^2 = \mathbf{v}^T \mathbf{v}$. Let $\mathbf{v} = X\mathbf{w} - \mathbf{y}$.

$$J(\mathbf{w}) = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Distribute the transpose $(\mathbf{X}\mathbf{w} - \mathbf{y})^T = \mathbf{w}^T\mathbf{X}^T - \mathbf{y}^T$:

$$J(\mathbf{w}) = (\mathbf{w}^T\mathbf{X}^T - \mathbf{y}^T)(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Multiply the terms out:

$$J(\mathbf{w}) = \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y}$$

Simplification: The terms $\mathbf{w}^T\mathbf{X}^T\mathbf{y}$ and $\mathbf{y}^T\mathbf{X}\mathbf{w}$ are scalars and are transposes of each other. Since the transpose of a scalar is the scalar itself, they are equal. We combine them:

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

3. Compute the Gradient We take the derivative with respect to \mathbf{w} . Based on the **Vector Calculus Review** image you uploaded, we use the convention where the derivative of a scalar w.r.t a column vector is a **row vector**.

Using the identities derived in the previous problem:

- **Quadratic term:** $\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T A \mathbf{w}) = \mathbf{w}^T(A + A^T)$. Here $A = \mathbf{X}^T\mathbf{X}$, which is symmetric, so this becomes $2\mathbf{w}^T\mathbf{X}^T\mathbf{X}$.
- **Linear term:** $\frac{\partial}{\partial \mathbf{w}}(-2\mathbf{y}^T\mathbf{X}\mathbf{w})$. This is of the form $\mathbf{c}^T\mathbf{w}$ (where $\mathbf{c}^T = -2\mathbf{y}^T\mathbf{X}$). The derivative is simply the row vector coefficient: $-2\mathbf{y}^T\mathbf{X}$.
- **Constant term:** $\frac{\partial}{\partial \mathbf{w}}(\mathbf{y}^T\mathbf{y}) = 0$.

Combining these:

$$\frac{\partial J}{\partial \mathbf{w}} = 2\mathbf{w}^T\mathbf{X}^T\mathbf{X} - 2\mathbf{y}^T\mathbf{X}$$

4. Set Gradient to Zero and Solve To find the minimum, set the derivative to zero:

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

Divide by 2 and transpose the entire equation to work with column vectors (which is the standard form for the final answer):

$$(\mathbf{X}^T\mathbf{X})^T\mathbf{w} = (\mathbf{X}^T)^T\mathbf{y}$$

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

This result is known as the **Normal Equations**.

Assuming \mathbf{X} has full column rank (which is true since the problem states the singular values are non-zero), $\mathbf{X}^T\mathbf{X}$ is invertible. We multiply by the inverse to isolate \mathbf{w} :

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

Summary

The weights \mathbf{w} that minimize the error are given by:

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

Would you like me to proceed to **Part (b)** and plug in the SVD components (U, Σ, V^T) to simplify this expression?

(d) Now, let's consider the case where $m < n$, i.e. the data matrix \mathbf{X} has more columns than rows and the system is underdetermined. There exist infinitely many solutions for \mathbf{w} , but we seek the minimum norm solution, i.e. we want to solve $\min\|\mathbf{w}\|^2$ s.t. $\mathbf{X}\mathbf{w} = \mathbf{y}$. What is the minimum norm solution?

- ◆ This problem asks for the **minimum norm solution** for an underdetermined system ($m < n$). In this scenario, there are infinitely many solutions that satisfy $X\mathbf{w} = \mathbf{y}$, so we need to pick the specific \mathbf{w} that is "smallest" (closest to the origin).

This is a classic **Constrained Optimization** problem.

Problem Statement

We want to solve:

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|_2^2$$

subject to $X\mathbf{w} = \mathbf{y}$

(Note: Minimizing $\frac{1}{2} \|\mathbf{w}\|^2$ is equivalent to minimizing $\|\mathbf{w}\|$ but makes the derivatives cleaner.)

Step 1: Set up the Lagrangian

To solve a constrained optimization problem, we use **Lagrange Multipliers**. We introduce a vector of Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^m$ (one for each constraint equation in $X\mathbf{w} = \mathbf{y}$).

The Lagrangian function $\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})$ is:

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

Step 2: Find the Gradient

We need to find the critical points by taking the derivative with respect to \mathbf{w} and setting it to zero.

Using the row-vector convention established in your homework:

1. **Derivative of the objective:** $\frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2} \mathbf{w}^T \mathbf{w} \right) = \mathbf{w}^T$
2. **Derivative of the constraint term:** The term $\boldsymbol{\lambda}^T X\mathbf{w}$ is linear in \mathbf{w} . The coefficient matrix acting on \mathbf{w} is $X^T \boldsymbol{\lambda}$. So, the derivative is $X^T \boldsymbol{\lambda}$.

Set the sum to zero:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w}^T + \boldsymbol{\lambda}^T X = \mathbf{0}^T$$

Transpose this equation to work with column vectors:

$$\mathbf{w} + X^T \boldsymbol{\lambda} = \mathbf{0}$$

$$\mathbf{w} = -X^T \boldsymbol{\lambda}$$

Intuition: This tells us that the optimal solution \mathbf{w}^* must lie in the row space of X (the span of the columns of X^T).

Step 3: Solve for the Multipliers ($\boldsymbol{\lambda}$)

Now we substitute our expression for \mathbf{w} back into the original constraint equation $X\mathbf{w} = \mathbf{y}$:

$$X(-X^T \boldsymbol{\lambda}) = \mathbf{y}$$

$$-(XX^T)\boldsymbol{\lambda} = \mathbf{y}$$

Since X is a "fat" matrix ($m < n$) and typically has full row rank (m), the matrix XX^T is an $m \times m$ invertible matrix. We can solve for $\boldsymbol{\lambda}$:

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

Step 4: Final Solution

Substitute λ back into the equation for \mathbf{w} :

$$\mathbf{w} = -X^T \lambda$$

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

The negatives cancel out, giving us the final result:

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

Summary

The minimum norm solution is:

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

Connection to Part (c): In part (c), you saw the "Left Inverse" for the overdetermined case.

Here, the matrix $X^T(XX^T)^{-1}$ is known as the **Right Inverse** (or Right Pseudo-inverse) of X . If you right-multiply X by this matrix, you get the identity:

$$X \cdot [X^T(XX^T)^{-1}] = (XX^T)(XX^T)^{-1} = I$$

Would you like to solve the next step (plugging in the SVD for this case) or move to a different problem?

(e) Plug in the SVD $X = U\Sigma V^T$ and simplify. Be careful with dimensions!



Based on your uploaded images, here is a step-by-step solution for **Question 3(e)**.

This problem asks you to take the algebraic solution you found in part (d) for the **minimum norm solution** (underdetermined system) and plug in the SVD components to simplify it.

Problem Statement

From part (d), we know the minimum norm solution is:

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

We are given the SVD: $X = U\Sigma V^T$, where:

- U is an $m \times m$ orthogonal matrix ($U^T = U^{-1}$).
- V is an $n \times n$ orthogonal matrix ($V^T = V^{-1}$).
- Σ is an $m \times n$ rectangular diagonal matrix (fat matrix, since $m < n$).

Step 1: Substitute SVD into the term (XX^T)

First, let's simplify the term inside the inverse, XX^T .

$$XX^T = (U\Sigma V^T)(U\Sigma V^T)^T$$

Distribute the transpose (remembering to reverse the order):

$$XX^T = U\Sigma V^T V\Sigma^T U^T$$

Since V is orthogonal, $V^T V = I$ (the identity matrix):

$$XX^T = U(\Sigma\Sigma^T)U^T$$

Step 2: Invert (XX^T)

Now we need the inverse of the expression we just found: $(U(\Sigma\Sigma^T)U^T)^{-1}$. Using the property $(ABC)^{-1} = C^{-1}B^{-1}A^{-1}$ and the fact that $U^{-1} = U^T$:

$$(XX^T)^{-1} = (U^T)^{-1}(\Sigma\Sigma^T)^{-1}U^{-1}$$

$$(XX^T)^{-1} = U(\Sigma\Sigma^T)^{-1}U^T$$

Step 3: Substitute back into the full equation for \mathbf{w}

Recall the equation $\mathbf{w} = X^T(XX^T)^{-1}\mathbf{y}$. Let's plug in $X^T = (U\Sigma V^T)^T = V\Sigma^T U^T$ and the inverse term we just calculated.

$$\mathbf{w} = \underbrace{V\Sigma^T U^T}_{X^T} \underbrace{U(\Sigma\Sigma^T)^{-1}U^T}_{(XX^T)^{-1}} \mathbf{y}$$

Notice that $U^T U = I$ in the middle:

$$\mathbf{w} = V \underbrace{\Sigma^T (\Sigma\Sigma^T)^{-1}}_{\text{Simplify this}} U^T \mathbf{y}$$

Step 4: Simplify the Diagonal Matrices

Let's look closely at the term $\Sigma^T(\Sigma\Sigma^T)^{-1}$.

