EECS 182 Deep Neural Networks
Spring 2025 Anant Sahai

Homework 0

This homework is due on Sunday, Feb 2, 2025, at 11:59PM. A failure to do this homework will be deemed "insufficient engagement" and will be used to make room in the class for students who are actually committed to doing the work.

1. Reflection on your learning goals at the start of the semester

Deep learning is a particularly challenging subject *culturally* given the state of our understanding as well as the rapid advancements and economic/cultural/intellectual impacts of this emergent technology. You've come into this class with your own individual background. Think about it and do a little online exploration and concisely write up your response.

(a) **Before doing any further reading or exploration and just based on what you know, please briefly describe what you think your learning goals are.**

(b) Open up any modern top-tier LLM-based chat system — you can use the Berkeley promotion of Perplexity for example, or just use OpenAI's ChatGPT, Anthropic's Claude, Google's Gemini, etc. Interact with the system to have a conversation about Deep Learning. **After this interaction, please describe how — if at all — this has modified your learning goals.**

(c) Now, the approach taken in this course is a reasonably intellectually conservative point of view that tries to approach Deep Learning through a lens of understanding that leverages mathematical intuition and models, as well as connections to the larger Machine Learning tradition, to the extent possible. However, it is important to understand that this is not the only possible perspective. At least skim through David Donoho's 2024 Paper "Data Science at the Singularity" available at https://doi.org/10.1162/99608f92.b91339ef. **After reading that paper, how have you thought about the role of this class in your learning?**

(d) **Finally, comment briefly about how your own understanding of deep learning can benefit from activities that complement what we are doing in this course? What kind of guidance from your peers and course staff do you think would be helpful?**

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

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

Solution: This is a vector derivative of a scalar quantity, so our result will be a row vector. Looking at the i-th entry, $\frac{\partial}{\partial x_i}(\mathbf{x}^T\mathbf{c}) = \frac{\partial}{\partial x_i}(\sum_j c_j x_j) = c_i$. Stacking all the entries into a row vector, we get $\mathbf{c}^T$.

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

Solution: This is a vector derivative of a scalar quantity, so our result will be a row vector. Looking at the i-th entry, $\frac{\partial}{\partial x_i}(||\mathbf{x}||_2^2) = \frac{\partial}{\partial x_i}(\sum_j x_j^2) = 2x_i$. Stack all the entries into a row to get $2\mathbf{x}^T$.

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

Solution: This is a vector derivative of a vector quantity, so the result will be a matrix. Let $\mathbf{f} = A\mathbf{x}$. Note that $f_i = \sum_k A_{ik}x_k$

Looking at the (i, j)-th entry of our matrix, $\frac{\partial f_i}{\partial x_j} = \frac{\partial}{\partial x_j}(\sum_k A_{ik}x_k) = A_{ij}$. Arranging all of these in a matrix will recover A.

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

Solution: This is a vector derivative of a scalar quantity, so our result will be a vector. Before taking any derivatives, we can write $\mathbf{x}^T A\mathbf{x} = \sum_i \sum_j A_{ij}x_i x_j$. Taking the derivative with respect to an arbitrary x_k and focusing on just the terms involving x_k (as the derivative of the other terms wrt x_k is zero), we can write

$$\frac{\partial}{\partial x_k}(\mathbf{x}^T A\mathbf{x}) = \frac{\partial}{\partial x_k}((\sum_{j\neq k} A_{kj}x_k x_j) + (\sum_{i\neq k} A_{ik}x_i x_k) + A_{kk}x_k^2)$$

$$= (\sum_{j\neq k} A_{kj}x_j) + (\sum_{i\neq k} A_{ik}x_i) + 2A_{kk}x_k$$

$$= (\sum_{j} A_{kj}x_j) + (\sum_{i} A_{ik}x_i) = \sum_i (A_{ki}x_i + A_{ik}x_i)$$

$$= \mathbf{x}^T(\text{kth row of A} + \text{kth col of A}) = \mathbf{x}^T(A_k + A_k^T)$$

Stacking all the results in a row vector, we get $\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}^T A\mathbf{x}) = \mathbf{x}^T(A + A^T)$ as desired.

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

Solution: We want $(A+A^T) = 2A$. This is true if and only if $A = A^T$, ie. the matrix A is symmetric.

