EECS 182 Deep Neural Networks
Fall 2022 Anant Sahai

Homework 1

This homework is due on Saturday, September 10, 2022, at 10:59PM.

1. 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.

2. Bias-Variance Tradeoff Review

(a) Suppose we have a randomly sampled training set $\mathcal{D}$ (drawn independently of our test data), and we select an estimator denoted $\theta = \hat{\theta}(\mathcal{D})$ (for example, via empirical risk minimization). **Show that we can decompose our expected mean squared error for a test input x into a bias, a variance and an irreducible error term as below:**

$$\mathbb{E}_{Y\sim p(y|x),\mathcal{D}}[(Y - f_{\hat{\theta}(\mathcal{D})}(x))^2] = \mathrm{Var}(f_{\hat{\theta}(\mathcal{D})}(x)) + \mathrm{Bias}(f_{\hat{\theta}(\mathcal{D})}(x))^2 + \sigma^2$$

You may find it helpful to recall the formulaic definitions of Variance and Bias, reproduced for you below:

$$\mathrm{Var}(f_{\hat{\theta}(\mathcal{D})}(x)) = \mathbb{E}_{\mathcal{D}}\left[(f_{\hat{\theta}(\mathcal{D})}(x) - \mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)])^2\right]$$
$$\mathrm{Bias}(f_{\hat{\theta}(\mathcal{D})}(x)) = \mathbb{E}_{Y\sim p(Y|x),\mathcal{D}}[f_{\hat{\theta}(\mathcal{D})}(x) - Y]$$

Solution: For simplicity of notation, let $\mathbb{E}[\cdot]$ denote $\mathbb{E}_{Y\sim p(y|x),\mathcal{D}}[\cdot]$

$$\begin{aligned}\mathbb{E}[(Y - f_{\hat{\theta}(\mathcal{D})}(x))^2] &= \mathbb{E}[(Y - f_{\hat{\theta}(\mathcal{D})}(x))^2] \\ &= \mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)^2 - 2Y f_{\hat{\theta}(\mathcal{D})}(x) + Y^2]\end{aligned}$$

By independence of Y and $\mathcal{D}$ and linearity of expectation,

$$\mathbb{E}[(Y - f_{\hat{\theta}(\mathcal{D})}(x))^2] = \mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)^2] - 2\mathbb{E}[Y]\mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)] + \mathbb{E}[Y^2]$$

Noting the definition of variance,

$$\begin{aligned}\mathbb{E}[(Y - f_{\hat{\theta}(\mathcal{D})}(x))^2] &= \mathrm{Var}(f_{\hat{\theta}(\mathcal{D})}(x)) + \mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)]^2 - 2\mathbb{E}[Y]\mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)] + \mathbb{E}[Y^2] \\ &= \mathrm{Var}(f_{\hat{\theta}(\mathcal{D})}(x)) + (\mathbb{E}[f_{\hat{\theta}(\mathcal{D})}(x)] - \mathbb{E}[Y])^2 + \mathrm{Var}(Y|X = x) \\ &= \mathrm{Var}(f_{\hat{\theta}(\mathcal{D})}(x)) + \mathrm{Bias}(f_{\hat{\theta}(\mathcal{D})}(x))^2 + \mathrm{Var}(Y|X = x)\end{aligned}$$

The conditional variance $\mathrm{Var}(Y|x)$, which we will denote σ^2 captures the irreducible error that will be incurred no matter what learner $\hat{\theta}$ we use.

(b) Suppose our training dataset consists of $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where the only randomness is coming from the label vector $Y = \mathbf{X}\theta^* + \boldsymbol{\varepsilon}$ where θ^* is the true underlying linear model and each noise variable ε_i is i.i.d. with zero mean and variance 1. We use ordinary least squares to estimate a $\hat{\theta}$ from this data. **Calculate the bias and covariance of the $\hat{\theta}$ estimate and use that to compute the bias and variance of the prediction at particular test inputs x.** Recall that the OLS solution is given by

$$\hat{\theta} = (X^\top X)^{-1} X^\top Y,$$

where $X \in \mathbb{R}^{n\times d}$ is our (nonrandom) data matrix, $Y \in \mathbb{R}^n$ is the (random) vector of training targets. For simplicity, assume that $X^\top X$ is diagonal.

Solution: We first compute the bias of $\hat{\theta}$. Recalling that we have $Y = X\theta + \varepsilon$ for a noise vector ε, we then have

$$\begin{aligned}
\mathbb{E}[\hat{\theta}] &= \mathbb{E}[(X^\top X)^{-1} X^\top (X\theta + \varepsilon)] \\
&= \mathbb{E}[\theta + (X^\top X)^{-1} X^\top \varepsilon] \\
&= \theta + (X^\top X)^{-1} X^\top \mathbb{E}[\varepsilon] \\
&= \theta && \varepsilon \text{ has 0 mean.}
\end{aligned}$$

We thus see that the OLS estimator $\hat{\theta}$ is an unbiased estimator of the true parameter θ. Considering the bias of our estimate at a particular test input x, we see that our prediction is also unbiased.