- Since $m < n$ (underdetermined), Σ is a "fat" matrix.
- $\Sigma\Sigma^T$ results in an $m \times m$ square diagonal matrix containing σ_i^2 .
- $(\Sigma\Sigma^T)^{-1}$ is a diagonal matrix containing $1/\sigma_i^2$.
- Multiplying Σ^T (which has σ_i on the diagonal) by this inverse scales the entries.

$$\text{Entry } i \rightarrow \sigma_i \cdot \frac{1}{\sigma_i^2} = \frac{1}{\sigma_i}$$

The result is an $n \times m$ matrix with $1/\sigma_i$ on the diagonal. This matches the definition of Σ^\dagger (the pseudoinverse of Σ) provided in the problem description.

$$\Sigma^T(\Sigma\Sigma^T)^{-1} = \Sigma^\dagger$$

Final Answer

Substituting this back into our equation:

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

Observation: You should notice that this is the **exact same formula** as the solution for the overdetermined (Least Squares) case in part (b). This is the beauty of the SVD and the Moore-Penrose Pseudoinverse—it provides a unified mathematical form ($X^\dagger = V\Sigma^\dagger U^T$) that solves both the Least Squares problem ($m > n$) and the Minimum Norm problem ($m < n$).



(?) You'll notice that the min-norm solution is in the form $\mathbf{w} = B\mathbf{y}$. 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.

- ◆ Based on the problem statement in the image, here is the step-by-step solution for **Question 3(f)**.

This question asks to verify the property of the matrix B derived in the minimum-norm solution and explain why it is called a "right-inverse."

1. Identify the Matrix B

From your solution to **part (d)** (the minimum norm problem), we established that the optimal weights are given by:

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

Therefore, our matrix B is:

$$B = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$$

2. Perform the Right-Multiplication

The question asks what happens when we **right-multiply** \mathbf{X} by B . We compute the product $\mathbf{X}B$:

$$\mathbf{X}B = \mathbf{X}[\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}]$$

Using the associative property of matrix multiplication, we can group the first two terms (\mathbf{X} and \mathbf{X}^T):

$$\mathbf{X}B = (\mathbf{X}\mathbf{X}^T)(\mathbf{X}\mathbf{X}^T)^{-1}$$

3. Simplify

The term $(\mathbf{X}\mathbf{X}^T)$ is an $m \times m$ square matrix. Assuming \mathbf{X} has full row rank (which is typical for the underdetermined "fat matrix" case where $m < n$), $(\mathbf{X}\mathbf{X}^T)$ is invertible.

Any matrix multiplied by its inverse yields the Identity matrix I :

$$\mathbf{X}B = I_m$$

(Where I_m is the $m \times m$ identity matrix.)

Conclusion

Result: The product is the **Identity Matrix**.

Reasoning: Because multiplying \mathbf{X} by B on the **right** side results in the identity ($\mathbf{X}B = I$), the matrix B is formally known as a **Right Inverse** of \mathbf{X} .

- Note: Unlike square invertible matrices where the left and right inverses are the same, rectangular matrices often have different one-sided inverses.
 - In the overdetermined case (Least Squares), we used $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$, which is a **Left Inverse** ($\mathbf{A}\mathbf{X} = I$).
 - In this underdetermined case (Min-Norm), we use $\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$, which is a **Right Inverse** ($\mathbf{X}B = I$).

Would you like to move on to the next problem set or discuss the geometric intuition behind these two inverses?

4. The 5 Interpretations of Ridge Regression

(a) **Perspective 1: Optimization Problem**: Ridge regression can be understood as the unconstrained optimization problem

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2, \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a data matrix, and $\mathbf{y} \in \mathbb{R}^n$ is the target vector of measurement values. What's new compared to the simple OLS problem is the addition of the $\lambda \|\mathbf{w}\|_2^2$ 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 \mathbf{w} .

- ◆ Based on the problem description in **Question 4(a)**, here is the step-by-step derivation for the Ridge Regression solution.

Problem Statement

We want to find the weights \mathbf{w} that minimize the Ridge Regression objective function:

$$J(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

Where:

where:

- $X \in \mathbb{R}^{n \times d}$ is the data matrix.
- $\mathbf{y} \in \mathbb{R}^n$ is the target vector.
- $\lambda \|\mathbf{w}\|_2^2$ is the regularization penalty term.

Step 1: Expand the Objective Function

First, we rewrite the squared Euclidean norms using vector notation ($\|\mathbf{v}\|_2^2 = \mathbf{v}^T \mathbf{v}$).

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

Expand the first term by distributing the transpose:

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

Multiply out the terms:

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

Simplify: The terms $\mathbf{y}^T X \mathbf{w}$ and $\mathbf{w}^T X^T \mathbf{y}$ are scalars (results of dot products) and are transposes of each other. Therefore, they are equal. We combine them:

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

We can also group the quadratic terms involving \mathbf{w} :

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

Step 2: Compute the Gradient

We take the derivative with respect to \mathbf{w} and set it to zero. Using the row-vector convention from your homework:

1. **Constant Term:** $\frac{\partial}{\partial \mathbf{w}}(\mathbf{y}^T \mathbf{y}) = 0$.
2. **Linear Term:** $\frac{\partial}{\partial \mathbf{w}}(-2\mathbf{y}^T X \mathbf{w}) = -2\mathbf{y}^T X$.
3. **Quadratic Term:** $\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^T A \mathbf{w}) = 2\mathbf{w}^T A$ (since $A = X^T X + \lambda I$ is symmetric).

Putting it all together:

$$\frac{\partial J}{\partial \mathbf{w}} = -2\mathbf{y}^T X + 2\mathbf{w}^T (X^T X + \lambda I)$$

Step 3: Solve for \mathbf{w}

Set the gradient to zero to find the minimum:

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

Move the linear term to the other side and divide by 2:

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

Transpose the entire equation to solve for the column vector \mathbf{w} (recall that $(AB)^T = B^T A^T$ and symmetric matrices are their own transpose):

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

Finally, multiply by the inverse of $(X^T X + \lambda I)$:

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

The optimal weights for Ridge Regression are:

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

Intuition: Notice this is very similar to the Ordinary Least Squares (OLS) solution $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$. The addition of λI to $X^T X$ ensures the matrix is always invertible (non-singular), even if features are correlated or $n < d$.

Would you like to move on to the next perspective of Ridge Regression?



(b) *Perspective 2: "Hack" of shifting the Singular Values.* In the previous part, you should have found the optimal \mathbf{w} is given by
 $\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$

If you didn't get this, you could check the previous solutions part.
Let's instead try to be more clever with the X . Recall that U and V are square orthonormal (matrix-preserving) matrices, and Σ is a $n \times d$ matrix with singular values σ_i along the "diagonal". Plug this into the Ridge Regression solution and simplify. What happens to the singular values of $(X^T X + \lambda I)^{-1} X^T$ when $\sigma_i \ll \lambda$? What about when $\sigma_i \gg \lambda$?

- ◆ Based on the problem description in **Question 4(b)**, here is the step-by-step derivation and analysis.

Problem Statement

We are given the Ridge Regression solution:

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

We need to plug in the SVD $X = U\Sigma V^T$ and simplify. Then, we analyze the behavior of the singular values in two extreme cases ($\sigma_i \ll \lambda$ and $\sigma_i \gg \lambda$).

Step 1: Substitute the SVD into $X^T X$

First, substitute $X = U\Sigma V^T$ into the term $X^T X$.

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T$$

Since U is orthonormal, $U^T U = I$.

$$X^T X = V(\Sigma^T \Sigma)V^T$$

Note that $\Sigma^T \Sigma$ is a $d \times d$ diagonal matrix containing the squared singular values σ_i^2 on the diagonal.

Step 2: Simplify the Inverse Term $(X^T X + \lambda I)^{-1}$

We substitute our expression for $X^T X$ into the inverse term:

$$(X^T X + \lambda I)^{-1} = (V(\Sigma^T \Sigma)V^T + \lambda I)^{-1}$$

Crucial Trick: Since V is square ($d \times d$) and orthonormal, $VV^T = I$. We can rewrite the identity matrix I as VIV^T .

$$(V(\Sigma^T \Sigma)V^T + \lambda(VIV^T))^{-1} = (V(\Sigma^T \Sigma + \lambda I)V^T)^{-1}$$

Now we invert the product. Recall that for orthogonal matrices $V^{-1} = V^T$.

$$= (V^T)^{-1}(\Sigma^T \Sigma + \lambda I)^{-1}V^{-1}$$

$$= V(\Sigma^T \Sigma + \lambda I)^{-1}V^T$$

Let $D = (\Sigma^T \Sigma + \lambda I)^{-1}$. This is a diagonal matrix where the i -th diagonal entry is $\frac{1}{\sigma_i^2 + \lambda}$.