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

Consider the equation $X\mathbf{w} = \mathbf{y}$, 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 Σ is an $m \times n$ matrix with non-zero singular values (σ_i) on the "diagonal".

For this problem, we define $\Sigma^\dagger$ an $n \times m$ matrix with the reciprocals of the singular values ($\frac{1}{\sigma_i}$) 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 $X\mathbf{w}$ and y?** In other words, we want to solve $\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2$.

Solution: Meta: Students may be confused about which form of SVD to be using. Make sure they know it is FULL SVD, U and V are square orthonormal matrices.

This is the classic least squares problem. The solution is given by

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

This can be derived from vector calculus, and also has an elegant interpretation in the context of orthogonal projection of **y** on the column space of X.

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

Solution:

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

Since U has orthonormal columns, $U^T U = I$. Notice $\Sigma^T \Sigma$ is a square, $n \times n$ diagonal matrix with squared singular values σ_i^2 along the diagonal.

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

Apply the fact that $(AB)^{-1} = B^{-1}A^{-1}$, and that $V^{-1} = V^T$ since the matrix is orthonormal.

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

Simplify since $V^T V = I$.

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

Notice that $(\Sigma^T \Sigma)^{-1} \Sigma^T$ is an $n \times m$ matrix with the reciprocals of the singular values, $\frac{1}{\sigma_i}$, on the "diagonal". We can call this matrix $\Sigma^\dagger$. Note that this isn't a true matrix inverse (since the matrix Σ is not square). So we can write our answer as

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

You should draw out the matrix shapes and convince yourself that all the matrix multiplications make sense.

(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.

Solution: $(X^T X)^{-1} X^T X = I$. We can also see this from our SVD interpretation,

$$V\Sigma^\dagger U^T U \Sigma V^T = V \Sigma^\dagger \Sigma V^T = V V^T = I$$

Students should make sure to understand why $\Sigma^\dagger \Sigma = I$ (What are the dimensions, and what are the entries?)

This is why 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, ie. we want to solve $\min \|\mathbf{w}\|^2 \text{s.t.} X\mathbf{w} = \mathbf{y}$. **What is the minimum norm solution?**

Solution: The min-norm problem is solved by

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

We can see this by choosing **w** that has a zero component in the nullspace of X, and thus **w** is in the range of X^T. Alternatively, one can write the Lagrangian, take the dual, apply the KKT conditions,

and solve to get the same answer.

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

Solution:

$$\begin{aligned} X^T(XX^T)^{-1} &= (U\Sigma V^T)^T(U\Sigma V^T(U\Sigma V^T)^T)^{-1} \\ &= V\Sigma^T U^T(U\Sigma V^T V\Sigma^T U^T)^{-1} \\ &= V\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T \\ &= V\Sigma^T(\Sigma\Sigma^T)^{-1}U^T \end{aligned}$$

Here, we have that $\Sigma^T(\Sigma\Sigma^T)^{-1}$ is an $n \times m$ matrix with the reciprocals of the singular values, $\frac{1}{\sigma_i}$, on the "diagonal". We can call this matrix $\Sigma^\dagger$ so that we have

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

(f) 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.

Solution: Similar to the previous part, $XX^T(XX^T)^{-1} = I$. This can also be seen from the SVD perspective.

This is why the min-norm solution is called the right-inverse.

4. The 5 Interpretations of Ridge Regression

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

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

where $X \in \mathbb{R}^{n\times d}$ 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 $\lambda\|\mathbf{w}\|^2$ 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 $\mathbf{w}$.

Solution: Call our objective f. Expand

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

Take gradient wrt $\mathbf{w}$ and set it to zero:

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

Solve for $\mathbf{w}$:

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

(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 should check your work for the previous part).

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 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 $\sigma_i << \lambda$? What about when $\sigma_i >> \lambda$?**

Solution: We want to plug in $X = U\Sigma V^T$ into $\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$. U and V are square (although they may be different sizes). Recall that U and V are orthonormal (real unitary) matrices, so $U^T U = UU^T = I$ and $VV^T = V^T V = I$.