$$\mathbb{E}[x^\top \hat{\theta} - x^\top \theta - \epsilon] = 0$$

Next, we compute the variance of $\hat{\theta}$, and we will proceed by first computing the covariance of $\hat{\theta}$,

$$\begin{aligned}
\mathbb{E}[(\hat{\theta} - \theta)(\hat{\theta} - \theta)^\top] &= \mathbb{E}[(X^\top X)^{-1} X^\top \varepsilon \varepsilon^\top ((X^\top X)^{-1} X^\top)^\top] \\
&= (X^\top X)^{-1} X^\top \mathbb{E}[\varepsilon \varepsilon^\top] X ((X^\top X)^{-1})^\top \\
&= (X^\top X)^{-1} X^\top I_n ((X^\top X)^{-1} X^\top)^\top && \text{noise variables are iid} \\
&= (X^\top X)^{-1} X^\top X ((X^\top X)^{-1})^\top \\
&= (X^\top X)^{-1}.
\end{aligned}$$

Now for a particular test input x, we can compute the variance

$$\begin{aligned}
\mathrm{Var}[x^\top (\hat{\theta} - \theta)] &= \mathbb{E}[x^\top (\hat{\theta} - \theta)(\hat{\theta} - \theta) x] \\
&= x^\top (X^\top X)^{-1} x.
\end{aligned}$$

Unlike the bias, we see that the variance of our estimate depends on which test input x we're measuring the risk for. For simplicity, suppose $X^\top X$ were a diagonal matrix (we could have applied an orthogonal transformation to achieve this) with sorted entries $\sigma_1^2 \geq \sigma_2^2 \ldots \geq \sigma_d^2$ (corresponding to the data variances in each dimension). Now we can easily compute the variance as $\sum_{i=1}^d x_i^2 / \sigma_i^2$, and we see that in directions where σ_i is close to 0 (which means there is very little variance in the data in this dimension), the variance of our estimate can explode (and thus our risk as well).

3. The Many Interpretations of Ridge Regression

(a) Recall that ridge regression can be understood as the unconstrained optimization problem

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

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a design matrix (data), and $\mathbf{y} \in \mathbb{R}^n$ is the target vector of measurement values.

One way to interpret "ridge regression" is as the ordinary least squares for an augmented data set — i.e. adding a bunch of fake data points to our data. Consider the following augmented measurement vector $\hat{\mathbf{y}}$ and data matrix $\hat{\mathbf{X}}$:

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

where $\mathbf{0}_d$ is the zero vector in $\mathbb{R}^d$ and $\mathbf{I}_d \in \mathbb{R}^{d\times d}$ is the identity matrix. **Show that the optimization problem** $\min_w \|\hat{\mathbf{y}} - \hat{\mathbf{X}}\mathbf{w}\|_2^2$ **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

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

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

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

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

(b) Perhaps more surprisingly, one can achieve the same effect as in the previous part by adding fake features to each data point instead of adding fake data points. Let's construct the augmented design matrix in the following way:

$$\hat{\mathbf{X}} = [\mathbf{X}\ \alpha\mathbf{I_n}]$$

i.e. we stack $\mathbf{X}$ with $\alpha\mathbf{I}_n$ horizontally. Here α is a scalar multiplier. 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 $\hat{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}$. Consider the following problem:

$$\min_{\boldsymbol{\eta}} \|\boldsymbol{\eta}\|_2^2 \text{ s.t. } \hat{\mathbf{X}}\boldsymbol{\eta} = \mathbf{y}. \tag{2}$$

Find the α **such that if** $\boldsymbol{\eta}^*$ **is the minimizer of** (2)**, then the first** d **coordinates of** $\boldsymbol{\eta}^*$ **form the minimizer of** (1)**.**

Can you interpret what the final n **coordinates of** $\boldsymbol{\eta}^*$ **represent?**

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

$$\min_{\mathbf{w},\boldsymbol{\xi}} \|\mathbf{w}\|_2^2 + \|\boldsymbol{\xi}\|_2^2 \text{ s.t. } \mathbf{X}\mathbf{w} + \alpha\boldsymbol{\xi} = \mathbf{y}.$$

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

$$\min_{\mathbf{w},\boldsymbol{\xi}} \|\mathbf{w}\|_2^2 + \frac{1}{\alpha^2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

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:

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

which is equivalent to (1) for $\alpha = \sqrt{\lambda}$.

The last n rows of η^* therefore represent a scaled version of the classic OLS residual.

Notice that by taking the limit $\alpha \to 0$, we would just get OLS.

(c) 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

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

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**

$$\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \alpha^2\mathbf{I})^{-1}\mathbf{y}$$

Then, show that with the α you identified in the previous part, this is equivalent to the standard formula for Ridge Regression ($\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$)

(Hint: For the last part, it might be useful to lookup Kernel Ridge Regression in your machine learning notes or textbook.)

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

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

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

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

Looking at just the top d terms, we see $\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \alpha^2\mathbf{I})^{-1}\mathbf{y}$ as desired.

Now, let's show the two forms of Ridge Regression are equivalent. That is, let's plug in $\alpha = \sqrt{\lambda}$ and show

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

With the expression above, let's left-multiply both sides by $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ and right-multiply both sides by $(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})$ to get

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

And distributing the matrix multiplication we have

$$\mathbf{X}^T\mathbf{X}\mathbf{X}^T + \lambda\mathbf{X}^T = \mathbf{X}^T\mathbf{X}\mathbf{X}^T + \lambda\mathbf{X}^T$$

which we can see is always true as desired.

