

# CS182 Special Participation A: HW0

Alena Chao

December 5, 2025

## Executive Summary

Because the problems in this homework are relatively “simple” and well-defined (mostly standard vector calculus identities, SVD manipulations, and linear least-squares derivations) the model is generally able to answer each part correctly in a single shot. The tasks have clear mathematical structure, fixed conventions, and well-known closed-form solutions, which reduces ambiguity and makes the model’s responses more stable and reliable.

Across the problems, the model tends to follow a consistent pattern: it identifies the mathematical object being differentiated or solved for, checks shapes, rewrites expressions in a standard form (e.g., using the SVD or matrix calculus rules), and produces a clean final expression. For these kinds of textbook-style exercises, the model rarely makes conceptual errors.

## 2. Vector Calculus Review

### User

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.,

$$\frac{\partial f}{\partial x} = \left[ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right]$$

since a row acting on a column gives a scalar. You may have seen alternative conventions before, but the important thing is that you need to understand the types of objects and how they map to the shapes of the multidimensional arrays we use to represent those types.

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

### ChatGPT

Type check and plan before differentiating:

- $x \in \mathbb{R}^n$  is a column vector.
- $c \in \mathbb{R}^n$  is a column vector.
- $x^T c$  is a scalar.

By the problem convention, the derivative of a scalar with respect to a column vector is a **row vector**:  $\frac{\partial f}{\partial x} \in \mathbb{R}^{1 \times n}$ .

We will compute an arbitrary component  $\frac{\partial}{\partial x_j}(x^T c)$  and then assemble those components into the row vector.

**Elementwise calculation:**

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

Differentiate with respect to a single component  $x_j$ :

$$\frac{\partial}{\partial x_j}(x^T c) = \frac{\partial}{\partial x_j} \left( \sum_{i=1}^n x_i c_i \right) = \sum_{i=1}^n c_i \frac{\partial x_i}{\partial x_j} = \sum_{i=1}^n c_i \delta_{ij} = c_j$$

where  $\delta_{ij}$  is the Kronecker delta.

**Stacking the components:** Stacking the components into the row vector (in the order  $j = 1, \dots, n$ ) gives:

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

So,

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

User

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

ChatGPT

Type check first:

- $x \in \mathbb{R}^n$  is a column vector.
- $\|x\|_2^2 = x^T x$  is a scalar.

By the convention in the problem,  $\frac{\partial}{\partial x}(\text{scalar}) \in \mathbb{R}^{1 \times n}$ , so our final answer must be a **row vector**.

**Step 1: Write the function in coordinates**

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

**Step 2: Differentiate an arbitrary component** Take derivative w.r.t. a specific component  $x_j$ :

$$\frac{\partial}{\partial x_j} \left( \sum_{i=1}^n x_i^2 \right) = \sum_{i=1}^n \frac{\partial}{\partial x_j} (x_i^2) = \sum_{i=1}^n 2x_i \delta_{ij} = 2x_j$$

**Step 3: Stack all components** The derivative (row vector) is:

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

**Final result:**

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

(If the problem used the column-vector convention, the answer would be  $2x$ , but under this problem's row-vector convention, the correct form is  $2x^T$ .)

User

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

### ChatGPT

Let's carefully follow the problem's convention:

- $x \in \mathbb{R}^n$  is a column vector.
- $A \in \mathbb{R}^{m \times n}$ .
- $Ax \in \mathbb{R}^m$ .

The derivative  $\frac{\partial}{\partial x}(Ax)$  must be an  $m \times n$  matrix. (Why? Because each output component  $(Ax)_i$  is scalar, and its derivative w.r.t.  $x$  is a row vector, so stacking  $m$  rows gives an  $m \times n$  Jacobian).

#### Step 1: Write components

$$(Ax)_i = \sum_{j=1}^n A_{ij}x_j$$

**Step 2: Derivative of a single component** Take derivative with respect to a single coordinate  $x_k$ :

$$\frac{\partial}{\partial x_k}(Ax)_i = \frac{\partial}{\partial x_k} \left( \sum_{j=1}^n A_{ij}x_j \right) = \sum_{j=1}^n A_{ij}\delta_{jk} = A_{ik}$$

So the  $(i, k)$ -entry of the derivative matrix is:

$$\left[ \frac{\partial(Ax)}{\partial x} \right]_{ik} = A_{ik}$$

**Step 3: Stack all entries** Every entry matches the corresponding entry of  $A$ . Thus the derivative is simply:

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

### User

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

### ChatGPT

Let's go through it cleanly using the problem's convention that the vector derivative of a scalar w.r.t. a column vector is a row vector.