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

Now, let's consider the case when $n > d$. We have

$$\begin{aligned}
\mathbf{w} &= V \begin{bmatrix} \sigma_1^2 + \lambda & 0 & \dots & 0 \\ 0 & \sigma_2^2 + \lambda & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & \sigma_d^2 + \lambda \end{bmatrix}^{-1} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \dots & & & & & & \\ 0 & 0 & \dots & \sigma_d & 0 & \dots & 0 \end{bmatrix} U^T \mathbf{y} \\
&= V \begin{bmatrix} \frac{\sigma_1}{\sigma_1^2 + \lambda} & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \frac{\sigma_2}{\sigma_2^2 + \lambda} & \dots & 0 & 0 & \dots & 0 \\ \dots & & & & & & \\ 0 & 0 & \dots & \frac{\sigma_d}{\sigma_d^2 + \lambda} & 0 & \dots & 0 \end{bmatrix} U^T \mathbf{y}
\end{aligned}$$

You can see the "diagonal" terms in the SVD form are $\frac{\sigma_i}{\sigma_i^2 + \lambda}$. By adding the λ to the denominator, we prevent an explosion if any σ_i was close to zero. That is, when $\sigma_i << \lambda$, the σ_i^2 term becomes negligible and the effective singular value of the matrix in the solution is $\frac{\sigma_i}{\lambda}$. When $\sigma_i >> \lambda$, the λ term is negligible and the effective singular value of the matrix in the solution is $\frac{1}{\sigma_i}$, the same as it would be without any regularization.

The case when $n = d$ is similar, but without the extra zero-columns since the matrix Σ is square. The case when $n < d$ is also similar, except we would now have additional rows of 0's below the square. You can work out for yourself to see what those look like.

(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) $\mathbf{W}$. In particular, we can think of the prior for $\mathbf{W}$ as being $\mathcal{N}(\mathbf{0}, I)$ and view the random Y as being generated using $Y = \mathbf{x}^T \mathbf{W} + \sqrt{\lambda} N$ where the noise N is distributed iid (across training samples) as $\mathcal{N}(0, 1)$. 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 $\mathbf{Y} = \mathbf{y}$.

Solution:

From how we define MAP estimation,

$$MAP(\mathbf{w}|\mathbf{Y} = \mathbf{y}) = \underset{\mathbf{w}}{\text{argmax}}\, f(\mathbf{w}|\mathbf{Y} = \mathbf{y})$$

$$= \underset{\mathbf{w}}{\text{argmax}}\, \frac{f(\mathbf{w}, \mathbf{y})}{f(\mathbf{y})}$$

The denominator doesn't affect the argmax since it doesn't depend on $\mathbf{w}$, so we can omit it. Then we can use the chain rule to expand out the numerator

$$= \underset{\mathbf{w}}{\text{argmax}}\, f(\mathbf{w}) f(\mathbf{y}|\mathbf{w})$$

Now, split up the conditional joint density into the product of conditional densities since each element of the $\mathbf{y}$ is independent given $\mathbf{w}$.

$$= \underset{\mathbf{w}}{\text{argmax}}\, f(\mathbf{w}) \prod_{i=1}^{n} f(y_i|\mathbf{w})$$

We can now recall the formula for standard normal pdf is $f_Z(z) = \frac{e^{-z^2/2}}{\sqrt{2\pi}}$. To find $f(y_i|\mathbf{w})$, we know that $y_i = \mathbf{x_i}^T\mathbf{w} + \sqrt{\lambda}N_i$, where $N_i \sim \mathcal{N}(0, 1)$, so we'd have $\frac{y_i - \mathbf{x_i}^T\mathbf{w}}{\sqrt{\lambda}} \sim \mathcal{N}(0, 1)$. Now, plugging in the pdf in the previous expressions we'd have

$$MAP = \underset{\mathbf{w}}{\text{argmax}}\, \frac{e^{-\|\mathbf{w}\|^2/2}}{\sqrt{2\pi}} \prod_{i=1}^{n} \frac{e^{-(\frac{y_i - \mathbf{x_i}^T\mathbf{w}}{\sqrt{\lambda}})^2/2}}{\sqrt{2\pi}}$$

We can ignore multiplicative scaling constants (since they don't affect the value of the argmax). Also, since log is a monotonically increasing function, we can take log of both sides without affecting the argmax. Taking logs is useful since it allows us to turn products into sums. So we now have:

$$MAP = \underset{\mathbf{w}}{\text{argmax}} -\frac{\|\mathbf{w}\|^2}{2} - \frac{1}{\lambda}\sum_i \frac{(y_i - \mathbf{x}_i^T\mathbf{w})^2}{2}$$