(d) Suppose the SVD of $\mathbf{X}$ is $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$. Let's make a change of coordinates in the feature space, such that $\mathbf{V}$ becomes identity: $\mathbf{X}_{\text{new}} = \mathbf{X}\mathbf{V}$ and $\mathbf{w}_{\text{new}} = \mathbf{V}^\top\mathbf{w}$. Denote the the solution to ridge regression (1) in these new coordinates as $\hat{\mathbf{w}}_{\text{new}}$. **Show that the i-th coordinate of $\hat{\mathbf{w}}_{\text{new}}$ can be obtained from the corresponding coordinate of $\mathbf{U}^\top\mathbf{y}$ by multiplication by $\frac{\sigma_i}{\sigma_i^2+\lambda}$, where σ_i is the i-th singular value of $\mathbf{X}$ (or zero if i is greater than the rank of X.)**

Solution: We know that $\mathbf{w}_{\text{new}}$ is the solution of the minimization procedure

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

We would like to have $\mathbf{\Sigma}$ instead of $\mathbf{X}_{\text{new}}$ to make use of the singular values of $\mathbf{X}$. Note that $\mathbf{X}_{\text{new}} = \mathbf{U}\mathbf{\Sigma}$, so we just need to multiply $\mathbf{X}_{\text{new}}$ by $\mathbf{U}^T$ from the left to get $\mathbf{\Sigma}$. The only question is what will happen with our optimization procedure. Recall that multiplication by an orthonormal (unitary) matrix does not change Euclidean norm, thus

$$\|\mathbf{X}_{\text{new}}\mathbf{w}_{\text{new}} - \mathbf{y}\|_2^2 = \|\mathbf{U}^\top\mathbf{X}_{\text{new}}\mathbf{w}_{\text{new}} - \mathbf{U}^\top\mathbf{y}\|_2^2 = \|\mathbf{\Sigma}_{\text{new}}\mathbf{w}_{\text{new}} - \mathbf{U}^\top\mathbf{y}\|_2^2.$$

Therefore, our optimization may be rewritten as

$$\min_{\mathbf{w}_{\text{new}}} \|\mathbf{\Sigma}_{\text{new}}\mathbf{w}_{\text{new}} - \mathbf{U}^\top\mathbf{y}\|_2^2 + \lambda\|\mathbf{w}_{\text{new}}\|_2^2.$$

Now, we can rewrite that as

$$\min_{\mathbf{w}_{\text{new}}} \left[\sum_{i=1}^{\text{rank}(\mathbf{X})} \left(\sigma_i(\mathbf{w}_{\text{new}})_i - (\mathbf{U}^\top\mathbf{y})_i\right)^2 + \lambda\sum_{i=1}^{d}(\mathbf{w}_{\text{new}})_i^2 \right].$$

We see that the target function is a sum of functions, each of which only depends on one coordinate of $\mathbf{w}_{\text{new}}$. Thus, we can minimize each of those terms separately in it's own argument. We see that for $i > \text{rank}(\mathbf{X})$

$$\underset{(\mathbf{w}_{\text{new}})_i}{\text{argmin}}(\mathbf{w}_{\text{new}})_i^2 = 0$$

and for $i \leq \text{rank}(\mathbf{X})$

$$\underset{(\mathbf{w}_{\text{new}})_i}{\text{argmin}} \left[\left(\sigma_i(\mathbf{w}_{\text{new}})_i - (\mathbf{U}^\top\mathbf{y})_i\right)^2 + \lambda(\mathbf{w}_{\text{new}})_i^2 \right] = \frac{\sigma_i}{\sigma_i^2 + \lambda}(\mathbf{U}^\top\mathbf{y})_i.$$

The remaining parts (e-h) of this question are optional.

(e) One reason why we might want to have small weights $\mathbf{w}$ has to do with the sensitivity of the predictor to its input. Let $\mathbf{x}$ be a d-dimensional list of features corresponding to a new test point. Our predictor is $\mathbf{w}^\top \mathbf{x}$. **What is an upper bound on how much our prediction could change if we added noise $\boldsymbol{\epsilon} \in \mathbb{R}^d$ to a test point's features $\mathbf{x}$?**

Solution: The change in prediction is given by

$$|\mathbf{w}^\top(\mathbf{x}+\boldsymbol{\epsilon}) - \mathbf{w}^\top \mathbf{x}| = |\mathbf{w}^\top \boldsymbol{\epsilon}| \leq \|\mathbf{w}\|_2 \|\boldsymbol{\epsilon}\|_2 \tag{3}$$

where the second step is a result of the Cauchy-Schwarz inequality. To build intuition as to why this is true, you can divide both sides by $\|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2$ which gives you

$$\left| \langle \frac{\mathbf{w}}{\|\mathbf{w}\|_2}, \frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|_2} \rangle \right| \leq 1 \tag{4}$$

In words, this inequality is saying that if you project a unit vector $\frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|_2}$ onto some direction $\frac{\mathbf{w}}{\|\mathbf{w}\|_2}$, the resulting length must be less than or equal to 1. The prediction will thus vary from the original prediction by at most $\|\mathbf{w}\|_2\|\boldsymbol{\epsilon}\|_2$.