Step 3: Combine with $X^T \mathbf{y}$

Now substitute everything back into the full equation for \mathbf{w} . Recall $X^T = V\Sigma^T U^T$.

$$\mathbf{w} = \underbrace{V(\Sigma^T \Sigma + \lambda I)^{-1} V^T}_{\text{Inverse part}} \underbrace{V \Sigma^T U^T \mathbf{y}}_{X^T}$$

Since $V^T V = I$, the middle terms cancel:

$$\mathbf{w} = V \underbrace{(\Sigma^T \Sigma + \lambda I)^{-1} \Sigma^T}_{\text{Scale Matrix}} U^T \mathbf{y}$$

Let's look at the "Scale Matrix" in the middle. It is the product of a diagonal matrix with entries $\frac{1}{\sigma_i^2 + \lambda}$ and a diagonal matrix with entries σ_i . The resulting effective singular values for the Ridge transformation are:

$$s_i = \frac{\sigma_i}{\sigma_i^2 + \lambda}$$

So the final simplified form is:

$$\mathbf{w} = V \tilde{\Sigma} U^T \mathbf{y}$$

Where $\tilde{\Sigma}$ is a diagonal matrix with entries $\frac{\sigma_i}{\sigma_i^2 + \lambda}$.

Step 4: Analyzing the Limits

The question asks what happens to these singular values ($\frac{\sigma_i}{\sigma_i^2 + \lambda}$) in two cases.

Case 1: $\sigma_i \ll \lambda$ (Small singular values / Noise) When σ_i is very small compared to λ , the λ term dominates the denominator.

$$\frac{\sigma_i}{\sigma_i^2 + \lambda} \approx \frac{\sigma_i}{\lambda}$$

- **Interpretation:** The weight is scaled down linearly by λ . As $\sigma_i \rightarrow 0$, the weight goes to 0. This means Ridge Regression aggressively suppresses directions in the data with very low variance (noise), preventing the "exploding weights" problem seen in standard inverse calculations.

Case 2: $\sigma_i \gg \lambda$ (Large singular values / Signal) When σ_i is very large compared to λ , the σ_i^2 term dominates the denominator.

$$\frac{\sigma_i}{\sigma_i^2 + \lambda} \approx \frac{\sigma_i}{\sigma_i^2} = \frac{1}{\sigma_i}$$

- **Interpretation:** This is exactly the behavior of the standard Ordinary Least Squares (OLS) or Moore-Penrose pseudo-inverse solution. For strong signals (high variance directions), Ridge Regression has very little effect and behaves like standard regression.

Summary

The effective singular values become $\frac{\sigma_i}{\sigma_i^2 + \lambda}$.

- If $\sigma_i \ll \lambda$, they decay to zero (suppressing noise).
- If $\sigma_i \gg \lambda$, they approach $1/\sigma_i$ (preserving signal, like OLS).

(c) Perspective 3: Maximum A Posteriori (MAP) estimation. Ridge Regression can be viewed as finding the MAP estimate when we apply a prior for the true vector to the observation parameter \mathbf{W} . In particular, we can think of \mathbf{w} as being drawn from $N(0, I)$ and view the random \mathbf{Y} as being generated using $\mathbf{Y}' = \sqrt{\lambda} \mathbf{W} + \sqrt{\lambda} \mathbf{N}$ where the noise \mathbf{N} is distributed (all entries being independent) as $N(0, I)$. At the vector level, we have $\mathbf{Y} = X \mathbf{W} + \sqrt{\lambda} \mathbf{N}$. Note that the X matrix whose rows are the n different training points are not random.

Show that (1) is the MAP estimate for \mathbf{W} given an observation $\mathbf{Y}' = \mathbf{y}$.



- ◆ Based on the problem description in **Question 4(c)**, here is the step-by-step derivation showing that Ridge Regression is equivalent to Maximum A Posteriori (MAP) estimation.

Problem Statement

We want to show that finding the MAP estimate for weights \mathbf{W} is equivalent to solving the Ridge Regression optimization problem:

$$\operatorname{argmin}_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

Given:

1. **Prior:** $\mathbf{W} \sim \mathcal{N}(\mathbf{0}, I)$.
 2. **Likelihood Model:** $\mathbf{Y} = X\mathbf{W} + \sqrt{\lambda}\mathbf{N}$, where $\mathbf{N} \sim \mathcal{N}(\mathbf{0}, I)$.
-

Step 1: Formulate the Prior Probability $P(\mathbf{w})$

The weights \mathbf{W} follow a standard multivariate normal distribution with mean $\mathbf{0}$ and covariance I .

The probability density function is:

$$P(\mathbf{w}) \propto \exp\left(-\frac{1}{2}(\mathbf{w} - \mathbf{0})^T I^{-1}(\mathbf{w} - \mathbf{0})\right)$$

$$P(\mathbf{w}) \propto \exp\left(-\frac{1}{2}\|\mathbf{w}\|_2^2\right)$$

Step 2: Formulate the Likelihood $P(\mathbf{y}|\mathbf{w})$

The observation model is $\mathbf{Y} = X\mathbf{W} + \boldsymbol{\epsilon}$, where the noise term is $\boldsymbol{\epsilon} = \sqrt{\lambda}\mathbf{N}$.

First, let's determine the distribution of the noise $\boldsymbol{\epsilon}$:

- Since $\mathbf{N} \sim \mathcal{N}(\mathbf{0}, I)$, the mean is $\sqrt{\lambda} \cdot \mathbf{0} = \mathbf{0}$.
- The covariance is $\text{Cov}(\sqrt{\lambda}\mathbf{N}) = (\sqrt{\lambda})^2 \text{Cov}(\mathbf{N}) = \lambda I$.

Thus, given \mathbf{w} , the target \mathbf{y} follows a normal distribution:

$$\mathbf{Y}|\mathbf{w} \sim \mathcal{N}(X\mathbf{w}, \lambda I)$$

The likelihood function is:

$$P(\mathbf{y}|\mathbf{w}) \propto \exp\left(-\frac{1}{2}(\mathbf{y} - X\mathbf{w})^T (\lambda I)^{-1}(\mathbf{y} - X\mathbf{w})\right)$$

Since $(\lambda I)^{-1} = \frac{1}{\lambda}I$, we can pull the scalar out:

$$P(\mathbf{y}|\mathbf{w}) \propto \exp\left(-\frac{1}{2\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2\right)$$

Step 3: Apply Bayes' Rule for the Posterior

The MAP estimate maximizes the posterior probability $P(\mathbf{w}|\mathbf{y})$. Using Bayes' rule:

$$P(\mathbf{w}|\mathbf{y}) \propto P(\mathbf{y}|\mathbf{w})P(\mathbf{w})$$

Substitute the expressions from Steps 1 and 2:

$$P(\mathbf{w}|\mathbf{y}) \propto \exp\left(-\frac{1}{2\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2\right) \cdot \exp\left(-\frac{1}{2}\|\mathbf{w}\|_2^2\right)$$

Combine the exponents:

$$P(\mathbf{w}|\mathbf{y}) \propto \exp\left(-\left[\frac{1}{2\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2 + \frac{1}{2}\|\mathbf{w}\|_2^2\right]\right)$$

Step 4: Maximize the Posterior

To find \mathbf{w}_{MAP} , we maximize the posterior probability. Maximizing $e^{-f(x)}$ is equivalent to minimizing $f(x)$ (minimizing the negative log-posterior).

$$\int \left(\frac{1}{\|\mathbf{y} - X\mathbf{w}\|_2^2 + \|\mathbf{w}\|_2^2} \right)$$

$$\mathbf{w}_{MAP} = \operatorname{argmin}_{\mathbf{w}} \left(\frac{1}{2\lambda} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)$$

To clean up the expression, we multiply the entire objective function by the constant scalar 2λ .
 (Note: Multiplying an objective function by a positive constant does not change the location of the minimum).

$$\text{Objective} \cdot (2\lambda) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

Conclusion

The resulting optimization problem is:

$$\mathbf{w}_{MAP} = \operatorname{argmin}_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

This is exactly the **Ridge Regression** objective function defined in Equation (1).

