

It might be more convenient to upload the entire document. However, uploading the entire document can make the answer very long and each subpart answer too concise. Therefore I upload it by copy-paste from the homework a question at a time.

Vector Calculus Review Let  $x, c \in \mathbb{R}^n$  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 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.  $\frac{\partial f}{\partial x} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}]$  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. Homework 0, © Faculty teaching EECS 182, Fall 2025. All Rights Reserved. This may not be publicly shared without explicit permission. 1 Homework 0 @ 2025-08-27 14:45:28-07:00 (a) Show  $\frac{\partial}{\partial x}(x^T c) = c^T$  (b) Show  $\frac{\partial}{\partial x} \|x\|_2^2 = 2x^T$  (c) Show  $\frac{\partial}{\partial x}(Ax) = A$  (d) Show  $\frac{\partial}{\partial x}(x^T Ax) = x^T(A + A^T)$  (e) Under what condition is the previous derivative equal to  $2x^T A$ ?

3:52pm

 Let's tackle each part of your **Vector Calculus Review** step by step, focusing on the **type of the derivative** (scalar, vector, or matrix), how to compute each term, and then deriving the final result.

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

#### Type of the Derivative

- $x^T c$  is a **scalar** (inner product of two vectors).
- The derivative of a scalar with respect to a vector  $x$  is a **row vector** (by convention).

**Derivation** The model gives me the shape of the derivative, which is helpful for warm-up.

Let  $x = [x_1, x_2, \dots, x_n]^T$  and  $c = [c_1, c_2, \dots, c_n]^T$ .

Then,

$$x^T c = \sum_{i=1}^n x_i c_i$$

Taking the partial derivative with respect to  $x_j$ :

$$\frac{\partial}{\partial x_j}(x^T c) = c_j$$

So, the gradient is:

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

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

Type of the Derivative

- $\|x\|_2^2 = x^T x$  is a scalar.
- The derivative is a **row vector**.

Derivation

$$\|x\|_2^2 = \sum_{i=1}^n x_i^2$$

Taking the partial derivative with respect to  $x_j$ :

$$\frac{\partial}{\partial x_j} \|x\|_2^2 = 2x_j$$

$$\frac{\partial}{\partial x} \|x\|_2^2 = [2x_1, 2x_2, \dots, 2x_n] = 2x^T$$

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

Type of the Derivative

- $Ax$  is a **vector** (size  $n \times 1$ ).
- The derivative of a vector with respect to a vector is a **matrix** (size  $n \times n$ ).

Derivation

Let  $y = Ax$ , so  $y_i = \sum_{j=1}^n A_{ij}x_j$ .

The  $(i, j)$ -th entry of the derivative is:

$$\frac{\partial y_i}{\partial x_j} = A_{ij}$$

So, the derivative is the matrix  $A$ .

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

### Type of the Derivative

- $x^T Ax$  is a **scalar**.
- The derivative is a **row vector**.

### Derivation

$$x^T Ax = \sum_{i=1}^n \sum_{j=1}^n x_i A_{ij} x_j$$

Taking the partial derivative with respect to  $x_k$ :

$$\frac{\partial}{\partial x_k}(x^T Ax) = \sum_{i=1}^n A_{ik} x_i + \sum_{j=1}^n A_{kj} x_j = (Ax)_k + (A^T x)_k$$

So, the gradient is:

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

**(e) Under what condition is the previous derivative equal to  $2x^T A$ ?**

From part (d), we have:

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

This holds if and only if:

$$A + A^T = 2A \implies A^T = A$$

Le Chat answers question 2 correctly.  
It is not surprising since question 2 is basic calculus and  
the conclusion is mostly given in the question.

So, the condition is that  $A$  is symmetric.