Trying to keep the learned weights $\mathbf{w}$ small is therefore a way to keep the learned function from having too strong of a dependence on small changes in the features.

(f) We know that the solution to ridge regression (1) is given by $\hat{\mathbf{w}}_\mathbf{r} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$. **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 $(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}$ converges to the zero matrix, and so we have $\mathbf{w} = \mathbf{0}$.

(g) Another advantage of ridge regression can be seen for under-determined systems. Say we have the data drawn from a $d = 5$ parameter model, but only have $n = 4$ training samples of it, i.e. $\mathbf{X} \in \mathbb{R}^{4\times 5}$. Now this is clearly an underdetermined system, since $n < d$. **Show that ridge regression with $\lambda > 0$ results in a unique solution, whereas ordinary least squares can have an infinite number of solutions.**

[Hint: To make this point, it may be helpful to consider $\mathbf{w} = \mathbf{w}_0 + \mathbf{w}^*$ where $\mathbf{w}_0$ is in the null space of $\mathbf{X}$ and $\mathbf{w}^*$ is a solution.]

[Alternative Hint: You might want to consider (2) as the way to interpret ridge regression.]

Solution: First, we will follow the first hint and show that ridge regression always leads to a unique solution. We know that the minimizer is given by

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

We also know that the eigenvalues of the matrix $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ are all at least λ, and so the matrix is invertible, thus leading to a unique solution.

The alternative hint lets us see uniqueness immediately since in (2), we have a full row rank matrix $\hat{\mathbf{X}}$ because the $\sqrt{\lambda}\mathbf{I_n}$ is clearly full rank and $\hat{\mathbf{X}}$ is a wide matrix by construction. Consequently, the solution for $\boldsymbol{\eta}$ is given by the classic Moore-Penrose pseudo-inverse $\hat{\mathbf{X}}^\top(\hat{\mathbf{X}}\hat{\mathbf{X}}^\top)^{-1}$ which is unique since $(\hat{\mathbf{X}}\hat{\mathbf{X}}^\top)$ is invertible.

For ordinary least squares, let us assume that $\mathbf{w'}$ minimizes $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$, i.e., $\|\mathbf{X}\mathbf{w'} - \mathbf{y}\|_2 \leq \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$ for all $\mathbf{w} \in \mathbb{R}^d$.

Since $d > n$, the matrix $\mathbf{X}$ has a non-trivial nullspace. We can take any vector $\mathbf{w_0}$ in the nullspace of $\mathbf{X}$ and consider the new vector $\mathbf{w''}(\alpha) = \mathbf{w'} + \alpha\mathbf{w_0}$.

Notice that $\|\mathbf{X}\mathbf{w''}(\alpha) - \mathbf{y}\|_2 = \|\mathbf{X}\mathbf{w'} - \mathbf{y}\|_2 \leq \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2$ for all $\mathbf{w} \in \mathbb{R}^d$ because $\mathbf{w_0}$ is in the null space of $\mathbf{X}$. Thus, the vector $\mathbf{w''}(\alpha)$ is a minimizer for any choice of α. We have thus shown that the least squares problem has an infinite number of solutions.

(h) For the previous part, **what will the answer be if you take the limit $\lambda \to 0$ for ridge regression?** *[Hint: You might want to consider (2) as the way to interpret ridge regression.]*

Solution:

Plugging the singular value decomposition $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top$ into the solution of the ridge regression, we get

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

and for $\lambda \to 0$ this converges to the unique solution $\mathbf{w}^* = \mathbf{V}\Sigma^{-1}\mathbf{U}^\top\mathbf{y}$, also called the *minimum norm solution*. (Notice here that Σ^{-1} has the obvious definition of just involving the relevant square matrix of nonzero singular values.)

If we wanted to use the augmented features approach to ridge regularization, the answer is immediate since what happens is that the constraint in (2) just becomes the $\mathbf{X}\mathbf{w} = \mathbf{y}$ constraint when $\lambda \to 0$. So we get the minimum norm solution to this.

4. General Case Tikhonov Regularization

Consider the optimization problem:

$$\min_{\mathbf{x}} ||W_1(A\mathbf{x} - \mathbf{b})||_2^2 + ||W_2(\mathbf{x} - \mathbf{c})||_2^2$$

Where W_1, A, and W_2 are matrices and $\mathbf{x}$, $\mathbf{b}$ and $\mathbf{c}$ are vectors. W_1 can be viewed as a generic weighting of the residuals and W_2 along with c can be viewed as a generic weighting of the parameters.

(a) Solve this optimization problem manually by expanding it out as matrix-vector products, setting the gradient to $\mathbf{0}$, and solving for $\mathbf{x}$.