Intuition:

- The **Likelihood** term ($\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$) corresponds to the "data fit" (Least Squares).
- The **Prior** term ($\lambda \|\mathbf{w}\|_2^2$) corresponds to the regularization penalty. A Gaussian prior centered at zero implies we believe the weights are likely to be small, which justifies the L_2 penalty.



(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 $\hat{\mathbf{y}}$ and data matrix $\hat{\mathbf{X}}$:

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

where $\mathbf{0}_d$ is the zero vector in \mathbb{R}^d and $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ is the identity matrix. Show that the classical OLS optimization problem $\operatorname{argmin}_{\mathbf{w}} \|\hat{\mathbf{y}} - \hat{\mathbf{X}}\mathbf{w}\|_2^2$ has the same minimizer as (1).

- ◆ Based on the problem description in **Question 4(d)**, here is the step-by-step verification that Ridge Regression is equivalent to OLS on augmented data.

Problem Statement

We want to show that minimizing the OLS objective on the augmented dataset $(\hat{\mathbf{X}}, \hat{\mathbf{y}})$ yields the same optimal weights \mathbf{w} as the Ridge Regression objective.

Augmented Data:

- $\hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} \in \mathbb{R}^{n+d}$
- $\hat{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda} \mathbf{I}_d \end{bmatrix} \in \mathbb{R}^{(n+d) \times d}$

OLS Objective on Augmented Data:

$$J_{aug}(\mathbf{w}) = \|\hat{\mathbf{y}} - \hat{\mathbf{X}}\mathbf{w}\|_2^2$$

Step 1: Expand the Norm Term

Let's substitute the block definitions of $\hat{\mathbf{y}}$ and $\hat{\mathbf{X}}$ into the objective function.

$$J_{aug}(\mathbf{w}) = \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda} \mathbf{I}_d \end{bmatrix} \mathbf{w} \right\|_2^2$$

Perform the matrix-vector multiplication inside the norm:

$$\begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda} \mathbf{I}_d \end{bmatrix} \mathbf{w} = \begin{bmatrix} \mathbf{X}\mathbf{w} \\ \sqrt{\lambda} \mathbf{I}_d \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{X}\mathbf{w} \\ \sqrt{\lambda} \mathbf{w} \end{bmatrix}$$

Substitute this back into the norm expression:

$$J_{aug}(\mathbf{w}) = \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} - \begin{bmatrix} \mathbf{X}\mathbf{w} \\ \sqrt{\lambda} \mathbf{w} \end{bmatrix} \right\|_2^2$$

$$J_{aug}(\mathbf{w}) = \left\| \begin{bmatrix} \mathbf{y} - X\mathbf{w} \\ \mathbf{0}_d - \sqrt{\lambda}\mathbf{w} \end{bmatrix} \right\|_2^2$$

$$J_{aug}(\mathbf{w}) = \left\| \begin{bmatrix} \mathbf{y} - X\mathbf{w} \\ -\sqrt{\lambda}\mathbf{w} \end{bmatrix} \right\|_2^2$$

Step 2: Decompose the Squared Norm

The squared Euclidean norm of a stacked vector $\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}$ is simply the sum of the squared norms of its parts: $\|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2$.

Applying this property here:

$$J_{aug}(\mathbf{w}) = \|\mathbf{y} - X\mathbf{w}\|_2^2 + \|-\sqrt{\lambda}\mathbf{w}\|_2^2$$

Simplify the second term. The negative sign disappears in the square, and the scalar $\sqrt{\lambda}$ becomes λ :

$$\|-\sqrt{\lambda}\mathbf{w}\|_2^2 = (-\sqrt{\lambda}\mathbf{w})^T(-\sqrt{\lambda}\mathbf{w}) = \lambda\mathbf{w}^T\mathbf{w} = \lambda\|\mathbf{w}\|_2^2$$

Step 3: Final Comparison

Substitute the simplified term back into the equation:

$$J_{aug}(\mathbf{w}) = \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

This expression is identical to Equation (1), which is the original **Ridge Regression optimization problem**.

Conclusion: Minimizing the error on this "fake" augmented dataset (which consists of the real data plus d "virtual" data points that encourage the weights to be zero) is mathematically identical to solving Ridge Regression. This provides a useful implementation trick: you can solve Ridge Regression using any standard OLS solver just by padding your data matrices.

Would you like to continue to the next problem?

(e) *Perspective 5: Fake Features.* For this last interpretation, let's instead construct an augmented design matrix in the following way:
 $\tilde{\mathbf{X}} = [X \quad \sqrt{\lambda}I_n]$.
 i.e., we stack X with $\sqrt{\lambda}I_n$. Interestingly, since our problem is underdetermined, the new dimension $d = n$ is larger than the number of points n . Therefore, there are infinitely many values $\eta \in \mathbb{R}^{d \times 1}$ for which $\tilde{\mathbf{X}}\eta = \mathbf{y}$. We are interested in the **min-norm solution**, i.e., the solution to

$$\underset{\eta}{\operatorname{argmin}} \|\eta\|_2^2 \text{ s.t. } \tilde{\mathbf{X}}\eta = \mathbf{y}. \quad (2)$$

Show that this is yet another form of ridge regression and that the first d coordinates of η^* form the minimizer of (1).

- ◆ Based on the problem description in **Question 4(e)**, here is the step-by-step derivation.

Problem Statement

We are given an augmented design matrix $\tilde{\mathbf{X}} = [X \quad \sqrt{\lambda}I_n]$ and an underdetermined system $\tilde{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}$. We need to find the **minimum norm solution**:

$$\boldsymbol{\eta}^* = \underset{\boldsymbol{\eta}}{\operatorname{argmin}} \|\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \tilde{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}$$

And show that the first d coordinates of $\boldsymbol{\eta}^*$ correspond to the Ridge Regression solution.

Step 1: Apply the Min-Norm Solution Formula

From **Question 3(d)**, we know that for a "fat" matrix A (underdetermined system $A\mathbf{x} = \mathbf{b}$), the minimum norm solution is:

$$\mathbf{x}^* = A^T(AA^T)^{-1}\mathbf{b}$$

In this problem:

- $A = \check{X} = [X \quad \sqrt{\lambda}I_n]$
- $\mathbf{x} = \boldsymbol{\eta}$
- $\mathbf{b} = \mathbf{y}$

So, our solution is:

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

Step 2: Compute the Inner Term ($\check{X}\check{X}^T$)

Let's compute the product $\check{X}\check{X}^T$:

$$\check{X}\check{X}^T = [X \quad \sqrt{\lambda}I_n] \begin{bmatrix} X^T \\ \sqrt{\lambda}I_n^T \end{bmatrix}$$

Perform the block matrix multiplication:

$$\check{X}\check{X}^T = XX^T + (\sqrt{\lambda}I_n)(\sqrt{\lambda}I_n)$$

$$\check{X}\check{X}^T = XX^T + \lambda I_n$$

So, the inverse term is $(XX^T + \lambda I_n)^{-1}$.

Step 3: Solve for $\boldsymbol{\eta}^*$

Now substitute the inverse term and \check{X}^T back into the solution equation. Recall that $\check{X}^T = \begin{bmatrix} X^T \\ \sqrt{\lambda}I_n \end{bmatrix}$.

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

We perform the multiplication to separate the vector $\boldsymbol{\eta}^*$ into its top d components and bottom n components:

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

The problem asks about the **first d coordinates**. Let's call this vector \mathbf{w}_{dual} :

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

Step 4: Verify Equivalence to Ridge Regression

We need to show that this \mathbf{w}_{dual} is equal to the standard Ridge solution derived in 4(a):

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

We use the famous **Push-Through Identity** (a specific case of the Matrix Inversion Lemma), which states that for any matrices appropriate for multiplication:

$$(X^T X + \lambda I)^{-1} X^T = X^T (X X^T + \lambda I)^{-1}$$

Proof of Identity:

1. Start with the identity: $X^T(X X^T + \lambda I_n) = (X^T X + \lambda I_d)X^T$. (Both sides expand to $X^T X X^T + \lambda X^T$).
2. Left-multiply by $(X^T X + \lambda I_d)^{-1}$:

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

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

3. Right-multiply by $(XX^T + \lambda I_n)^{-1}$:

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

Conclusion: Since the identity holds, the top d coordinates of $\boldsymbol{\eta}^*$ are indeed exactly the Ridge Regression weights:

$$\mathbf{w}_{dual} = \mathbf{w}_{ridge}$$

Intuitive Check (Alternative Interpretation)

You can also see this by substituting the constraint into the objective. Let $\boldsymbol{\eta} = [\mathbf{w}^T, \mathbf{v}^T]^T$. Minimizing $\|\boldsymbol{\eta}\|^2 = \|\mathbf{w}\|^2 + \|\mathbf{v}\|^2$. Constraint: $\check{X}\boldsymbol{\eta} = X\mathbf{w} + \sqrt{\lambda}\mathbf{v} = \mathbf{y} \implies \mathbf{v} = \frac{1}{\sqrt{\lambda}}(\mathbf{y} - X\mathbf{w})$. Substitute \mathbf{v} into the objective:

$$\begin{aligned} \min \|\mathbf{w}\|^2 + \left\| \frac{1}{\sqrt{\lambda}}(\mathbf{y} - X\mathbf{w}) \right\|^2 \\ \min \|\mathbf{w}\|^2 + \frac{1}{\lambda} \|\mathbf{y} - X\mathbf{w}\|^2 \end{aligned}$$

Multiplying by λ , this recovers the standard Ridge objective: $\lambda \|\mathbf{w}\|^2 + \|\mathbf{y} - X\mathbf{w}\|^2$.