We can ignore scaling factors again, and arrange all the summation terms in a vector and taking the norm-squared. We can also turn argmax into argmin by negating the objective function:

$$MAP = \underset{\mathbf{w}}{\text{argmin}}\, \|\mathbf{w}\|^2 + \frac{1}{\lambda}\|\mathbf{y} - X\mathbf{w}\|^2$$

Finally, we can multiply through by λ without changing the argmin since it is a positive constant:

$$MAP = \underset{\mathbf{w}}{\text{argmin}}\, \lambda\|\mathbf{w}\|^2 + \|X\mathbf{w} - \mathbf{y}\|^2$$

And now it is the same form as the original Ridge Regression optimization problem.

(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 $\hat{\mathbf{X}}$:

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

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)**.**

Solution: There are two easy ways of seeing the answer. The first is to look at the optimization problem itself and expand out the terms.

Recall that $\|\hat{\mathbf{y}} - \hat{\mathbf{X}}\mathbf{w}\|_2^2 = \sum_{i=1}^{n+d}(\hat{y}_i - \hat{\mathbf{x}}_i\mathbf{w})^2$ where $\hat{\mathbf{x}}_i$ are rows of $\hat{\mathbf{X}}$: the squared norm of the error is the sum of squared errors in individual coordinates. Our augmentation adds d more terms to that sum, which exactly give the ridge regularization. To see that we can write

$$\begin{aligned}
\sum_{i=1}^{n}(\hat{y}_i - \hat{\mathbf{x}}_i\mathbf{w})^2 &= \sum_{i=1}^{n}(y_i - \mathbf{x}_i\mathbf{w})^2 = \|\mathbf{y} - X\mathbf{w}\|_2^2 \\
\sum_{i=n+1}^{d}(\hat{y}_i - \hat{\mathbf{x}}_i\mathbf{w})^2 &= \sum_{i=n+1}^{d}(\sqrt{\lambda}w_i)^2 = \lambda\|\mathbf{w}\|_2^2 \\
\sum_{i=1}^{n+d}(\hat{y}_i - \hat{\mathbf{x}}_i\mathbf{w})^2 &= \|\mathbf{y} - X\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_2^2.
\end{aligned}$$

Alternatively, we can look at the solution and simplify it. We know that the solution to ordinary least squares for the augmented data is just

$$\begin{aligned}
(\hat{X}^\top\hat{X})^{-1}\hat{X}^\top\hat{\mathbf{y}} &= \left(\begin{bmatrix} X \\ \sqrt{\lambda}\mathbf{I}_d \end{bmatrix}^\top \begin{bmatrix} X \\ \sqrt{\lambda}\mathbf{I}_d \end{bmatrix}\right)^{-1} \begin{bmatrix} X \\ \sqrt{\lambda}\mathbf{I}_d \end{bmatrix}^\top \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} \\
&= \left(\begin{bmatrix} X^\top & \sqrt{\lambda}\mathbf{I}_d \end{bmatrix} \begin{bmatrix} X \\ \sqrt{\lambda}\mathbf{I}_d \end{bmatrix}\right)^{-1} \begin{bmatrix} X^\top & \sqrt{\lambda}\mathbf{I}_d \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} \\
&= (X^\top X + \lambda\mathbf{I}_d)^{-1}X^\top\mathbf{y}
\end{aligned}$$

Notice that this is the same as the solution for ridge regression. Either way, we get the desired result.

(e) *Perspective 5: Fake Features.* For this last interpretation, let's instead construct an augmented design matrix in the following way:

$$\check{\mathbf{X}} = [X \ \sqrt{\lambda}\mathbf{I}_\mathbf{n}]$$

i.e. we stack X with $\sqrt{\lambda}\mathbf{I}_n$ 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 $\boldsymbol{\eta} \in \mathbb{R}^{d+n}$ for which $\check{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}$. We are interested in the **min-norm** solution, ie. the solution to

$$\operatorname*{argmin}_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_2^2 \text{ s.t. } \check{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}. \tag{2}$$

Show that this is yet another form of ridge regression and that the first d **coordinates of** $\boldsymbol{\eta}^*$ **form**

the minimizer of (1).