### Summary Table

Part	Expression	Type of Derivative	Result	Condition (if any)
(a)	$\frac{\partial}{\partial x}(x^T c)$	Row vector	$c^T$	—
(b)	$\frac{\partial}{\partial x} \ x\ _2^2$	Row vector	$2x^T$	—
(c)	$\frac{\partial}{\partial x}(Ax)$	Matrix	$A$	—
(d)	$\frac{\partial}{\partial x}(x^T Ax)$	Row vector	$x^T(A + A^T)$	—
(e)	$x^T(A + A^T) = 2x^T A$	—	—	$A$ is symmetric

### In summary:

- For (a) and (b), the derivatives are row vectors.
- For (c), the derivative is a matrix.
- For (d), the result is a row vector, and for (e), the condition is that  $A$  is symmetric.

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

Consider the equation  $Xw = y$ , where  $X \in \mathbb{R}^{m \times n}$  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  $X = U\Sigma V^T$  is the (full) SVD of  $X$ . Here,  $U$  and  $V$  are orthonormal square matrices, and  $\Sigma$

is an  $m \times n$  matrix with non-zero singular values ( $\sigma_i$ ) on the "diagonal".

For this problem, we define  $\Sigma^{\dagger}$  an  $n \times m$  matrix with the reciprocals of the singular values ( $1/\sigma_i$ ) 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  $\min_w \|Xw - y\|^2$

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

(c) You'll notice that the least-squares solution is in the form  $w^*$

$= A^{\dagger}y$ . What happens if we left-

multiply  $X$  by our matrix  $A^{\dagger}$ ? This is why the matrix  $A^{\dagger}$  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 minimum-

norm solution, i.e. we want to solve  $\min_w \|w\|^2$  s.t.  $Xw = y$ . What is the minimum norm solution?

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

(f) You'll notice that the min-norm solution is in the form  $w^*$

$= B^{\dagger}y$ . What happens if we right-multiply

$X$  by our matrix  $B^{\dagger}$ ? This is why the matrix  $B^{\dagger}$  of the min-norm solution is called the right-inverse.

3:53pm



Let's break down each part of your **Least Squares and Min-norm Problem from the Perspective of SVD** step by step, focusing on the use of the SVD and the properties of the pseudoinverse.

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

Consider the equation  $Xw = y$ , where  $X \in \mathbb{R}^{m \times n}$  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  $X = U\Sigma V^T$  is the (full) SVD of  $X$ . Here,  $U$  and  $V$  are orthonormal square matrices, and  $\Sigma$

is an  $m \times n$  matrix with non-zero singular values ( $\sigma_i$ ) on the "diagonal".

For this problem, we define  $\Sigma^\dagger$  an  $n \times m$  matrix with the reciprocals of the singular values ( $1/\sigma_i$ ) 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  $\min_w \|Xw - y\|_2^2$

- (b) Plug in the SVD  $X = U\Sigma V^T$  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 left-multiply  $X$  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 minimum-norm solution, i.e. we want to solve  $\min \|w\|_2$  s.t.  $Xw = y$ . What is the minimum norm solution?
- (e) Plug in the SVD  $X = U\Sigma V^T$  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 break down each part of your **Least Squares and Min-norm Problem from the Perspective of SVD** step by step, focusing on the use of the SVD and the properties of the pseudoinverse.

### (a) Least Squares Problem ( $m > n$ , overdetermined system)

#### Goal:

Find  $w$  that minimizes  $\|Xw - y\|_2^2$ .

#### Solution:

The least squares solution is given by the **normal equations**:

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

This is the unique solution when  $X$  has full column rank.

### (b) Plug in SVD and Simplify ( $m > n$ )

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

- $U$  is  $m \times m$  orthonormal,
- $\Sigma$  is  $m \times n$  with singular values  $\sigma_i$  on the diagonal,
- $V$  is  $n \times n$  orthonormal.

Compute  $X^T X$ :

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

Since  $\Sigma^T \Sigma$  is  $n \times n$  diagonal with  $\sigma_i^2$  on the diagonal:

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

The AI clearly gives me the dimension of the matrices, allowing me to track my calculations, which is very helpful for beginners. Also the question cautions me about dimensions. Mistral AI is careful with that.