(1) We know that the Moore-Penrose pseudo-inverse for an underdetermined system (wide matrix) is given by $A^\dagger = A^T (AA^T)^{-1}$, which corresponds to the min-norm solution for $A\mathbf{q} = \mathbf{z}$. That is, the optimization problem

$$\arg\min \|\mathbf{q}\|^2 \text{ s.t. } A\mathbf{q} = \mathbf{z}$$

is solved by $\mathbf{q} = A^\dagger \mathbf{z}$. Let $\hat{\mathbf{w}}$ be the minimizer of (1).
Use the pseudo-inverse to show that solving to the optimization problem in (2) yields

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

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

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

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

- ◆ Based on the problem statement in Question [] there is the step-by-step derivation connecting the "Dual Form" of Ridge Regression to the standard form.

Problem Statement

We established in part (e) that the Ridge Regression solution corresponds to the first d coordinates of the minimum-norm solution to the underdetermined system $\check{X}\boldsymbol{\eta} = \mathbf{y}$, where $\check{X} = [X \quad \sqrt{\lambda}I_n]$.

We need to:

1. Derive the expression $\hat{\mathbf{w}} = X^T(XX^T + \lambda I)^{-1}\mathbf{y}$ using the pseudo-inverse.
2. Show this is equivalent to the standard Ridge solution $\hat{\mathbf{w}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$.

Part 1: Deriving the Dual Form

1. **Min-Norm Solution Formula** For an underdetermined system $A\mathbf{x} = \mathbf{b}$ (where A has full row rank), the minimum norm solution is given by the right pseudo-inverse:

$$\mathbf{x}^* = A^T (AA^T)^{-1} \mathbf{b}$$

In our specific problem:

- $A = \check{X} = [X \quad \sqrt{\lambda}I_n]$
- $\mathbf{x} = \boldsymbol{\eta}$
- $\mathbf{b} = \mathbf{y}$

2. **Compute AA^T** Substitute the block definition of \check{X} :

$$AA^T = \check{X}\check{X}^T = \begin{bmatrix} X & \sqrt{\lambda}I_n \end{bmatrix} \begin{bmatrix} X^T \\ \sqrt{\lambda}I_n \end{bmatrix}$$

$$AA^T = XX^T + (\sqrt{\lambda}I_n)(\sqrt{\lambda}I_n)$$

$$AA^T = XX^T + \lambda I_n$$

3. Substitute into the Solution Now plug this back into the expression for $\boldsymbol{\eta}^*$. Note that $A^T = \check{X}^T = \begin{bmatrix} X^T \\ \sqrt{\lambda}I_n \end{bmatrix}$.

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

Performing the matrix multiplication, we split the result into the top d coordinates (corresponding to \mathbf{w}) and the bottom n coordinates:

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

Thus, the weights are:

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

Part 2: Proving Equivalence (The Push-Through Identity)

We must now show that the expression derived above is identical to the standard Ridge Regression formula:

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

To prove this, we verify the matrix identity:

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

Proof Steps:

- Start with the identity equation we want to prove.

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

- Multiply from the **left** by $(X^T X + \lambda I_d)$ and from the **right** by $(XX^T + \lambda I_n)$ to remove the inverses.

$$(X^T X + \lambda I_d)X^T \stackrel{?}{=} X^T(XX^T + \lambda I_n)$$

- Expand both sides.

- Left Side:** $(X^T X + \lambda I_d)X^T = X^T XX^T + \lambda X^T$
- Right Side:** $X^T(XX^T + \lambda I_n) = X^T XX^T + \lambda X^T$

- Conclusion:** Since the expanded sides are identical, the original identity holds. Thus:

$$\hat{\mathbf{w}}_{dual} = \hat{\mathbf{w}}_{standard}$$

Why does this matter? (Expert Intuition)

This equivalence is the mathematical foundation of the **Kernel Trick**.

- Standard Form:** $(X^T X + \lambda I)^{-1}$ requires inverting a $d \times d$ matrix. If you have few features (d) but many data points (n), this is fast.
- Dual Form:** $(XX^T + \lambda I)^{-1}$ requires inverting an $n \times n$ matrix (the Gram matrix). If you have huge dimensions ($d \gg n$, e.g., in genomics or image processing) but fewer samples, this form is much computationally cheaper.
- Furthermore, in the Dual Form, we only ever see dot products between data points (XX^T), allowing us to replace dot products with Kernel functions $k(\mathbf{x}_i, \mathbf{x}_j)$ to solve non-linear regression efficiently.

Q) We know that the solution to ridge regression (1) is given by $\hat{\mathbf{w}}_R = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$. What happens when $\lambda \rightarrow \infty$? It is for this reason that sometimes ridge regularization is referred to as "shrinkage."
Solution:
As $\lambda \rightarrow \infty$, the matrix $(X^T X + \lambda I)^{-1}$ converges to the zero matrix, and so we have $\mathbf{w} = \mathbf{0}$.

- ◆ Based on the image provided for **Question 4(g)**, here is the detailed explanation of the solution.

Although the image includes the final answer, it is helpful to understand *why* the math behaves this way from both an algebraic and a geometric perspective.

Problem Statement

We want to determine the behavior of the Ridge Regression weights as the regularization parameter λ approaches infinity:

$$\lim_{\lambda \rightarrow \infty} \hat{\mathbf{w}}_{ridge} = \lim_{\lambda \rightarrow \infty} (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

1. Algebraic Explanation

Let's analyze the term inside the inverse: $A = X^T X + \lambda I$. As λ becomes extremely large (much larger than the entries in $X^T X$), the matrix is dominated by the identity term:

$$X^T X + \lambda I \approx \lambda I$$

Now, take the inverse of this approximation:

$$(X^T X + \lambda I)^{-1} \approx (\lambda I)^{-1} = \frac{1}{\lambda} I$$

Substitute this back into the weight equation:

$$\hat{\mathbf{w}} \approx \left(\frac{1}{\lambda} I \right) X^T \mathbf{y} = \frac{1}{\lambda} (X^T \mathbf{y})$$

Now, take the limit as $\lambda \rightarrow \infty$: Since $X^T \mathbf{y}$ is just a constant vector, dividing it by an infinitely large number λ drives every element to zero.

$$\lim_{\lambda \rightarrow \infty} \frac{1}{\lambda} (X^T \mathbf{y}) = \mathbf{0}$$

Conclusion: The weights shrink all the way to the zero vector.

2. The SVD Perspective (From Part 4b)

In Question 4(b), you derived that Ridge Regression scales the singular values of the data by a factor of:

$$s_i = \frac{\sigma_i}{\sigma_i^2 + \lambda}$$

If we let $\lambda \rightarrow \infty$ in this expression:

$$\lim_{\lambda \rightarrow \infty} \frac{\sigma_i}{\sigma_i^2 + \lambda} = 0$$

Since the singular values (the "gain" of the transformation) go to zero, the resulting weights \mathbf{w} must also go to zero.

3. Intuitive "Cost" Explanation

Recall the Ridge Regression objective function:

$$J(\mathbf{w}) = \underbrace{\|\mathbf{y} - X\mathbf{w}\|_2^2}_{\text{Error}} + \underbrace{\lambda \|\mathbf{w}\|_2^2}_{\text{Penalty}}$$

- The parameter λ represents the "price" you pay for having non-zero weights.

The parameter λ represents the "price" you pay for having non-zero weights.

- If $\lambda \rightarrow \infty$, the cost of having *any* non-zero weight becomes infinitely high.
- To minimize the total loss $J(\mathbf{w})$, the optimization algorithm is forced to set $\|\mathbf{w}\| \approx 0$ to avoid an infinite penalty, even if that means the error term $\|\mathbf{y} - X\mathbf{w}\|^2$ (the fit to the data) becomes bad.

This confirms why Ridge Regression is often called **shrinkage**: as you increase λ , you "shrink" the weights toward zero.