Solution: Let's look inside the $d+n$ dimensional vector $\boldsymbol{\eta}$ by writing it as $\boldsymbol{\eta} = \begin{bmatrix} \mathbf{w} \\ \boldsymbol{\xi} \end{bmatrix}$. Here, $\mathbf{w}$ is d-dimensional and $\boldsymbol{\xi}$ is n-dimensional. Then (2) expands to

$$\underset{\mathbf{w},\boldsymbol{\xi}}{\operatorname{argmin}} \|\mathbf{w}\|_2^2 + \|\boldsymbol{\xi}\|_2^2 \text{ s.t. } X\mathbf{w} + \sqrt{\lambda}\boldsymbol{\xi} = \mathbf{y}.$$

The constraint just says that $\sqrt{\lambda}\boldsymbol{\xi} = \mathbf{y} - X\mathbf{w}$. In other words, $\sqrt{\lambda}\boldsymbol{\xi}$ is the classic residual. This yields $\boldsymbol{\xi} = \frac{1}{\sqrt{\lambda}}(\mathbf{y} - X\mathbf{w})$ and plugging that into the first part we get

$$\underset{\mathbf{w},\boldsymbol{\xi}}{\operatorname{argmin}} \|\mathbf{w}\|_2^2 + \frac{1}{\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2.$$

When considering whether the optimization problem is equivalent, we need to think about the minimizer and not the minimum itself. For this, we simply notice that scaling the objective by a constant factor doesn't change the minimizers and so:

$$\underset{\mathbf{w},\boldsymbol{\xi}}{\operatorname{argmin}} \|\mathbf{w}\|_2^2 + \frac{1}{\lambda}\|\mathbf{y} - X\mathbf{w}\|_2^2 = \underset{\mathbf{w},\boldsymbol{\xi}}{\operatorname{argmin}} \lambda\|\mathbf{w}\|_2^2 + \|\mathbf{y} - X\mathbf{w}\|_2^2$$

which is equivalent to (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 $A\boldsymbol{\eta} = \mathbf{z}$. That is, the optimization problem

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

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

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

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

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

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

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

Solution: First, we simply need to plug in our matrix into the pseudo-inverse formula provided and simplify

$$\hat{X}^\top(\hat{X}\hat{X}^\top)^{-1}\hat{\mathbf{y}} = \begin{bmatrix} X & \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}^\top \left(\begin{bmatrix} X & \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}\begin{bmatrix} X & \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}^\top\right)^{-1}\mathbf{y}$$

$$\begin{bmatrix} \hat{\mathbf{w}} \\ \boldsymbol{\xi} \end{bmatrix} = \begin{bmatrix} X^\top \\ \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}\left(\begin{bmatrix} X & \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}\begin{bmatrix} X^\top \\ \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}\right)^{-1}\mathbf{y}$$

$$= \begin{bmatrix} X^\top \\ \sqrt{\lambda}\mathbf{I}_n \end{bmatrix}(XX^\top + \sqrt{\lambda}^2\mathbf{I})^{-1}\mathbf{y}$$

Looking at just the top d terms, we see $\hat{\mathbf{w}} = X^T(XX^T + \lambda\mathbf{I})^{-1}\mathbf{y}$ as desired.

Now, let's show the two forms of Ridge Regression are equivalent. That is, let's show

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

With the expression above, we can left-multiply both sides by $(X^TX + \lambda \mathbf{I})$ and right-multiply both sides by $(XX^T + \lambda \mathbf{I})$ to get

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

And distributing the matrix multiplication we have

$$X^TXX^T + \lambda X^T = X^TXX^T + \lambda X^T$$

which we can see is always true as desired.

(g) We know that the solution to ridge regression (1) is given by $\hat{\mathbf{w}}_\mathbf{r} = (X^\top X + \lambda \mathbf{I})^{-1}X^\top \mathbf{y}$. **What happens when $\lambda \to \infty$?** It is for this reason that sometimes ridge regularization is referred to as "shrinkage."

Solution:

As $\lambda \to \infty$ the matrix $(X^\top X + \lambda \mathbf{I})^{-1}$ converges to the zero matrix, and so we have $\mathbf{w} = \mathbf{0}$.

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

Solution:

When X is wide (underdetermined), we converge to the min-norm solution.

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

When X is tall (overdetermined), we converge to the OLS solution.