So,

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

Let  $\Sigma^\dagger$  be the  $n \times m$  matrix with  $1/\sigma_i$  on the diagonal:

$$w^* = V \Sigma^\dagger U^T y$$

### (c) Left-Inverse

The matrix  $A = V \Sigma^\dagger U^T$  is called the **left-inverse** of  $X$  because:

$$AX = V \Sigma^\dagger U^T U \Sigma V^T = V \Sigma^\dagger \Sigma V^T = V I_n V^T = I_n$$

So,  $AX = I_n$ .

### (d) Minimum-Norm Problem ( $m < n$ , underdetermined system)

**Goal:**

Find  $w$  that minimizes  $\|w\|_2^2$  subject to  $Xw = y$ .

**Solution:**

The minimum-norm solution is:

Mistral AI does not give the interpretations as the homework solution do. But the answer is still correct and concise.

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

This is the unique solution with the smallest norm.

### (e) Plug in SVD and Simplify ( $m < n$ )

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

- $U$  is  $m \times m$  orthonormal,
- $\Sigma$  is  $m \times n$  with singular values  $\sigma_i$  on the diagonal,
- $V$  is  $n \times n$  orthonormal.

**Compute  $XX^T$ :**

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

Since  $\Sigma \Sigma^T$  is  $m \times m$  diagonal with  $\sigma_i^2$  on the diagonal:

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

So,

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

Let  $\Sigma^\dagger$  be the  $n \times m$  matrix with  $1/\sigma_i$  on the diagonal:

$$w^* = V\Sigma^\dagger U^T y$$

### (f) Right-Inverse

The matrix  $B = V\Sigma^\dagger U^T$  is called the **right-inverse** of  $X$  because:

$$XB = U\Sigma V^T V\Sigma^\dagger U^T = U\Sigma\Sigma^\dagger U^T = UI_m U^T = I_m$$

So,  $XB = I_m$ .

### Summary Table

Case	Problem Type	Solution $w^*$	Inverse Type	Inverse Property
$m > n$	Least Squares	$V\Sigma^\dagger U^T y$	Left-inverse	$AX = I_n$
$m < n$	Minimum-Norm	$V\Sigma^\dagger U^T y$	Right-inverse	$XB = I_m$

#### In summary:

- For **least squares ( $m > n$ )**, the solution is  $w^* = V\Sigma^\dagger U^T y$ , and  $V\Sigma^\dagger U^T$  is the left-inverse of  $X$ .
- For **minimum-norm ( $m < n$ )**, the solution is  $w^* = V\Sigma^\dagger U^T y$ , and  $V\Sigma^\dagger U^T$  is the right-inverse of  $X$ .
- In both cases, the solution is expressed using the **pseudoinverse**  $X^\dagger = V\Sigma^\dagger U^T$ .

Mistral AI gives very helpful summaries at the end.

In this way I can really gain insight and intuition of pseudoinverse.

I'm not familiar with SVD. Please explain SVD to me.

I am a little bit rusty on SVD when doing HW0.



Absolutely! **Singular Value Decomposition (SVD)** is one of the most powerful and widely used matrix factorizations in linear algebra. It breaks down any real or complex matrix into three simpler matrices, revealing essential properties about the original matrix.

### What is SVD?

Given any matrix  $X \in \mathbb{R}^{m \times n}$ , its **SVD** is:

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

where:

- $U$  is an  $m \times m$  **orthogonal** matrix (its columns are called **left singular vectors**).
- $\Sigma$  is an  $m \times n$  **diagonal** matrix with non-negative real numbers on the diagonal (called **singular values**).
- $V$  is an  $n \times n$  **orthogonal** matrix (its columns are called **right singular vectors**).

## Key Properties

- The singular values  $\sigma_1, \sigma_2, \dots, \sigma_r$  are ordered from largest to smallest.
- The number of non-zero singular values is equal to the **rank** of  $X$ .
- $U$  and  $V$  are orthogonal, so  $U^T U = I$  and  $V^T V = I$ .

## Geometric Interpretation

- $U$ : Rotates/reflects the space to align with the principal directions of  $X$ .
- $\Sigma$ : Scales the space along these principal directions by the singular values.
- $V^T$ : Rotates/reflects the space back to the original coordinate system.