(b) What happens to the solution of ridge regression when you take the limit $\lambda \rightarrow 0$? Consider both the cases when X is wide (underdetermined system) and X is tall (overdetermined system).

- ◆ Based on the problem description in **Question 4(h)**, here is the step-by-step analysis of the limit $\lambda \rightarrow 0$.

Problem Statement

We want to analyze the behavior of the Ridge Regression solution $\hat{\mathbf{w}}_{ridge}$ as the regularization strength λ approaches 0.

$$\hat{\mathbf{w}}_{ridge} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

We need to consider two cases:

1. **Overdetermined System** (X is "tall," $n > d$, full column rank).
2. **Underdetermined System** (X is "wide," $d > n$, full row rank).

Case 1: The Tall Matrix (Overdetermined, $n > d$)

Mathematical Derivation: Start with the standard Ridge formula derived in 4(a):

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

As $\lambda \rightarrow 0$, the regularization term λI vanishes.

$$\lim_{\lambda \rightarrow 0} (X^T X + \lambda I)^{-1} X^T \mathbf{y} = (X^T X)^{-1} X^T \mathbf{y}$$

Interpretation: Assuming X has full column rank (which is typical for a "tall" matrix in these problems), the matrix $X^T X$ is invertible. The result $(X^T X)^{-1} X^T \mathbf{y}$ is exactly the standard **Ordinary Least Squares (OLS)** solution derived in Question 3(a).

- **Result:** The solution converges to the **Least Squares solution**.

Case 2: The Wide Matrix (Underdetermined, $d > n$)

Mathematical Derivation: In this case, $X^T X$ is a $d \times d$ matrix. However, since $d > n$, the maximum rank of X is n . This means $X^T X$ is rank-deficient (singular) and **not invertible**. We cannot simply remove λ from the standard formula because $(X^T X)^{-1}$ does not exist.

Instead, we use the **Dual Form** (or Kernel Form) derived in Question 4(f):

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

Since X is "wide" and typically has full row rank (n), the matrix $X X^T$ is an $n \times n$ invertible matrix. We can safely take the limit here:

$$\lim_{\lambda \rightarrow 0} X^T (X X^T + \lambda I)^{-1} \mathbf{y} = X^T (X X^T)^{-1} \mathbf{y}$$

Interpretation: This expression, $X^T (X X^T)^{-1} \mathbf{y}$, is exactly the **Minimum Norm Solution** derived in Question 3(d).

- **Result:** The solution converges to the **Minimum Norm solution**.

Intuitive Explanation (why different results?)

Recall the Ridge objective function:

$$J(\mathbf{w}) = \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

As $\lambda \rightarrow 0$, the "cost" of the weights ($\|\mathbf{w}\|^2$) becomes negligible compared to the error term. The algorithm prioritizes driving the error $\|\mathbf{y} - X\mathbf{w}\|^2$ to zero (or as low as possible).

1. **Overdetermined:** We typically *cannot* get zero error (no solution exists for $X\mathbf{w} = \mathbf{y}$). The algorithm settles for the solution that minimizes the error squared → **Least Squares**.
2. **Underdetermined:** We can find *infinitely* many solutions with zero error. Since the error is zero for all of them, the optimization uses the tiny remaining regularization term $\lambda \|\mathbf{w}\|^2$ as a "tie-breaker." It picks the solution among them with the smallest norm → **Minimum Norm**.

Unified Answer

In both cases, taking the limit $\lambda \rightarrow 0$ results in the solution given by the **Moore-Penrose Pseudoinverse** (X^\dagger):

$$\lim_{\lambda \rightarrow 0} \hat{\mathbf{w}}_{\text{ridge}} = X^\dagger \mathbf{y}$$

5. ReLU Elbow Update under SGD

In this question we will explore the behavior of the ReLU mediante with Stochastic Gradient Descent (SGD) update. 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

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

where $x, y \in \mathbb{R}$, $\mathbf{b} \in \mathbb{R}^d$, $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times 1}$, and $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times d}$. We define our loss function to be the squared error,

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

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,

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

Notice that the slope of $\phi(x)$ is w in the non-zero domain.

We define a loss function $\ell(x, y, \phi) = \frac{1}{2} \|\phi(x) - y\|^2$. Find the following:

- (i) The location of the 'elbow' of the function, where it transitions from 0 to something else.
- (ii) The derivative of the loss w.r.t. $\phi(x)$, namely $\frac{d\ell}{d\phi}$.
- (iii) The partial derivative of the loss w.r.t. w , namely $\frac{\partial \ell}{\partial w}$.
- (iv) The partial derivative of the loss w.r.t. b , namely $\frac{\partial \ell}{\partial b}$.

- ◆ Based on the problem statement in **Question 5(a)** from the uploaded image, here is the step-by-step solution.

This problem asks us to analyze the behavior of a single ReLU unit $\phi(x) = \max(0, wx + b)$ and compute gradients for a simple squared error loss.

(i) The location of the 'elbow' e

Definition: The "elbow" (or kink) of a ReLU function occurs where the input to the non-linearity is exactly zero. This is the point where the function transitions from being "inactive" (output 0) to "active" (linear output).

Derivation: We set the input of the ReLU to zero:

$$wx + b = 0$$

Solving for x :

$$wx = -b$$

$$x = -\frac{b}{w}$$

Answer: The elbow is located at $e = -\frac{b}{w}$.

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

Problem: Calculate $\frac{d\ell}{d\phi}$. **Loss Function:** $\ell(x, y, \phi) = \frac{1}{2} \|\phi(x) - y\|^2$. Since the output is 1D (scalar), this simplifies to $\frac{1}{2} (\phi(x) - y)^2$.

Derivation: Using the chain rule (or simple power rule) on the term $\frac{1}{2} (\phi(x) - y)^2$:

$$\frac{d\ell}{d\phi} = 2 \cdot \frac{1}{2}(\phi(x) - y) \cdot (1)$$

Answer:

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

(iii) The partial derivative of the loss w.r.t. w

Problem: Calculate $\frac{\partial \ell}{\partial w}$. **Chain Rule:**

$$\frac{\partial \ell}{\partial w} = \frac{d\ell}{d\phi} \cdot \frac{\partial \phi}{\partial w}$$

We already know $\frac{d\ell}{d\phi} = (\phi(x) - y)$. Now we need $\frac{\partial \phi}{\partial w}$. Recall:

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

Taking the derivative w.r.t. w :

- **Case 1 ($wx + b > 0$):** The derivative of $(wx + b)$ w.r.t w is x .
- **Case 2 ($wx + b \leq 0$):** The derivative of 0 is 0. (Note: The problem states to define the gradient at 0 as 0).

We can write this compactly using the indicator function $\mathbb{I}(\text{condition})$, which is 1 if true and 0 if false.

$$\frac{\partial \phi}{\partial w} = x \cdot \mathbb{I}(wx + b > 0)$$

Combining terms: **Answer:**

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

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

Problem: Calculate $\frac{\partial \ell}{\partial b}$. **Chain Rule:**

$$\frac{\partial \ell}{\partial b} = \frac{d\ell}{d\phi} \cdot \frac{\partial \phi}{\partial b}$$

Now we need $\frac{\partial \phi}{\partial b}$.

- **Case 1 ($wx + b > 0$):** The derivative of $(wx + b)$ w.r.t b is 1.
- **Case 2 ($wx + b \leq 0$):** The derivative is 0.

Using indicator notation:

$$\frac{\partial \phi}{\partial b} = 1 \cdot \mathbb{I}(wx + b > 0)$$

Combining terms: **Answer:**

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



(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 in light and are trying to pull the function downward. Describe what happens 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

- ◆ Based on the problem description in **Question 5(b)**, here is the step-by-step analysis of how the ReLU parameters update.

Problem Setup

We are given a specific training point (x, y) where the prediction is too high:

$$\phi(x) - y = 1$$

We perform a gradient descent update:

$$w' = w - \lambda \frac{\partial \ell}{\partial w}$$

$$b' = b - \lambda \frac{\partial \ell}{\partial b}$$

From part (a), we know the gradients when $\phi(x) - y = 1$:

- $\frac{\partial \ell}{\partial w} = 1 \cdot x = x$ (if active)
- $\frac{\partial \ell}{\partial b} = 1 \cdot 1 = 1$ (if active)

So the update rules are:

- $w' = w - \lambda x$
- $b' = b - \lambda$

Let's analyze the four cases.

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

- **Condition:** The unit is inactive (dead). The input to the ReLU $wx + b \leq 0$.
- **Gradients:** The gradient of the ReLU in the inactive region is 0.
- **Updates:**

$$w' = w - 0 = w$$

$$b' = b - 0 = b$$

- **Result: Nothing happens.** The slope and elbow location remain exactly the same. The gradient vanishes, so the model cannot "learn" from this point.