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

Both of these can be seen by using the relevant form of Ridge Regression and plugging in $\lambda = 0$ directly, or by using the SVD perspective.

5. 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

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

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,

$$\ell\left(x, y, \mathbf{W}^{(1)}, \mathbf{b}, \mathbf{W}^{(2)}\right) = \frac{1}{2}\left\|\hat{f}(x) - y\right\|_2^2.$$

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,

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

Notice that the slope of $\phi(x)$ is w in the non-zero domain.
We define a loss function $\ell(x, y, \phi) = \frac{1}{2}\|\phi(x) - y\|_2^2$. **Find the following:**

(i) The location of the 'elbow' e of the function, where it transitions from 0 to something else.

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

(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 $\frac{\partial \ell}{\partial b}$

Solution:

(i)
$$e = -\frac{b}{w}$$

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

(iii)
$$\frac{\partial \ell}{\partial w} = \frac{\partial \ell}{\partial \phi}\frac{\partial \phi}{\partial w} = \begin{cases} (\phi(x) - y)x, & wx + b > 0 \\ 0, & \text{else} \end{cases}$$

(iv)
$$\frac{\partial \ell}{\partial b} = \frac{\partial \ell}{\partial \phi}\frac{\partial \phi}{\partial b} = \begin{cases} (\phi(x) - y), & wx + b > 0 \\ 0, & \text{else} \end{cases}$$

(b) Now suppose we have some training point (x, y) such that $\phi(x) - y = 1$. In other words, the prediction $\phi(x)$ is 1 unit above the target y — we are too high and are trying to pull the function downward.
Describe what happpens to the slope and elbow of $\phi(x)$ when we perform gradient descent in the following cases:

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

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

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

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

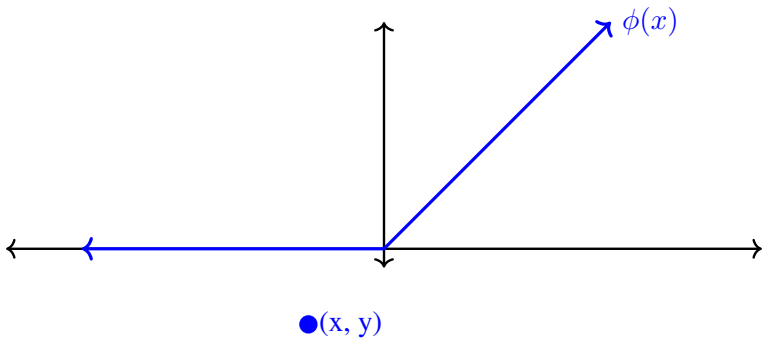
Additionally, draw and label $\phi(x)$, the elbow, and the qualitative changes to the slope and elbow after a gradient update to w and b. You should label the elbow location and a candidate (x, y) pair. Remember that the update for some parameter vector $\mathbf{p}$ and loss ℓ under SGD is

$$\mathbf{p}' = \mathbf{p} - \lambda \nabla_{\mathbf{p}}(\ell),\ \lambda > 0.$$

Solution: For each of the cases the change in the slope is determined by $\frac{\partial \ell}{\partial w}$ and the new elbow is located at

$$e' = \frac{-(b - \Delta b)}{w - \Delta w} = \frac{-b + \lambda \frac{\partial \ell}{\partial b}}{w - \lambda \frac{\partial \ell}{\partial w}}$$

(i) No changes since the derivatives are 0.



(ii) The slope gets shallower but the elbow can move left or right depending on the step size.

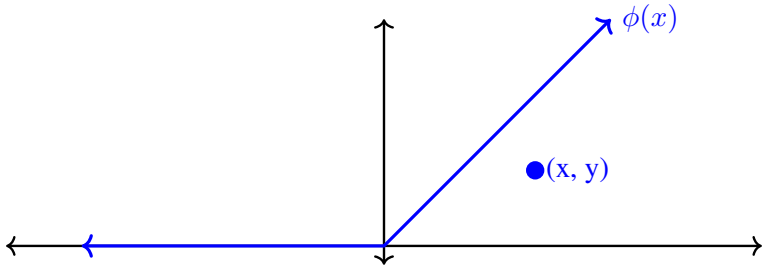
$$\frac{\partial \ell}{\partial w} = 1x > 0 \implies w > w - \lambda x$$