#### Type Check:

- $x \in \mathbb{R}^n$  (column vector)
- $A \in \mathbb{R}^{n \times n}$
- $x^T Ax$  is a scalar

Thus  $\frac{\partial}{\partial x}(x^T Ax) \in \mathbb{R}^{1 \times n}$  (a row vector).

#### Step 1: Expand the expression elementwise

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

#### Step 2: Take derivative w.r.t. a single component $x_k$

$$\frac{\partial}{\partial x_k}(x^T Ax) = \sum_{i,j} A_{ij} \frac{\partial}{\partial x_k}(x_i x_j)$$

Use product rule:  $\frac{\partial}{\partial x_k}(x_i x_j) = x_j \delta_{ik} + x_i \delta_{jk}$ . So:

$$\frac{\partial}{\partial x_k}(x^T Ax) = \sum_{i,j} A_{ij} (x_j \delta_{ik} + x_i \delta_{jk})$$

Simplify each term separately.

Term 1:

$$\sum_{i,j} A_{ij}x_j\delta_{ik} = \sum_j A_{kj}x_j = (Ax)_k$$

Term 2:

$$\sum_{i,j} A_{ij}x_i\delta_{jk} = \sum_i A_{ik}x_i = (A^T x)_k$$

**Step 3: Combine**

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

Stacking all components into a row vector:

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

Factor out  $x$ :

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

**Final Answer:**

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

User

Under what condition is  $\frac{\partial}{\partial x}(x^T Ax)$  equal to  $2x^T A$ ?

ChatGPT

We previously showed that in general:

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

For this to equal  $2x^T A$ , we need:

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

Subtract  $A$  from both sides:

$$A^T = A$$

**Condition:**  $A$  must be **symmetric**.

If  $A$  is symmetric, then  $A = A^T$ , and the derivative simplifies:

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

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

ANNOTATION: For this section, I noticed that Chat often gave more details than expected. For example, in part (a) and (d) they already plugged in the SVD, so I didn't have to ask part (b) and (e)

User

**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 a 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  $\Sigma^\dagger$  as 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 minimize the error between  $Xw$  and  $y$ ? In other words, we want to solve  $\min_w \|Xw - y\|^2$ .

### ChatGPT

#### (a) Overdetermined case ( $m > n$ )

We want:

$$\min_w \|Xw - y\|_2^2$$

Set the gradient to zero (normal equations):

$$\frac{\partial}{\partial w} \|Xw - y\|_2^2 = 2X^T(Xw - y) = 0 \implies X^T X w = X^T y$$

Using the full SVD  $X = U\Sigma V^T$  (with  $\Sigma \in \mathbb{R}^{m \times n}$  and  $\Sigma^\dagger \in \mathbb{R}^{n \times m}$ ), the least-squares solution can be written with the (Moore–Penrose) pseudoinverse:

$$w_{LS} = X^+ y = V \Sigma^\dagger U^T y$$

**Remarks / equivalent forms:** If  $X$  has full column rank ( $\text{rank}(X) = n$ ), then  $\Sigma$  has all  $n$  diagonal singular values nonzero and the solution is unique and equals:

$$w_{LS} = (X^T X)^{-1} X^T y = V \Sigma^{-1} U^T y$$

In the rank-deficient case, the set of minimizers is infinite; the pseudoinverse formula  $w = X^+ y$  picks the **minimum-norm** solution among them.

The fitted vector is the projection of  $y$  onto the column space of  $X$ :

$$Xw_{LS} = U\Sigma V^T V \Sigma^\dagger U^T y = U(\Sigma \Sigma^\dagger) U^T y = P_{\text{col}(X)} y$$

So the answer: the minimizer is  $w_{LS} = X^+ y$ .

### User

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.