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

- **Updates:**
 - $w' = w - \lambda x$. Since $x > 0$, the slope **decreases** (becomes flatter).
 - $b' = b - \lambda$. The intercept decreases.
- **Elbow Behavior ($e = -b/w$):**
 - Let's check numerically as suggested.
 - **Start:** Let $w = 1, b = 0$. Elbow $e = -0/1 = 0$. Let $x = 2$. (Condition satisfied: $1(2) + 0 > 0$.)
 - **Update:** Let $\lambda = 0.1$.
 - $w' = 1 - 0.1(2) = 0.8$.
 - $b' = 0 - 0.1(1) = -0.1$.
 - **New Elbow:** $e' = -(-0.1)/0.8 = 0.125$.
 - **Comparison:** The elbow moved from 0 to 0.125. Since $x = 2$, the elbow moved **to the right (towards x)**.

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

- **Updates:**

- $w' = w - \lambda x$. Since $x < 0$, we are subtracting a negative number. The slope **increases** (becomes steeper).
- $b' = b - \lambda$. The intercept decreases.

- **Elbow Behavior:**

- **Start:** Let $w = 1, x = -1$. For $\phi(x) > 0$, we need $b > 1$. Let $b = 2$. Elbow $e = -2/1 = -2$.
- **Update:** Let $\lambda = 0.1$.
 - $w' = 1 - 0.1(-1) = 1.1$.
 - $b' = 2 - 0.1(1) = 1.9$.
- **New Elbow:** $e' = -1.9/1.1 \approx -1.73$.
- **Comparison:** The elbow moved from -2 to -1.73 . Since $x = -1$, the elbow moved **to the right (towards x)**.

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

- **Updates:**

- $w' = w - \lambda x$. Since w is negative and x is positive, w becomes **more negative** (slope becomes steeper downwards).
- $b' = b - \lambda$. The intercept decreases.

- **Elbow Behavior:**

- **Start:** Let $w = -1, x = 1$. For $\phi(x) > 0$, we need $b > 1$. Let $b = 2$. Elbow $e = -2/-1 = 2$. Note that for negative slope, the active region is to the left of the elbow ($x < e$). Here $1 < 2$.
- **Update:** Let $\lambda = 0.1$.
 - $w' = -1 - 0.1(1) = -1.1$.
 - $b' = 2 - 0.1(1) = 1.9$.
- **New Elbow:** $e' = -1.9/-1.1 \approx 1.73$.
- **Comparison:** The elbow moved from 2 to 1.73 . Since $x = 1$, the elbow moved **to the left (towards x)**.

Summary of Qualitative Changes

In all active cases where the model is predicting "too high":

1. **Slope:** Adjusts to flatten the line (or steepen it negatively) to reduce the value at x .
2. **Elbow:** Always shifts **towards the data point x** .

Drawing Description

Since I cannot draw directly on the screen, here is how you should label your diagram for **Case (ii)**:

1. **Draw Axes:** Standard x and y axes.
2. **Plot Point:** Mark a point (x, y) somewhere in the first quadrant.
3. **Old Function (ϕ):** Draw a line starting from the origin ($e = 0$) passing **above** your point (x, y) . Label this "Old $\phi(x)$ ".
4. **New Function (ϕ'):**
 - Draw the new elbow slightly to the **right** of the origin on the x -axis. Label this "New Elbow".
 - Draw the line extending from this new elbow with a **flatter slope** (lower angle) than the old line.
 - This new line should pass closer to your point (x, y) (reducing the error).

The numerical checks are just examples for you or a student to think through it. Just testing one value doesn't mean the answer will generalize. Try thinking about it with more intuition as to what happens in case (ii). Try out other values of lambda if you want

- ◆ That is an excellent point. Relying on a single numerical example can hide the nuances of the dynamics. Let's step back and look at the **geometry** of the update to build a general intuition that holds for any value of λ .

The "Pivot Point" Intuition

The most powerful way to understand Gradient Descent on a linear layer (which is what the active part of a ReLU is) is to realize that **every update is a rotation of the line around a fixed "Pivot Point"**.

Let's find this point. We are updating the line $y = wx + b$ to $y' = w'x + b'$. Using the update rules for our error (where $\phi(x) - y = 1$):

- $w' = w - \lambda x$
- $b' = b - \lambda$

Let's find the input value x_{pivot} where the output of the function **does not change**, regardless of λ .

$$wx_{pivot} + b = w'x_{pivot} + b'$$

$$wx_{pivot} + b = (w - \lambda x)x_{pivot} + (b - \lambda)$$

$$0 = -\lambda x \cdot x_{pivot} - \lambda$$

$$x \cdot x_{pivot} = -1$$

$$x_{pivot} = -\frac{1}{x}$$

Intuition: The gradient update acts like a lever. To reduce the error at the training point x , the line acts like a seesaw pivoting around $x_{pivot} = -1/x$.

Re-analyzing Case (ii)

- **Conditions:** $w > 0$ (positive slope), $x > 0$ (positive input), $\phi(x) > 0$ (active).
- **The Goal:** The prediction is too high, so we want to **lower** the line at x .
- **The Pivot:** Since $x > 0$, the pivot $x_{pivot} = -1/x$ is **negative**.

Visualizing the Rotation: Imagine a seesaw fulcrum placed at a negative x-value ($-\frac{1}{x}$).

1. Your data point x is far to the right of the fulcrum.
2. You push **down** on the line at x (to reduce the error).
3. Because of the fulcrum, the part of the line to the *left* of the pivot must go **up**.

What happens to the Elbow (e)? The elbow is simply the point where the line crosses the x-axis ($y = 0$). Because we are pushing down on the right side of the pivot:

- Any crossing point (elbow) to the **right** of the pivot will slide further to the **right**.
- Any crossing point (elbow) to the **left** of the pivot will slide further to the **left**.

Conclusion for Case (ii):

- **The Slope:** Always decreases (flattens), because we are pushing down on the positive side of the pivot.

- **The Elbow:**

- **Typical Scenario:** In most models, the bias b is small, so the elbow $e = -b/w$ is close to 0. Since the pivot $-1/x$ is negative, the elbow is to the **right** of the pivot. Therefore, the elbow slides **right (towards x)**.
- **Edge Scenario:** If you have a massive positive bias b such that the elbow is very far negative (specifically $e < -1/x$), the elbow would actually slide **left (away from x)**.

The numerical check in the previous step corresponded to the "Typical Scenario," which is what happens 99% of the time in standard networks.

Revised Diagram Instructions

To capture this "general" intuition in your diagram for Case (ii):

1. **Draw the Pivot:** Mark a point P on the x-axis at $-1/x$ (negative side).
2. **Draw the Old Line:** Draw a straight line passing through P and your elbow e .
3. **Draw the Update:**
 - Show an arrow pushing the line **down** at your data point x .
 - Show the line rotating clockwise around P .
 - This visualizes clearly why the x-intercept (the elbow) slides to the right.



(c) Now we return to the full network function $\hat{f}(x)$. Derive the location e_i of the elbow of the i -th elementwise ReLU activation.



Based on the problem description in **Question 5(c)**, here is the derivation for the location of the elbow for the i -th neuron.

Problem Statement

We are working with the full network function defined in the preamble:

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

We need to find the input value $x = e_i$ that corresponds to the "elbow" (the transition point) of the i -th element of the hidden layer.

Step 1: Identify the Input to the i -th Neuron

Let's look at the argument passed to the activation function ϕ . The term inside is the affine transformation:

$$\mathbf{z} = \mathbf{W}^{(1)}x + \mathbf{b}$$

Since $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times 1}$ and $x \in \mathbb{R}$ is a scalar, the product $\mathbf{W}^{(1)}x$ is a vector where the i -th component is simply the scalar multiplication of the i -th weight and x . Let $W_i^{(1)}$ denote the i -th element of the weight vector $\mathbf{W}^{(1)}$, and b_i denote the i -th element of the bias vector \mathbf{b} .

The input to the i -th ReLU unit is:

$$z_i = W_i^{(1)}x + b_i$$

Step 2: Define the Elbow Condition

The "elbow" of a ReLU activation $\phi(z) = \max(0, z)$ occurs when the input z is exactly **zero**. This is the specific input x where the neuron switches from being "off" (output 0) to "on" (linear output).