## Visualization

If  $X$  represents a linear transformation, SVD decomposes it into:

1. A rotation/reflection ( $V^T$ ),
2. A scaling ( $\Sigma$ ),
3. Another rotation/reflection ( $U$ ).

## How to Compute SVD

1. **Compute  $X^T X$  and  $XX^T$ :**
  - The eigenvalues of  $X^T X$  are the squares of the singular values of  $X$ .
  - The eigenvectors of  $X^T X$  are the columns of  $V$ .
  - The eigenvectors of  $XX^T$  are the columns of  $U$ .
2. **Construct  $\Sigma$ :**
  - The diagonal entries are the square roots of the eigenvalues of  $X^T X$  (or  $XX^T$ ).

## Why is SVD Useful?

- **Data Compression:** By keeping only the largest singular values, you can approximate  $X$  with a lower-rank matrix.
- **Solving Linear Systems:** SVD is used to solve least squares and minimum-norm problems, as in your previous question.
- **Principal Component Analysis (PCA):** SVD is the mathematical backbone of PCA, a popular dimensionality reduction technique.
- **Image Processing:** SVD is used for image compression and noise reduction.

### Example

After careful computation, I find the solution of this example to be:

I was not patient enough to calculate this previously.

Let's say:

I did not prompt Mistral AI again since this is brute force computation (no need for AI).

$$X = \begin{bmatrix} 3 & 1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix}$$

The SVD of  $X$  might look like:

$$U = \begin{bmatrix} -0.61 & -0.52 & -0.60 \\ -0.35 & 0.85 & -0.39 \\ -0.71 & -0.04 & 0.70 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 4.22 & 0 \\ 0 & 1.20 \\ 0 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} -0.58 & -0.82 \\ -0.82 & 0.58 \end{bmatrix}$$

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

## Summary Table

Matrix	Size	Description
$U$	$m \times m$	Left singular vectors (orthogonal)
$\Sigma$	$m \times n$	Singular values (diagonal)
$V$	$n \times n$	Right singular vectors (orthogonal)

### In summary:

SVD decomposes any matrix into a product of three matrices, revealing its fundamental structure and enabling a wide range of applications in data science, engineering, and beyond.

Would you like to see a step-by-step numerical example, or do you want to explore a specific application of SVD?

⚡ 3:54pm

↶ ↷ ↺

I understood, so no follow-up on SVD. (Comment: next time I should work through a concrete example) Mistral AI gives explanation of FULL SVD, which is just what the homework asks me to perform.

#### 4. The 5 Interpretations of Ridge Regression

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

$\operatorname{argmin}_w$

$w$

$\|y - Xw\|^2$

$2 + \lambda \|w\|^2$

2, (1)

where  $X \in \mathbb{R}^{n \times d}$  is a data matrix, and  $y \in \mathbb{R}^n$  is the target vector of measurement values. What's new

compared to the simple OLS problem is the addition of the  $\lambda \|w\|^2$  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

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

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

Homework 0, © Faculty teaching EECS 182, Fall 2025. All Rights Reserved. This may not be publicly shared without explicit permission. 2

Homework 0 @ 2025-08-27 14:45:28-07:00

Let  $X = U \Sigma V^T$  be the (full) SVD of the  $X$ . Recall that  $U$  and  $V$  are square orthonormal (norm-preserving) matrices, and  $\Sigma$  is a  $n \times d$  matrix with singular values  $\sigma_i$  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  $\sigma_i \gg \lambda$ ?

(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  $Y = XW + \sqrt{\lambda}N$  where the noise  $N$  is distributed iid (across training samples) as  $N(0, 1)$ . At the vector level, we have  $Y = XW + \sqrt{\lambda}N$ . Note that the  $X$  matrix whose rows are the  $n$  different training points are not random.