Solution: Expand our objective function:

$$f(\mathbf{x}) = (A\mathbf{x} - \mathbf{b})^T W_1^T W_1 (A\mathbf{x} - \mathbf{b}) + (\mathbf{x} - \mathbf{c})^T W_2^T W_2 (\mathbf{x} - \mathbf{c})$$

$$f(\mathbf{x}) = \mathbf{x}^T A^T W_1^T W_1 A\mathbf{x} - 2\mathbf{b}^T W_1^T W_1 A\mathbf{x} + \mathbf{b}^T W_1^T W_1 \mathbf{b} + \mathbf{x}^T W_2^T W_2 \mathbf{x} - 2\mathbf{c}^T W_2^T W_2 \mathbf{x} + \mathbf{c}^T W_2^T W_2 \mathbf{c}$$

Now take gradients and set it to $\mathbf{0}$:

$$\nabla f = 2A^T W_1^T W_1 A\mathbf{x} - 2A^T W_1^T W_1 \mathbf{b} + 2W_2^T W_2 \mathbf{x} - 2W_2^T W_2 \mathbf{c} = 0$$

Isolating the $\mathbf{x}$ terms on one side, we have:

$$(A^T W_1^T W_1 A + W_2^T W_2)\mathbf{x} = A^T W_1^T W_1 \mathbf{b} + W_2^T W_2 \mathbf{c}$$

So we can solve to get

$$\mathbf{x} = (A^T W_1^T W_1 A + W_2^T W_2)^{-1}(A^T W_1^T W_1 \mathbf{b} + W_2^T W_2 \mathbf{c})$$

(b) Construct an appropriate matrix $\mathbf{C}$ and vector $\mathbf{d}$ that allows you to rewrite this problem as

$$\min_x \|C\mathbf{x} - \mathbf{d}\|^2$$

and use the OLS solution ($\mathbf{x}^* = (C^TC)^{-1}C^T\mathbf{d}$ to solve. Confirm your answer is in agreement with the previous part.

Solution: We can rewrite our problem in least-squares form using

$$C = \begin{bmatrix} W_1A \\ W_2 \end{bmatrix}, \text{ and } \mathbf{d} = \begin{bmatrix} W_1\mathbf{b} \\ W_2\mathbf{c} \end{bmatrix}$$

Now, using least squares solution and solving, we get

$$\mathbf{x}^* = (C^TC)^{-1}C^T\mathbf{d} = (\begin{bmatrix} W_1A \\ W_2 \end{bmatrix}^\top \begin{bmatrix} W_1A \\ W_2 \end{bmatrix})^{-1} \begin{bmatrix} W_1A \\ W_2 \end{bmatrix}^\top \begin{bmatrix} W_1\mathbf{b} \\ W_2\mathbf{c} \end{bmatrix}$$

$$= (\begin{bmatrix} A^TW_1^T & W_2^T \end{bmatrix} \begin{bmatrix} W_1A \\ W_2 \end{bmatrix})^{-1} \begin{bmatrix} A^TW_1^T & W_2^T \end{bmatrix} \begin{bmatrix} W_1\mathbf{b} \\ W_2\mathbf{c} \end{bmatrix}$$

$$\mathbf{x}^* = (A^TW_1^TW_1A + W_2^TW_2)^{-1}(A^TW_1^TW_1\mathbf{b} + W_2^TW_2\mathbf{c})$$

Which is the same as the previous part, as desired.

(c) Under which case would this reduce to the simple case of ridge regression that you've seen in the previous problem $\mathbf{x}^* = (A^TA + \lambda I)^{-1}A^Tb$?

Solution: This reduces to ridge regression when $W_1 = I$, $W_2 = \sqrt{\lambda}I$, and $\mathbf{c} = \mathbf{0}$. You can see this in both the optimization problem and the result.

5. System ID for Continuous Systems: Understanding how PyTorch can be used to estimate underlying dynamics given observed samples of a trajectory

(a) Suppose we have a system that we believe is of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \lambda x(t) + bu(t) \tag{5}$$

where $x(t) \in \mathbb{R}$ and $u(t) \in \mathbb{R}$ is a scalar input.
Given an initial condition $x(t_0)$, and that $u(t)$ is some constant input $\bar{u}$ over the interval $[t_0, t_f)$, then for all $t \in [t_0, t_f)$, we know that this differential equation eq. (5) has the unique solution

$$x(t) = x(t_0)e^{\lambda(t-t_0)} + \frac{e^{\lambda(t-t_0)} - 1}{\lambda}b\bar{u}. \tag{6}$$

Assume that we record the state at known times $\tau_0, \tau_1, \ldots, \tau_n$ as having corresponding state values $x_0 = x(\tau_0), x_1 = x(\tau_1), \ldots, x_n = x(\tau_n)$. The continuous-time input is known to be piecewise constant $u(t) = u_i$ for $t \in [\tau_i, \tau_{i+1})$, where we know the sequence of inputs $u_0, u_1, \ldots, u_{n-1}$.
We now want to formulate this as a system ID question by relating the unknown parameters λ, b to the data we have. However, the relationship between the parameters and the data we collected is now

non-linear. **For the data point x_{i+1}, use eq. (6) to write out how x_{i+1} should be related to λ and b in the form**

$$x_{i+1} = f(\lambda, x_i, \tau_i, \tau_{i+1}) + bg(\lambda, x_i, u_i, \tau_i, \tau_{i+1}). \tag{7}$$

What are the functions f and g?