### ChatGPT

If  $w^* = Ay$  is the least-squares solution and  $A$  is the pseudoinverse  $X^+ = V \Sigma^\dagger U^T$ , then consider:

$$XA = XX^+$$

Using the SVD  $X = U\Sigma V^T$ :

$$XA = U\Sigma V^T V \Sigma^\dagger U^T = U(\Sigma \Sigma^\dagger) U^T$$

What is  $\Sigma \Sigma^\dagger$ ?  $\Sigma$  is  $m \times n$  with singular values  $\sigma_i$  on the diagonal.  $\Sigma^\dagger$  is  $n \times m$  with  $1/\sigma_i$ . Thus:

$$\Sigma \Sigma^\dagger = \begin{bmatrix} 1 & & 0 \\ & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \\ & & 0 & 0 \end{bmatrix} \in \mathbb{R}^{m \times m}$$

It is the **orthogonal projector onto the column space of  $X$** . Therefore,  $XA = P_{\text{col}(X)}$ .  
**So why is  $A$  called a left-inverse?** A true left-inverse  $A$  of  $X$  would satisfy  $AX = I_n$ . When  $m > n$  (tall matrix),  $X$  cannot have a two-sided inverse. However:

- $AX = I_n$  does hold for the pseudoinverse if  $X$  has full column rank.
- $XA = P_{\text{col}(X)}$  acts as the identity on the column space of  $X$ .

Thus,  $A = X^+$  behaves like a left inverse on the subspace where an inverse can exist.

### User

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?

### ChatGPT

When  $m < n$ , the system  $Xw = y$  is underdetermined. The minimum-norm solution is the unique solution with the smallest Euclidean norm.

Write the full SVD  $X = U\Sigma V^T$ . Partition  $V = [V_r \ V_0]$  where  $V_r$  spans the row space and  $V_0$  spans the nullspace. Any  $w$  can be written as:

$$w = V_r a + V_0 b$$

The constraint  $Xw = y$  becomes:

$$U_r \Sigma_r V_r^T (V_r a + V_0 b) = U_r \Sigma_r a = y \implies a = \Sigma_r^{-1} U_r^T y$$

Thus the general solution is:

$$w = V_r \Sigma_r^{-1} U_r^T y + V_0 b$$

To minimize  $\|w\|_2^2$ , choose  $b = 0$ . Therefore the minimum-norm solution is:

$$w_{\min} = X^+ y = V \Sigma^\dagger U^T y$$

If  $X$  has full row rank ( $r = m$ ), this reduces to  $w_{\min} = X^T (XX^T)^{-1} y$ .

**Geometric facts:**  $w_{\min}$  lies in the row space of  $X$  (orthogonal to  $\text{null}(X)$ ), and it is the unique solution of  $Xw = y$  with smallest  $\ell_2$ -norm.

### User

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.

### ChatGPT

Since in the underdetermined case the **minimum-norm solution** is  $w^* = By = X^+y$ , let's examine what happens when we **right-multiply  $X$  by  $B$** :

$$XB = XX^+$$

In the underdetermined case ( $m < n$ ), the rank is at most  $m$ . Using SVD to compute  $XX^+$ :

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

In this case,  $\Sigma \in \mathbb{R}^{m \times n}$  and  $\Sigma^\dagger \in \mathbb{R}^{n \times m}$  result in:

$$\Sigma\Sigma^\dagger = I_m$$

Therefore:

$$XX^+ = I_m$$

**Why does this make  $B = X^+$  a right-inverse?** A matrix  $B$  is a right-inverse of  $X$  if  $XB = I_m$ . Indeed,  $XX^+ = I_m$ . The pseudoinverse gives the unique map that:

1. Recovers exactly the original  $y$  (because  $XX^+y = y$ ).
2. Returns the minimum-norm solution among all  $w$  satisfying  $Xw = y$ .

## 4. The 5 Interpretations of Ridge Regression

ANNOTATION: I notice that as the interpretations get more complex chat starts to diverge from the solutions more. For example in part (f) chat uses the Woodbury Identity which isn't necessary to solve the problem.

User

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

$$\operatorname{argmin}_w \|y - Xw\|_2^2 + \lambda\|w\|_2^2 \quad (1)$$

where  $X \in \mathbb{R}^{n \times d}$  is a data matrix, and  $y \in \mathbb{R}^n$  is the target vector. What's new compared to simple OLS is the addition of the  $\lambda\|w\|^2$  term, which acts as a "penalty".