Show that (1) is the MAP estimate for  $W$  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  $\hat{y}$  and data matrix  $\hat{X}$ :

$\hat{y} =$

$y$

$0_d$

$\hat{X} =$

$X$

$\sqrt{\lambda}I_d$

where  $0_d$  is the zero vector in  $R_d$  and  $I_d \in R_d \times d$  is the identity matrix. Show that the classical OLS

optimization problem  $\operatorname{argmin}_w \| \hat{y} - Xw \|_2^2$

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:

$$X = [X \ \sqrt{\lambda}I_n]$$

i.e. we stack  $X$  with  $\sqrt{\lambda}I_n$  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  $\eta \in R^{d+n}$  for

which

$X\eta = y$ . We are interested in the min-norm solution, ie. the solution to  $\operatorname{argmin}$

$$\eta$$

$$\|\eta\|_2^2$$

s.t.

$$X\eta = y. \quad (2)$$

Show that this is yet another form of ridge regression and that the first coordinates of  $\eta$  form the minimizer of (1).

(f) 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  $A\eta = z$ . That is, the optimization problem

$$\operatorname{argmin} \|\eta\|_2^2 \text{ s.t. } A\eta = z$$

is solved by  $\eta = A^\dagger z$ . Let  $\hat{w}$  be the minimizer of (1).

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

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

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

$$\hat{w} = (XTX + \lambda I)^{-1}XTy$$

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

Homework 0, © Faculty teaching EECS 182, Fall 2025. All Rights Reserved. This may not be publicly shared without explicit permission. 3

Homework 0 @ 2025-08-27 14:45:28-07:00

(g) We know that the solution to ridge regression (1) is given by  $\hat{w} = (XTX + \lambda I)^{-1}XTy$ . What happens when  $\lambda \rightarrow \infty$ ? 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  $\lambda \rightarrow 0$ ? Consider both

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

3:54pm



Here's a clear, step-by-step breakdown of **all five interpretations of Ridge Regression**, including the requested derivations and explanations:

### (a) Optimization Problem

**Problem:**

$$\arg \min_w \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

**Solution:**

Expand the objective:

$$\begin{aligned} \|y - Xw\|_2^2 + \lambda \|w\|_2^2 &= (y - Xw)^T (y - Xw) + \lambda w^T w \\ &= y^T y - 2w^T X^T y + w^T X^T X w + \lambda w^T w \end{aligned}$$

Take the derivative with respect to  $w$ :

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

Set to zero:

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

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

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

## (b) SVD Perspective

Given  $X = U\Sigma V^T$ , plug into the Ridge solution:

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

**Effect of  $\lambda$ :**

- If  $\sigma_i \ll \lambda$ : The term  $\sigma_i^2 + \lambda \approx \lambda$ , so the corresponding component of  $w$  is shrunk toward zero.
- If  $\sigma_i \gg \lambda$ : The term  $\sigma_i^2 + \lambda \approx \sigma_i^2$ , so the component is barely affected.

## (c) MAP Estimation

**Model:**

The model correctly describes the shrinkage effect but doesn't connect it intuitively to the problem of ill-conditioning. It misses the opportunity to explain that  $\lambda$  acts as a stabilizer, preventing the term  $(\sigma_i^2 + \lambda)^{-1}$  from blowing up when  $\sigma_i$  is very small.

$$Y = XW + \sqrt{\lambda}N, \quad N \sim \mathbf{N}(0, I), \quad W \sim \mathbf{N}(0, I)$$

**Posterior:**

$$p(W|Y = y) \propto p(Y = y|W)p(W) = \exp\left(-\frac{1}{2}\|y - XW\|_2^2\right) \exp\left(-\frac{\lambda}{2}\|W\|_2^2\right)$$

Taking the log and maximizing:

$$\arg \max_W -\|y - XW\|_2^2 - \lambda\|W\|_2^2$$

This is equivalent to the Ridge objective (1).

#### (d) Fake Data

**Augmented data:**

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

**OLS solution:**

$$\arg \min_w \|\hat{y} - \hat{X}w\|_2^2 = \arg \min_w \|y - Xw\|_2^2 + \lambda\|w\|_2^2$$

This is identical to the Ridge objective (1).