Solution: We directly use (6) by plugging in $t = \tau_{i+1}$ and $t_0 = \tau_i$. We can use this formula as the input $u(t)$ is a constant u_i throughout this interval.

$$f(\lambda, x_i, \tau_i, \tau_{i+1}) = x_i e^{\lambda(\tau_{i+1}-\tau_i)} \tag{8}$$

$$g(\lambda, x_i, u_i, \tau_i, \tau_{i+1}) = \frac{u_i}{\lambda}(e^{\lambda(\tau_{i+1}-\tau_i)} - 1) \tag{9}$$

(b) The previous part gave rise to a sequence of n equations of the form eq. (7). Because of observation noise and imperfection in our model, we are going to assume that these equations hold only approximately and hope to find values for the two parameters λ, b that minimize the cost function:

$$c(\lambda, b) = \sum_{i=0}^{n-1} \ell\left(x_{i+1}, f(\lambda, x_i, \tau_i, \tau_{i+1}) + bg(\lambda, x_i, u_i, \tau_i, \tau_{i+1})\right) \tag{10}$$

where $f(\lambda, x, \sigma, \tau)$ and $g(\lambda, x, u, \sigma, \tau)$ are given nonlinear scalar functions, and $\ell(s, p)$ is a loss function that penalizes how far the prediction p is from the measured state s. For example, you could use squared loss $\ell(s, p) = (s - p)^2$.

To find the best possible λ_*, b_*, you observe that you want to solve the nonlinear system of equations:

$$\frac{\partial}{\partial \lambda} c(\lambda, b)\Big|_{\lambda_*, b_*} = 0 \tag{11}$$

$$\frac{\partial}{\partial b} c(\lambda, b)\Big|_{\lambda_*, b_*} = 0 \tag{12}$$

and decide to do so using Newton's method starting with an initial guess λ_0, b_0 and linearizing the system of equations eq. (11) and eq. (12) to get a system of linear equations to solve at each step.

The system of linear equations at each iteration $j + 1$ can be expressed in vector form as:

$$A \begin{bmatrix} \lambda - \lambda_j \\ b - b_j \end{bmatrix} = \mathbf{y}. \tag{13}$$

What are the entries of the matrix A and the vector $\mathbf{y}$ in terms of the appropriate partial derivatives of $c(\lambda, b)$ evaluated at the appropriate arguments?

Assume that you can use PyTorch to compute whatever derivatives of $c(\lambda, b)$ that you want — all given functions are sufficiently differentiable. You don't have to take the derivatives by hand. You just need to tell us what derivatives and what arguments to evaluate them at.

Solution: We want to solve a nonlinear vector equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. Linearizing this equation gives us

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*) = \mathbf{0} \tag{14}$$

We want to solve for $\mathbf{x}$ as a function of our current guess $\mathbf{x}^*$, which means we want to solve

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) = -\mathbf{f}(\mathbf{x}^*) \tag{15}$$

For our objective, we want

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \frac{\partial c(\lambda_j, b_j)}{\partial \lambda} \\ \frac{\partial c(\lambda_j, b_j)}{\partial b} \end{bmatrix} = \mathbf{0} \tag{16}$$

Then during each Newton's iteration, we want to solve the linear system of equations

$$\begin{bmatrix} \frac{\partial^2 c}{\partial \lambda \partial \lambda}(\lambda_j, b_j) & \frac{\partial^2 c}{\partial \lambda \partial b}(\lambda_j, b_j) \\ \frac{\partial^2 c}{\partial b \partial \lambda}(\lambda_j, b_j) & \frac{\partial^2 c}{\partial b \partial b}(\lambda_j, b_j) \end{bmatrix} \begin{bmatrix} \lambda - \lambda_j \\ b - b_j \end{bmatrix} = \begin{bmatrix} -\frac{\partial c}{\partial \lambda}(\lambda_j, b_j) \\ -\frac{\partial c}{\partial b}(\lambda_j, b_j) \end{bmatrix} \tag{17}$$

$$\frac{\partial^2 c}{\partial \mathbf{x} \partial \mathbf{x}}(\lambda_j, b_j) \begin{bmatrix} \lambda - \lambda_j \\ b - b_j \end{bmatrix} = -\frac{\partial c}{\partial \mathbf{x}}^\top (\lambda_j, b_j) \tag{18}$$

Thus the matrix A is the Hessian of $c(\lambda, b)$ evaluated at λ_j, b_j, and the vector $\mathbf{y}$ is the transpose of the derivative of $c(\lambda, b)$ evaluated at λ_j, b_j. You don't need to see this connection and full credit is given if you just write out the partial derivative entries of A and $\mathbf{y}$.

6. Movie Ratings with Missing Entries: understanding validation in a way that points towards self-supervision

In a matrix R, you have users' movie ratings. However, not all users watched all the movies.

$$R = \left[\begin{array}{cccc|cc} 0.50 & 0.00 & 0.50 & 0.50 & 0.20 & 1.0 \\ 0.60 & 0.20 & 0.40 & 0.50 & ? & ? \\ 0.50 & 0.50 & 0.00 & 0.25 & 0.60 & 1.0 \\ 0.60 & 0.10 & 0.50 & 0.55 & ? & ? \\ \hline 1.00 & 0.40 & 0.60 & ? & ? & ? \end{array}\right] \tag{19}$$

where the element at the ith row and jth column indicates the rating of movie i by user j. A "?" means that there's no rating for that movie.