Use vector calculus to expand the objective and solve this optimization problem for  $w$ .

ChatGPT

We want to minimize:

$$J(w) = \|y - Xw\|_2^2 + \lambda\|w\|_2^2$$

First, expand the quadratic term:

$$\|y - Xw\|_2^2 = (y - Xw)^T(y - Xw) = y^T y - 2y^T Xw + w^T X^T Xw$$

So the objective is:

$$J(w) = y^T y - 2y^T Xw + w^T X^T Xw + \lambda w^T w$$

Compute the gradient with respect to  $w$ . Using matrix derivative rules ( $\nabla_w(a^T w) = a$ ,  $\nabla_w(w^T Aw) = (A + A^T)w$ ), and noting  $X^T X$  is symmetric:

$$\nabla_w J(w) = -2X^T y + 2X^T Xw + 2\lambda w$$

Set the gradient to zero for a critical point:

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

Provided  $\lambda > 0$  (making  $X^T X + \lambda I$  positive definite and invertible), the unique minimizer is:

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

User

**(b) Perspective 2: "Hack" of shifting the Singular Values.** Let  $X = U\Sigma V^T$  be the (full) SVD of  $X$ . 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  $\sigma_i \gg \lambda$ ?

### ChatGPT

**Step 1: Substitute the SVD into  $X^T X + \lambda I$**

$$X^T X + \lambda I = (U \Sigma V^T)^T (U \Sigma V^T) + \lambda I = V \Sigma^T \Sigma V^T + \lambda I$$

Since  $V$  is orthonormal,  $I = VV^T$ , so we can factor  $V$ :

$$= V(\Sigma^T \Sigma + \lambda I)V^T$$

**Step 2: Invert and multiply by  $X^T$**

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

The resulting matrix is diagonal with entries:

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

**Step 3: Analyze the effect on singular values**

- **Case 1 ( $\sigma_i \ll \lambda$ ):**

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

Small singular values are heavily shrunk. Ridge suppresses directions of low variance (noise).

- **Case 2 ( $\sigma_i \gg \lambda$ ):**

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

Large singular values are almost unaffected (similar to OLS behavior).

### User

(c) **Perspective 3: MAP Estimation.** Ridge Regression can be viewed as finding the MAP estimate when we apply a prior  $W \sim \mathcal{N}(0, I)$  and view  $Y$  as generated by  $Y = XW + \sqrt{\lambda}N$  where  $N \sim \mathcal{N}(0, 1)$ . Show that (1) is the MAP estimate for  $W$  given  $Y = y$ .

### ChatGPT

**Step 1: Likelihood and Prior Likelihood ( $Y|W \sim \mathcal{N}(XW, \lambda I_n)$ ):**

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

Prior ( $W \sim \mathcal{N}(0, I_d)$ ):

$$p(W) \propto \exp\left(-\frac{1}{2}\|W\|_2^2\right)$$

**Step 2: Posterior**

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

**Step 3: MAP Optimization** Maximizing the posterior is equivalent to minimizing the negative log-posterior:

$$W^* = \operatorname{argmin}_W \left( \frac{1}{2\lambda}\|y - XW\|_2^2 + \frac{1}{2}\|W\|_2^2 \right)$$

Multiplying by  $2\lambda$ :

$$W^* = \operatorname{argmin}_W \|y - XW\|_2^2 + \lambda\|W\|_2^2$$

This is exactly the Ridge Regression objective.

User

(d) **Perspective 4: Fake Data.** Consider the augmented measurement vector  $\hat{y} = [y; 0_d]$  and data matrix  $\hat{X} = [X; \sqrt{\lambda}I_d]$ . Show that the classical OLS optimization problem  $\operatorname{argmin}_w \|\hat{y} - \hat{X}w\|_2^2$  has the same minimizer as (1).