$$\frac{\partial \ell}{\partial b} = 1 \implies b > b - \lambda$$

Thus, the elbow moves from $-\frac{b}{w}$ to $-\frac{b-\lambda}{w-\lambda x}$. The elbow moves right if and only if

$$\begin{aligned} & -\frac{b}{w} < \frac{b-\lambda}{w-\lambda x} \\ \Rightarrow & \frac{\lambda(bx-w)}{w(w-\lambda x)} < 0 \\ \Rightarrow & \frac{b-\frac{w}{x}}{\frac{w}{x}-\lambda} < 0 \\ \Rightarrow & (b-\frac{w}{x})(\lambda-\frac{w}{x}) > 0. \end{aligned}$$

In other words, if the bias b and the step size λ are on the same side of $\frac{w}{x}$, then the elbow moves right, and left otherwise.
Several numerical checks show the motion of the elbow.

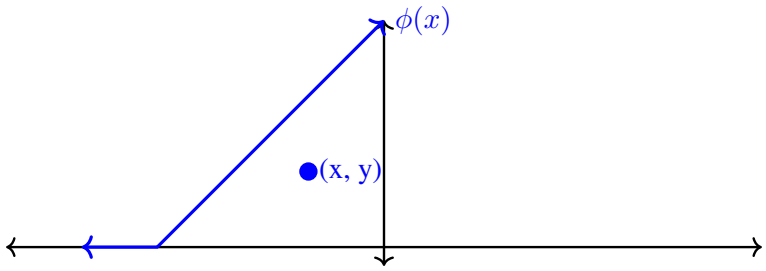


(iii) The slope gets steeper and the elbow moves right. Using the fact that $w, b > 0$ and $e < 0$ for this configuration,

$$\frac{\partial \ell}{\partial w} = 1x < 0 \implies |w| < |w - \lambda x|$$

$$\frac{\partial \ell}{\partial b} = 1 \implies |b| > |b - \lambda|$$

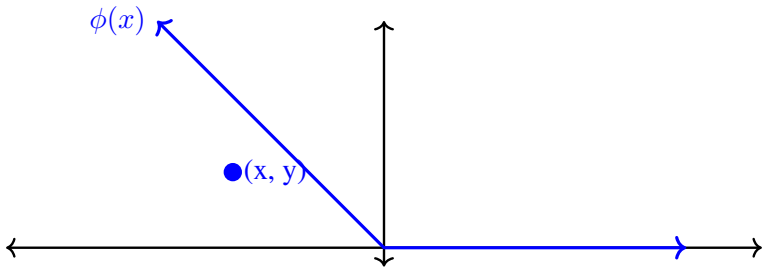
$$\therefore e' > e$$



(iv) The slope gets steeper and the elbow moves left. Since $w, x < 0$ for this configuration,

$$\frac{\partial \ell}{\partial w} = 1x > 0 \implies |w| < |w - \lambda x|$$

Several numerical checks show the motion of the elbow.



Believe it or not, even when the slope moves in the opposite direction you expect it to the error still decreases. You can try this yourself with a learning rate of 0.1, $x = 1$, $y = 1$, $w = -2$, and $b = 4$.

We encourage you to check the behavior (numerically is probably the easiest method) for these cases when $\phi(x) - y < 0$ and see what changes and what stays the same.

You may give yourself full credit for self-grades if you checked the behavior for a single value in each case rather than in general.

(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.**

Solution:

$$\mathbf{W}_i^{(1)} x + \mathbf{b}_i = 0$$

$$x = -\frac{\mathbf{b}_i}{\mathbf{W}_i^{(1)}}$$

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

Solution: The new location after an SGD update is

$$e_i' = \frac{-\mathbf{b}_i + \lambda \frac{\partial \ell}{\partial \mathbf{b}_i}}{\mathbf{W}_i^{(1)} - \lambda \frac{\partial \ell}{\partial \mathbf{W}_i^{(1)}}}.$$

We have

$$\frac{\partial \ell}{\partial \mathbf{W}^{(1)}} = x \left(\mathbf{W}^{(2)} \Phi \left(\mathbf{W}^{(1)} x + \mathbf{b} \right) - y \right) \mathbf{W}^{(2)} \frac{\partial \Phi}{\partial \left(\mathbf{W}^{(1)} x + \mathbf{b} \right)}$$