Our goal is to predict ratings for the missing entries, so we can recommend movies to users. In order to do this, you want to find the hidden goodness vectors for the movies, and the hidden sensitivity vectors of the users. However, due to missing entries, it is not possible to run an SVD on the entire rating matrix R.

It turns out that we have a submatrix R' in R that does not have any missing entries.

$$R' = \begin{bmatrix} 0.50 & 0.00 & 0.50 & 0.50 \\ 0.60 & 0.20 & 0.40 & 0.50 \\ 0.50 & 0.50 & 0.00 & 0.25 \\ 0.60 & 0.10 & 0.50 & 0.55 \end{bmatrix} \tag{20}$$

We provide a decomposition of this matrix:

$$R' = \underbrace{\begin{bmatrix} 0.5 & 0.0 \\ 0.4 & 0.2 \\ 0.0 & 0.5 \\ 0.5 & 0.1 \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} 4.0 & 0.0 \\ 0.0 & 2.0 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 0.25 & 0.0 & 0.25 & 0.25 \\ 0.5 & 0.5 & 0.0 & 0.25 \end{bmatrix}}_{S} \tag{21}$$

where the ith row of the matrix G represents the goodness row vector $\mathbf{g}_i^\top$ of the movie i, the jth column of the matrix S represents the sensitivity vector $\mathbf{s}_j$ of user j, and each diagonal entry of the matrix D shows the weight each attribute has in determining the rating of a movie by a user.

NOTE: This decomposition in eq. (21) is not an SVD; G and S do not have orthonormal vectors.

(a) Suppose $\mathbf{s}_6$ (the sensitivity vector of user 6) is:

$$\mathbf{s}_6 = \begin{bmatrix} 0.5 \\ 1.0 \end{bmatrix} \tag{22}$$

Use this to **estimate the rating of movie 2 as rated by user 6.** (Show work that uses $\mathbf{s}_6$. Unjustified answers will get no credit.)

Solution: This rating is in R_{26}. From the structure of R', if we generalize the sensitivity model, then this can be calculated with $\mathbf{g}_2^\top D \mathbf{s}_6$.

$$\begin{aligned} R_{26} &= \mathbf{g}_2^\top D \mathbf{s}_6 = d_1 s_{61} g_{21} + d_2 s_{62} g_{22} && (23) \\ &= 4 \cdot 0.5 \cdot 0.4 + 2 \cdot 1.0 \cdot 0.2 && (24) \\ &= 1.2 && (25) \end{aligned}$$

where d_i is ith diagonal element of matrix D.

(b) For the 5th movie, we have three ratings and want to find two parameters of goodness. **Formulate a least squares problem $A\mathbf{g}_5 \approx \mathbf{b}$ to estimate $\mathbf{g}_5$ (goodness vector of movie 5).** You need to tell us A explicitly as a matrix with numerical entries. We give you that $\mathbf{b} = \begin{bmatrix} 1.00 \\ 0.40 \\ 0.60 \end{bmatrix}$ since those are the three ratings we know for this movie.

Solution: Considering the $R_{51}, \ldots, R_{54}$ in the 6th row of the ratings matrix R, we have

$$\begin{bmatrix} g_{51} & g_{52} \end{bmatrix} \underbrace{\begin{bmatrix} 4.0 & 0.0 \\ 0.0 & 2.0 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 0.25 & 0.0 & 0.25 & 0.25 \\ 0.5 & 0.5 & 0.0 & 0.25 \end{bmatrix}}_{S} = \begin{bmatrix} 1.00 & 0.40 & 0.60 & ? \end{bmatrix} \tag{26}$$

Removing the 4th column of S (since there is no corresponding equation) and transposing each side,

$$\underbrace{\begin{bmatrix} 0.25 & 0.5 \\ 0.0 & 0.5 \\ 0.25 & 0.0 \end{bmatrix}}_{S^\top} \underbrace{\begin{bmatrix} 4.0 & 0.0 \\ 0.0 & 2.0 \end{bmatrix}}_{D} \begin{bmatrix} g_{51} \\ g_{52} \end{bmatrix} = \begin{bmatrix} 1.00 \\ 0.40 \\ 0.60 \end{bmatrix} \tag{27}$$

Setting $A = S^\top D$ and $\mathbf{b}$ to be the ratings vector, we have a least squares problem

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|^2 \tag{28}$$

where

$$A = \begin{bmatrix} 1.0 & 1.0 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} g_{51} \\ g_{52} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1.00 \\ 0.40 \\ 0.60 \end{bmatrix} \tag{29}$$

(c) We now consider a ratings matrix R without missing entries (that is different from the previous R) where the matrix is partitioned into four blocks $R_{11}, R_{12}, R_{21}, R_{22}$ as below.

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \tag{30}$$

In order to find the optimal number of principal components, we compute a PCA model from the SVD of R_{11} with k principal components, with $k = 2, 3, 4, 5$. We then use the chosen components and the singular values of R_{11} together with the information in R_{12} and R_{21} to create an estimate $\widehat{R}_{22}$ for the held-out ratings in R_{22}. We can also use the first k terms of the SVD of R_{11} to reconstruct $\widehat{R}_{11}$ as the best rank-k approximation to R_{11}.