ChatGPT

Expand the augmented objective:

$$\begin{aligned}\|\hat{y} - \hat{X}w\|_2^2 &= \left\| \begin{bmatrix} y \\ 0 \end{bmatrix} - \begin{bmatrix} X \\ \sqrt{\lambda}I_d \end{bmatrix} w \right\|_2^2 \\ &= \left\| \begin{bmatrix} y - Xw \\ -\sqrt{\lambda}w \end{bmatrix} \right\|_2^2\end{aligned}$$

Summing the squared norms of the blocks:

$$= \|y - Xw\|_2^2 + \|\sqrt{\lambda}w\|_2^2 = \|y - Xw\|_2^2 + \lambda\|w\|_2^2$$

This recovers the Ridge objective exactly. Adding  $\sqrt{\lambda}I_d$  as "fake data" rows pulls  $w$  towards zero.

User

(e) **Perspective 5: Fake Features.** Construct an augmented design matrix  $\check{X} = [X \ \sqrt{\lambda}I_n]$  (horizontal stacking). The problem is underdetermined. Show that the minimum-norm solution to  $\check{X}\eta = y$  corresponds to Ridge regression.

ChatGPT

Let  $\eta = [w; z]$  where  $w \in \mathbb{R}^d$  and  $z \in \mathbb{R}^n$ . The constraint  $\check{X}\eta = y$  becomes:

$$Xw + \sqrt{\lambda}z = y \implies z = \frac{1}{\sqrt{\lambda}}(y - Xw)$$

We minimize  $\|\eta\|_2^2$ :

$$\|\eta\|_2^2 = \|w\|_2^2 + \|z\|_2^2 = \|w\|_2^2 + \frac{1}{\lambda}\|y - Xw\|_2^2$$

Multiplying by  $\lambda$ , we minimize:

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

This is the Ridge objective. The first  $d$  components of the min-norm solution  $\eta^*$  give the Ridge estimator  $w^*$ .

User

(f) Use the pseudo-inverse  $A^\dagger = A^T(AA^T)^{-1}$  on the augmented matrix from (e) to show that  $\hat{w} = X^T(XX^T + \lambda I)^{-1}y$ , and show this is equivalent to the standard formula.

ChatGPT

**Step 1: Pseudo-inverse solution** For  $\check{X} = [X \ \sqrt{\lambda}I_n]$ , the min-norm solution is  $\eta = \check{X}^T(\check{X}\check{X}^T)^{-1}y$ . Compute  $\check{X}\check{X}^T$ :

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

Now compute  $\check{X}^T(\check{X}\check{X}^T)^{-1}y$ :

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

The top part (corresponding to  $w$ ) is:

$$w^* = X^T(XX^T + \lambda I)^{-1}y \quad (\text{Kernel Ridge Form})$$

**Step 2: Equivalence via Woodbury Identity** Using the identity  $(I+AB)^{-1}A = A(I+BA)^{-1}$  (scaled appropriately):

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

Thus, the min-norm solution on fake features is equivalent to the standard Ridge solution.

User

(g) What happens when  $\lambda \rightarrow \infty$ ? Why is this called "shrinkage"?

ChatGPT

As  $\lambda \rightarrow \infty$ :

$$w_r = \frac{1}{\lambda} \left( \frac{1}{\lambda} X^T X + I \right)^{-1} X^T y \rightarrow 0$$

The regularization term  $\lambda \|w\|^2$  dominates the objective, forcing all weights to shrink toward zero. This reduces variance but introduces bias.

User

(h) What happens when  $\lambda \rightarrow 0$  for tall vs. wide matrices?

ChatGPT

- **Tall Matrix ( $n > d$ ):**  $X^T X$  is generally invertible.

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

This recovers the standard **OLS solution**.

- **Wide Matrix ( $n < d$ ):**  $X^T X$  is singular.

$$\lim_{\lambda \rightarrow 0} w_r = X^T (XX^T)^{-1} y$$

This recovers the **minimum-norm solution** to the underdetermined system  $Xw = y$ .

## 5. ReLU Elbow Update under SGD

User

(a) Let's start by examining the behavior of a single ReLU with a linear function of  $x$  as the input:

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

We define a loss function  $\ell(x, y, \phi) = \frac{1}{2} \|\phi(x) - y\|_2^2$ . Find the following: (i) The location of the 'elbow'  $e$  of the function. (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  $\frac{\partial \ell}{\partial b}$ .