#### (e) Fake Features

**Augmented design matrix:**

$$\tilde{X} = [X \ \sqrt{\lambda}I]$$

**Min-norm solution:**

$$\arg \min_{\eta} \|\eta\|_2^2 \text{ s.t. } \tilde{X}\eta = y$$

Let  $\eta = \begin{bmatrix} w \\ z \end{bmatrix}$ , so:

$$Xw + \sqrt{\lambda}Iz = y$$

The min-norm solution is:

$$\eta^* = \tilde{X}^T (\tilde{X} \tilde{X}^T)^{-1} y$$

$$\tilde{X} \tilde{X}^T = X X^T + \lambda I$$

$$\eta^* = \left[ \frac{X^T}{\sqrt{\lambda} I} \right] (X X^T + \lambda I)^{-1} y$$

The first  $d$  coordinates of  $\eta^*$  are:

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

Using the Woodbury identity:

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

This matches the Ridge solution.

### (f) Pseudo-Inverse

The min-norm solution is:

$$\eta^* = \tilde{X}^T (\tilde{X} \tilde{X}^T)^{-1} y = \left[ \frac{X^T}{\sqrt{\lambda} I} \right] (X X^T + \lambda I)^{-1} y$$

The first  $d$  coordinates are:

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

Using the Woodbury identity again:

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

This is the standard Ridge formula.

### (g) Limit $\lambda \rightarrow \infty$

As  $\lambda \rightarrow \infty$ :

$$(X^T X + \lambda I)^{-1} X^T y \approx \frac{1}{\lambda} X^T y \rightarrow 0$$

**Interpretation:** The weights "shrink" to zero.

The model's solution is overly terse. It states "Using the Woodbury identity..." without showing the steps. A strong solution would explicitly show the equivalence:  $w = X^T (X X^T + \lambda I)^{-1} y = (X^T X + \lambda I)^{-1} X^T y$ . I understand this identity from my statistics course, but using it directly may defeat the purpose of this hw. Deriving this equivalence is non-trivial and is the key to understanding the connection between the primal and dual forms of Ridge Regression. (The model provides the "what" but often skips the "why" or the "how" for the more challenging steps. It treats non-trivial linear algebra identities as black boxes, which hinders deep understanding.)

One-shot correct answer.

Mistral AI's explanation is not as detailed as the homework solution, but I understand it. That is probably because the question is very long. Which means giving it the entire homework is not a great idea.

### (h) Limit $\lambda \rightarrow 0$

- **Tall  $X$  (overdetermined):** The solution converges to the OLS solution  $(X^T X)^{-1} X^T y$ .
- **Wide  $X$  (underdetermined):** The solution converges to the minimum-norm solution  $X^T (X X^T)^{-1} y$ .

### Summary Table

Perspective	Interpretation	Key Formula/Result
Optimization	Penalized least squares	$w = (X^T X + \lambda I)^{-1} X^T y$
SVD	Singular value shrinkage	$w = V (\Sigma^T \Sigma + \lambda I)^{-1} \Sigma^T U^T y$
MAP	Bayesian with Gaussian prior	Same as Ridge
Fake Data	Augmented OLS	Same as Ridge
Fake Features	Min-norm underdetermined	Same as Ridge
Shrinkage	$\lambda \rightarrow \infty$	$w \rightarrow 0$
No Regularization	$\lambda \rightarrow 0$	OLS or min-norm solution

#### In summary:

Ridge regression can be viewed as penalized least squares, a Bayesian MAP estimate, or as OLS on augmented data or features. The regularization parameter  $\lambda$  controls the trade-off between fit and shrinkage, and its limits reveal the behavior of the solution.

#### 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

$$f(x) = W(2) \Phi W(1)x + b$$

where  $x, y \in \mathbb{R}$ ,  $b \in \mathbb{R}$

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

$$\ell(x, y, W(1))$$

$$, b, W(2) =$$

1

2

^

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,

$\phi(x) = wx + b$ ,  $wx + b > 0$

0, else.

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

We define a loss function  $\ell(x, y, \phi) = 1$

$2 \|\phi(x) - y\|^2$

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  $d\ell$

$d\phi$

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

$\partial w$

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

$\partial b$

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 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.