For the ReLU activation,

$$\frac{\partial \Phi}{\partial \left(\mathbf{W}^{(1)}x+\mathbf{b}\right)} = \operatorname{diag}\left(\mathbb{1}\left(\mathbf{W}^{(1)}x+\mathbf{b}>0\right)\right) = \begin{bmatrix} \mathbb{1}\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)_1 & 0 & \dots & 0 \\ 0 & \mathbb{1}\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbb{1}\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)_n \end{bmatrix}$$

where $\mathbb{1}(\cdot)$ is an indicator variable for whether each element of the contents is true or false. Since we are only interested in the i'th index of this gradient, we can simplify $\mathbf{W}^{(2)}\frac{\partial \Phi}{\partial(\cdot)}$ to

$$\begin{cases} \mathbf{W}_i^{(2)}, & \left(\mathbf{W}^{(1)}x+\mathbf{b}\right)_i > 0 \\ 0, & \text{else} \end{cases}$$

Therefore,

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{W}_i^{(1)}} &= \begin{cases} x\left(\mathbf{W}^{(2)}\Phi\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)-y\right)\mathbf{W}_i^{(2)}, & \mathbf{W}_i^{(1)}x+\mathbf{b}_i>0 \\ 0, & \text{else} \end{cases} \\ \frac{\partial \ell}{\partial \mathbf{b}_i} &= \left(\mathbf{W}^{(2)}\Phi\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)-y\right)\mathbf{W}_i^{(2)}\frac{\partial \Phi_i}{\partial\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)} \\ &= \begin{cases} \left(\mathbf{W}^{(2)}\Phi\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)-y\right)\mathbf{W}_i^{(2)}, & \mathbf{W}_i^{(1)}x+\mathbf{b}_i>0 \\ 0, & \text{else} \end{cases} \end{aligned}$$

Putting everything together,

$$\begin{aligned} e_i' &= \begin{cases} \frac{-\mathbf{b}_i+\lambda\left(\mathbf{W}^{(2)}\Phi\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)-y\right)\mathbf{W}_i^{(2)}}{\mathbf{W}_i^{(1)}-\lambda x\left(\mathbf{W}^{(2)}\Phi\left(\mathbf{W}^{(1)}x+\mathbf{b}\right)-y\right)\mathbf{W}_i^{(2)}}, & \mathbf{W}_i^{(1)}x+\mathbf{b}_i>0 \\ e_i, & \text{else} \end{cases} \\ &= \begin{cases} \frac{-\mathbf{b}_i+\lambda\left(\hat{f}(x)-y\right)\mathbf{W}_i^{(2)}}{\mathbf{W}_i^{(1)}-\lambda x\left(\hat{f}(x)-y\right)\mathbf{W}_i^{(2)}}, & \mathbf{W}_i^{(1)}x+\mathbf{b}_i>0 \\ e_i, & \text{else} \end{cases} \end{aligned}$$

We can still predict the change in each component ReLU function $\Phi_i(x)$ as long as we know $\hat{f}(x)-y$, $\mathbf{W}_i^{(1)}$, $\mathbf{b}_i$, and $\mathbf{W}_i^{(2)}$.

6. Coding Fully Connected Networks

In this coding assignment, you will be building a fully-connected neural network from scratch using NumPy.

Download the .zip file with the starter code and get it to work in either Google Colab or a local Conda environment.

Please submit the `.pdf` export of only the jupyter notebook when it is completed as a part of your submission. In addition, please answer the following question:

(a) **Did you notice anything about the comparative difficulty of training the three-layer net vs training the five layer net?**

Solution: Training a five-layer neural network is more difficult and more sensitive to hyperparameters (initialization scale and learning rate).

7. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student!
We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) **What sources (if any) did you use as you worked through the homework?**

(b) **If you worked with someone on this homework, who did you work with?**
List names and student ID's. (In case of homework party, you can also just describe the group.)

(c) **Roughly how many total hours did you work on this homework?**

Contributors:

- Saagar Sanghavi.
- Alexander Tsigler.
- Anant Sahai.
- Jane Yu.
- Philipp Moritz.
- Soroush Nasiriany.
- Josh Sanz.
- Linyuan Gong.
- Luke Jaffe.