Set the i -th input to zero:

$$W_i^{(1)}e_i + b_i = 0$$

Step 3: Solve for e_i

Now, simply solve this linear equation for the specific input value e_i :

$$W_i^{(1)} e_i = -b_i$$

$$e_i = -\frac{b_i}{W_i^{(1)}}$$

Final Answer

The location of the elbow for the i -th elementwise ReLU activation is:

$$e_i = -\frac{b_i}{W_i^{(1)}}$$

(Intuition: This means the full function $\hat{f}(x)$ is a piecewise linear function with "kinks" located at every $x = -b_i/W_i^{(1)}$. These are the points where the slope of the overall network function changes.)

(d) Derive the new elbow location e'_i of the i -th elementwise ReLU activation after one stochastic gradient update with learning rate λ .

- Based on the problem description in Question (d), here is the step-by-step derivation for the new elbow location e'_i .

Problem Statement

We want to find the new location of the elbow e'_i for the i -th neuron after a single Stochastic Gradient Descent (SGD) update with learning rate λ .

Recall from 5(c): The elbow is located at $e_i = -\frac{b_i}{W_i^{(1)}}$. **New Goal:** Compute $e'_i = -\frac{b'_i}{W_i'^{(1)}}$ using the updated parameters b'_i and $W_i'^{(1)}$.

Step 1: Compute the Gradients

First, we calculate the partial derivatives of the loss $\ell = \frac{1}{2}(\hat{f}(x) - y)^2$ with respect to the parameters of the i -th neuron ($W_i^{(1)}$ and b_i).

Let the error term be $\delta = \hat{f}(x) - y$. Let the pre-activation be $z_i = W_i^{(1)}x + b_i$. The activation is $h_i = \phi(z_i)$. The derivative of the ReLU is $\phi'(z_i) = \mathbb{I}(z_i > 0)$.

Gradient w.r.t bias b_i :

$$\frac{\partial \ell}{\partial b_i} = \frac{\partial \ell}{\partial \hat{f}} \cdot \frac{\partial \hat{f}}{\partial h_i} \cdot \frac{\partial h_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial b_i}$$

$$\frac{\partial \ell}{\partial b_i} = \delta \cdot W_i^{(2)} \cdot \mathbb{I}(z_i > 0) \cdot 1$$

Gradient w.r.t weight $W_i^{(1)}$:

$$\frac{\partial \ell}{\partial W_i^{(1)}} = \frac{\partial \ell}{\partial \hat{f}} \cdot \frac{\partial \hat{f}}{\partial h_i} \cdot \frac{\partial h_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial W_i^{(1)}}$$

$$\frac{\partial \ell}{\partial W_i^{(1)}} = \delta \cdot W_i^{(2)} \cdot \mathbb{I}(z_i > 0) \cdot x$$

Step 2: Apply the SGD Update Rule

Let $\Delta_i = \lambda \delta W_i^{(2)}$ be the scaled gradient magnitude factor.

Update for b_i :

$$b'_i = b_i - \lambda \frac{\partial \ell}{\partial b_i} = b_i - \Delta_i \mathbb{I}(z_i > 0)$$

Update for $W_i^{(1)}$:

$$W_i'^{(1)} = W_i^{(1)} - \lambda \frac{\partial \ell}{\partial W_i^{(1)}} = W_i^{(1)} - \Delta_i \mathbb{I}(z_i > 0)x$$

Step 3: Derive the New Elbow e'_i

Case 1: The neuron is inactive ($z_i \leq 0$) Here, $\mathbb{I}(z_i > 0) = 0$.

- $b'_i = b_i$
- $W_i'^{(1)} = W_i^{(1)}$
- **Result:** $e'_i = e_i$. (The elbow does not move if the neuron didn't fire).

Case 2: The neuron is active ($z_i > 0$) Here, $\mathbb{I}(z_i > 0) = 1$. The new elbow is defined as:

$$e'_i = -\frac{b'_i}{W_i'^{(1)}}$$

Substitute the updated parameters:

$$e'_i = -\frac{b_i - \Delta_i}{W_i^{(1)} - \Delta_i x}$$

We can clean this up by distributing the negative sign into the numerator:

$$e'_i = \frac{\Delta_i - b_i}{W_i^{(1)} - \Delta_i x}$$

Alternative Form (using previous elbow e_i): Since $e_i = -b_i/W_i^{(1)}$, we know that $b_i = -W_i^{(1)}e_i$. Substituting this in:

$$e'_i = \frac{\Delta_i - (-W_i^{(1)}e_i)}{W_i^{(1)} - \Delta_i x} = \frac{W_i^{(1)}e_i + \Delta_i}{W_i^{(1)} - \Delta_i x}$$

Final Answer

The new elbow location is:

$$e'_i = \begin{cases} e_i & \text{if } W_i^{(1)}x + b_i \leq 0 \\ \frac{\lambda \delta W_i^{(2)} - b_i}{W_i^{(1)} - \lambda \delta W_i^{(2)} x} & \text{if } W_i^{(1)}x + b_i > 0 \end{cases}$$

Where $\delta = (\hat{f}(x) - y)$ is the prediction error.

You are close. Instead of using terms like delta, try keeping it in terms of the original matrices $W^{(1)}$ and $W^{(2)}$. Also think a little more carefully about the derivative of the loss with respect to $W^{(1)}_{-i}$

- ◆ You are absolutely right to ask for a more careful breakdown using the explicit matrix terms. The relationship between the weight gradient and the bias gradient is the key to simplifying the final expression.

Here is the step-by-step derivation for the new elbow location e'_i using the original parameters.

1. Gradients in terms of Network Parameters

First, let's write out the gradients for the i -th neuron's bias b_i and weight $W_i^{(1)}$. We assume the

neuron is **active** (i.e., $W_i^{(1)}x + b_i > 0$), otherwise the gradients are zero and the elbow doesn't move.

- **Bias Gradient:** $\frac{\partial \ell}{\partial b_i} = (\hat{f}(x) - y)W_i^{(2)}$
 - **Weight Gradient:** $\frac{\partial \ell}{\partial W_i^{(1)}} = (\hat{f}(x) - y)W_i^{(2)}x$

Key Observation: Notice that the weight gradient is just the bias gradient multiplied by the input x :

$$\frac{\partial \ell}{\partial W_i^{(1)}} = x \cdot \frac{\partial \ell}{\partial b_i}$$

2. SGD Update Rules

We update the parameters using the learning rate λ :

- **New Bias:** $b'_i = b_i - \lambda \frac{\partial \ell}{\partial b_i}$
 - **New Weight:** $W_i'^{(1)} = W_i^{(1)} - \lambda \left(x \frac{\partial \ell}{\partial b_i} \right)$

3. Deriving the New Elbow e'_i

The new elbow is defined as the point where the new input to the ReLU is zero:

$$e'_i = -\frac{b'_i}{W_i'(1)}$$

Substitute the update rules into this expression:

$$e'_i = -\frac{b_i - \lambda \frac{\partial \ell}{\partial b_i}}{W_i^{(1)} - \lambda x \frac{\partial \ell}{\partial b_i}}$$

Distribute the negative sign into the numerator:

$$e'_i = \frac{\lambda \frac{\partial \ell}{\partial b_i} - b_i}{W_i^{(1)} - \lambda x \frac{\partial \ell}{\partial b_i}}$$

4. Simplifying into a Clean Form

We can express this in terms of the old elbow e_i . Recall that $b_i = -e_i W_i^{(1)}$. Let's substitute this for the b_i term in the numerator:

$$e'_i = \frac{\lambda \frac{\partial \ell}{\partial b_i} - (-e_i W_i^{(1)})}{W_i^{(1)} - \lambda x \frac{\partial \ell}{\partial b_i}}$$

$$e'_i = \frac{e_i W_i^{(1)} + \lambda \frac{\partial \ell}{\partial b_i}}{W_i^{(1)} - x \left(\lambda \frac{\partial \ell}{\partial b_i} \right)}$$

Now, divide both the numerator and the denominator by $W_i^{(1)}$:

$$e'_i = \frac{e_i + \frac{\lambda}{W_i^{(1)}} \frac{\partial \ell}{\partial b_i}}{1 - x \left(\frac{\lambda}{W_i^{(1)}} \frac{\partial \ell}{\partial b_i} \right)}$$

Finally, substitute the full expression for the bias gradient $\frac{\partial \ell}{\partial b_i} = (\hat{f}(x) - y)W_i^{(2)}$:

Final Answer

$$e'_i = \frac{e_i + \alpha}{1 - \alpha x}$$

where the step size term α is:

$$\alpha = \frac{\lambda(f(x) - y)W_i^{(1)}}{W_i^{(1)}}$$

Intuitive Interpretation: This result shows that the new elbow is a **fractional linear transformation** (also known as a Möbius transformation) of the old elbow, determined by the ratio of the update strength to the original weight magnitude.

[Google Privacy Policy](#) | [Google Terms of Service](#) | [Your privacy and Gemini Apps](#)