### ChatGPT

(i) **Location of the ‘elbow’**  $e$  The elbow is where the ReLU activates ( $wx + b = 0$ ):

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

(ii) **Derivative w.r.t.  $\phi(x)$**

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

(iii) **Partial derivative w.r.t.  $w$**  Using the chain rule  $\frac{\partial\ell}{\partial w} = \frac{d\ell}{d\phi} \cdot \frac{d\phi}{dw}$ :

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

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

(iv) **Partial derivative w.r.t.  $b$**  Using the chain rule  $\frac{\partial\ell}{\partial b} = \frac{d\ell}{d\phi} \cdot \frac{d\phi}{db}$ :

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

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

### User

(b) Suppose we have a training point  $(x, y)$  such that  $\phi(x) - y = 1$  (we are too high). 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)  $w > 0, x > 0, \phi(x) > 0$ . (iii)  $w > 0, x < 0, \phi(x) > 0$ . (iv)  $w < 0, x > 0, \phi(x) > 0$ . Draw and label  $\phi(x)$ , the elbow, and qualitative changes.

### ChatGPT

The SGD update is  $p' = p - \lambda \frac{\partial\ell}{\partial p}$ . Since  $\phi(x) - y = 1 > 0$ , the gradient attempts to decrease the output. Recall elbow  $e = -b/w$ . New elbow  $e' = -b'/w'$ .

(i)  $\phi(x) = 0$  (**Inactive**) Gradients are zero. **No update** occurs. Slope and elbow remain unchanged.

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

- Gradients:  $\frac{\partial\ell}{\partial w} = x > 0, \frac{\partial\ell}{\partial b} = 1 > 0$ .
- Update:  $w$  decreases,  $b$  decreases.
- **Slope:** Decreases (becomes less steep).
- **Elbow:** Shifts slightly (typically right for small  $\lambda$ , as numerator and denominator both decrease).

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

- Gradients:  $\frac{\partial\ell}{\partial w} = x < 0, \frac{\partial\ell}{\partial b} = 1 > 0$ .
- Update:  $w$  increases (becomes steeper),  $b$  decreases.
- **Slope:** Increases.
- **Elbow:** Shifts **right** (numerator decreases, denominator increases).

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

- Gradients:  $\frac{\partial \ell}{\partial w} = x > 0$ ,  $\frac{\partial \ell}{\partial b} = 1 > 0$ .
- Update:  $w$  decreases (becomes more negative),  $b$  decreases.
- **Slope:** Becomes more negative.
- **Elbow:** Shifts left (numerator decreases, denominator becomes more negative).

### User

(c) Derive the location  $e_i$  of the elbow of the  $i$ -th elementwise ReLU activation for the full network  $\hat{f}(x) = W^{(2)}\Phi(W^{(1)}x + b)$ .

### ChatGPT

For the  $i$ -th hidden unit:

$$\phi_i(x) = \text{ReLU}(w_i^{(1)}x + b_i)$$

The elbow occurs where the argument is zero:

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

Solving for  $e_i$ :

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

### User

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

### ChatGPT

#### Step 1: Gradients for the full network

$$\frac{\partial \ell}{\partial w_i^{(1)}} = (\hat{f}(x) - y) \cdot W_i^{(2)} \cdot x \cdot \mathbb{I}(\text{active})$$

$$\frac{\partial \ell}{\partial b_i} = (\hat{f}(x) - y) \cdot W_i^{(2)} \cdot \mathbb{I}(\text{active})$$

#### Step 2: Parameter Updates

$$w_i^{(1)'} = w_i^{(1)} - \lambda \frac{\partial \ell}{\partial w_i^{(1)}}$$

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

#### Step 3: New Elbow Location

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

Substituting the updates:

$$e_i' = -\frac{b_i - \lambda(\hat{f}(x) - y)W_i^{(2)}\mathbb{I}(w_i^{(1)}x + b_i > 0)}{w_i^{(1)} - \lambda(\hat{f}(x) - y)W_i^{(2)}x\mathbb{I}(w_i^{(1)}x + b_i > 0)}$$

**Interpretation:** If the unit is inactive, the elbow does not move. If active, the shift depends on the error sign, the output weight  $W_i^{(2)}$ , and the input  $x$ .