Homework 0, © Faculty teaching EECS 182, Fall 2025. All Rights Reserved. This may not be publicly shared without explicit permission. 4

Homework 0 @ 2025-08-27 14:45:28-07:00

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  $\ell$  under SGD is

$p'$

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

(c) Now we return to the full network function<sup>^</sup>

$f(x)$ . Derive the location  $e_i$  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  $\lambda$ .

Navigation icons: back, forward, search, etc.



Let's break down each part of your **ReLU Elbow Update under SGD** problem, focusing on the behavior of the ReLU nonlinearity and how SGD updates affect the "elbow" and slope of the function.

## (a) Single ReLU Function

The ReLU function is:

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

- If  $wx + b > 0$ ,  $\phi(x) = wx + b$ .
- If  $wx + b \leq 0$ ,  $\phi(x) = 0$ .

## (b) Loss Function and Gradients

### (i) Location of the 'Elbow' $e$

The elbow is where  $wx + b = 0$ :

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

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

The loss is:

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

So,

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

### (iii) Partial Derivative w.r.t. $w$

Using the chain rule:

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

- If  $\phi(x) > 0$ ,  $\frac{\partial \phi}{\partial w} = x$ .
- If  $\phi(x) = 0$ ,  $\frac{\partial \phi}{\partial w} = 0$ .

So,

$$\frac{\partial \ell}{\partial w} = \begin{cases} (\phi(x) - y) \cdot x & \text{if } \phi(x) > 0 \\ 0 & \text{if } \phi(x) = 0 \end{cases}$$

#### (iv) Partial Derivative w.r.t. $b$

Similarly:

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

- If  $\phi(x) > 0$ ,  $\frac{\partial \phi}{\partial b} = 1$ .
- If  $\phi(x) = 0$ ,  $\frac{\partial \phi}{\partial b} = 0$ .

So,

$$\frac{\partial \ell}{\partial b} = \begin{cases} \phi(x) - y & \text{if } \phi(x) > 0 \\ 0 & \text{if } \phi(x) = 0 \end{cases}$$

#### (b) SGD Update Analysis

Given  $\phi(x) - y = 1$  (prediction is 1 unit above target), let's analyze the update for each case:

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

- No gradient flows through  $w$  or  $b$ .
- **Elbow and slope do not change.**

For part b, Mistral AI does not consider the magnitude of the movement, resulting in neglected scenarios.

##### (ii) $w > 0, x > 0, \phi(x) > 0$

- Update:

$$w' = w - \lambda(\phi(x) - y)x = w - \lambda \cdot 1 \cdot x$$

$$b' = b - \lambda(\phi(x) - y) = b - \lambda \cdot 1$$

- **Elbow moves left** (since  $b$  decreases and  $w$  decreases).
- **Slope decreases** (since  $w$  decreases).

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

- Update:

$$w' = w - \lambda(\phi(x) - y)x = w - \lambda \cdot 1 \cdot x$$

Since  $x < 0$ ,  $w$  **increases**.

$$b' = b - \lambda(\phi(x) - y) = b - \lambda \cdot 1$$

- **Elbow moves left** (since  $b$  decreases).
- **Slope increases** (since  $w$  increases).

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

- Update:

$$w' = w - \lambda(\phi(x) - y)x = w - \lambda \cdot 1 \cdot x$$

Since  $w < 0$  and  $x > 0$ ,  $w$  becomes **less negative** (moves toward zero).

$$b' = b - \lambda(\phi(x) - y) = b - \lambda \cdot 1$$

- **Elbow moves left** (since  $b$  decreases).
- **Slope becomes less negative** (since  $w$  increases).

### (c) Elbow Location for Full Network

For the  $i$ -th ReLU in the hidden layer:

$$\phi_i(x) = \max(0, W_{i,1}^{(1)}x + b_i)$$

The elbow is where:

$$W_{i,1}^{(1)}x + b_i = 0 \implies x = -\frac{b_i}{W_{i,1}^{(1)}}$$

So, the elbow location for the  $i$ -th ReLU is:

$$e_i = -\frac{b_i}{W_{i,1}^{(1)}}$$

### (d) Elbow Location After SGD Update

After one SGD update with learning rate  $\lambda$ :

$$W_{i,1}^{(1)'} = W_{i,1}^{(1)} - \lambda \frac{\partial \ell}{\partial W_{i,1}^{(1)}}$$

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

The new elbow location is:

$$e_i' = -\frac{b_i'}{W_{i,1}^{(1)'}} = -\frac{b_i - \lambda \frac{\partial \ell}{\partial b_i}}{W_{i,1}^{(1)} - \lambda \frac{\partial \ell}{\partial W_{i,1}^{(1)}}}$$

## Summary Table

Case	Elbow Movement	Slope Change
$\phi(x) = 0$	None	None
$w > 0, x > 0$	Left	Decrease
$w > 0, x < 0$	Left	Increase
$w < 0, x > 0$	Left	Less negative

### In summary:

- The elbow moves left when  $b$  decreases.
- The slope changes according to the sign of  $x$  and the update to  $w$ .
- For the full network, each ReLU's elbow is at  $e_i = -b_i/W_{i,1}^{(1)}$ , and updates move the elbow and change the slope as above.

3:55pm

↶ ↷ ↺ ↻

I hinted Mistral AI to redo part b with case-by-case analysis. Mistral AI gives the correct answer this time.

for part b, you need to carefully consider the magnitude of each parameter's movement. Explain case by case the move of elbow and slope.

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

- **No changes** occur to the slope or elbow.
- **Reason:** The gradient of ReLU at  $\phi(x) = 0$  is 0, so no update is made to  $w$  or  $b$ .

**(ii)**  $w > 0, x > 0, \phi(x) > 0$

**Updates:**

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

- **Slope:** Gets **shallower** (since  $w' = w - \lambda x$  and  $x > 0$ , so  $w' < w$ ).
- **Elbow:** Moves **right** if  $\lambda(bx - w) < 0$ , else **left**.

**Elbow Movement Condition:**

$$-\frac{b}{w} < -\frac{b - \lambda}{w - \lambda x}$$

$$\Rightarrow \lambda(bx - w) < 0$$

- If  $bx > w$ , the elbow moves **right**.
- If  $bx < w$ , the elbow moves **left**.

**Intuition:**

The elbow moves **right** if the bias  $b$  is large enough relative to  $w$  and  $x$ , otherwise it moves **left**.

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

**Updates:**

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

- **Slope:** Gets **steeper** (since  $x < 0$ ,  $w' = w - \lambda x > w$ ).
- **Elbow:** Moves **right** (since  $e' > e$ ).

**Intuition:**

- The negative  $x$  causes  $w$  to **increase**, making the slope steeper.
- The elbow moves **right** because the decrease in  $b$  and increase in  $w$  both push the elbow to the right.

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

**Updates:**

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

- **Slope:** Gets **steeper** (since  $w < 0$  and  $x > 0$ ,  $w' = w - \lambda x$  is less negative, i.e., closer to zero, but the magnitude increases).
- **Elbow:** Moves **left** (since  $e' < e$ ).

**Intuition:**

- The positive  $x$  makes  $w$  less negative, increasing the slope's magnitude.
- The elbow moves **left** because the decrease in  $b$  and the change in  $w$  both push the elbow to the left.

### Summary Table

Case	Slope Change	Elbow Movement	Condition for Elbow Movement
$\phi(x) = 0$	No change	No change	—
$w > 0, x > 0$	Shallower	Right or left	Right if $bx > w$ , else left
$w > 0, x < 0$	Steeper	Right	—
$w < 0, x > 0$	Steeper (less neg)	Left	—

### Visualization

For each case, you can sketch:

- The ReLU function before and after the update.
- The elbow location before and after.
- The change in slope.

**Example for  $w > 0, x > 0$ :**

- If  $bx > w$ , the elbow moves **right**.
- If  $bx < w$ , the elbow moves **left**.



This problem requires a careful, quantitative analysis of how the ratio  $-b/w$  (the elbow) changes.

The model's initial failure and subsequent correction highlight that it can struggle with problems requiring conditional reasoning based on the interaction of multiple parameters. Its language also lacks the precision required for rigorous mathematical communication.