The training errors $\|R_{11} - \widehat{R}_{11}\|_F^2$ and validation errors $\|R_{22} - \widehat{R}_{22}\|_F^2$ for each candidate choice for k are given in the table below.

Select	k	Training error	Validation error
◯	1	1.428	3.104
◯	2	0.414	2.494
◯	3	0.093	0.462
◯	4	0.017	0.090
◯	5	0.011	0.132
◯	6	0.006	0.161

Find the optimal number of principal components k we should use. (No need to give any justification.)

Solution: As we increase the number of principal components k, we will observe training error decreasing, while validation error first decreases but increases at some point. The optimal k can be found by selecting k which gives minimum validation error, which is $k = 4$.

7. ReLU Elbow Update under SGD (Optional)

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.**

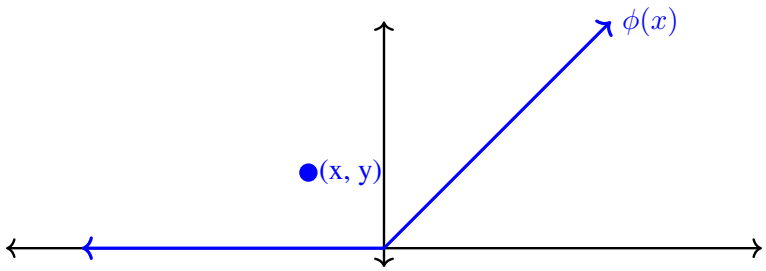
Additionally, draw and label $\phi(x)$, the elbow, and the qualitative changes to the slope and elbow after a gradient update to w and b. You should label the elbow location and a candidate (x, y) pair. Remember that the update for some parameter vector $\mathbf{p}$ and loss ℓ under SGD is

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

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

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

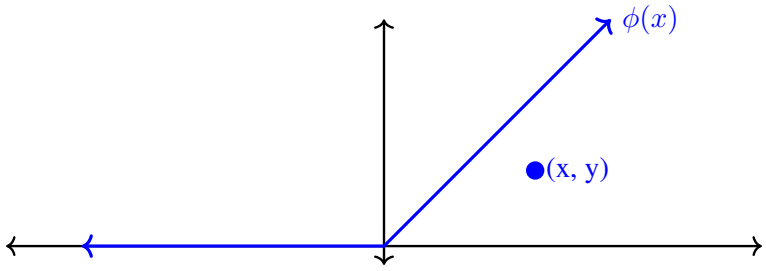
(i) No changes since the derivatives are 0.



(ii) The slope gets shallower and the elbow moves right.

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

Several numerical checks show the motion of the elbow.

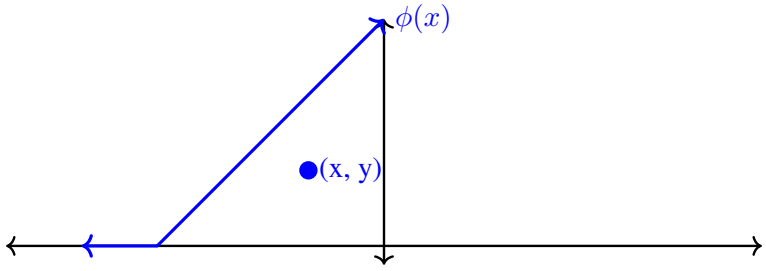


(iii) The slope gets steeper and the elbow moves right. Using the fact that $w, b > 0$ and $e < 0$ for this configuration,

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

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

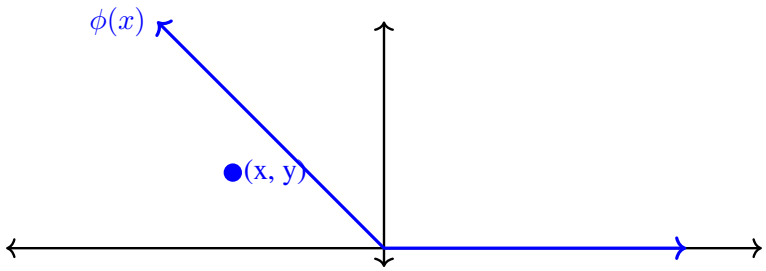
$$\therefore e' > e$$



(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,

$$\frac{\partial \Phi}{\partial \left(\mathbf{W}^{(1)} x + \mathbf{b} \right)} = \text{diag}\Big(\Big)(1 \left(\mathbf{W}^{(1)} x + \mathbf{b} > 0 \right) \Big)$$

where $1(\cdot)$ 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,

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

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

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 $\mathbf{W}_i^{(2)}$.

8. Coding Question: Fully Connected Networks

In this last question, you'll implement Fully Connected Networks with ReLU nonlinearity. Look at the Jupyter notebook FullyConnectedNets.ipynb in the hw1 folder to see what you have to do!

For this question, please submit a .zip file your completed work to the Gradescope assignment titled "HW 1 (Code)". No written portion.

9. 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? Write it down here where you'll need to remember it for the self-grade form.**

Contributors:

- Saagar Sanghavi.

- Brandon Trabucco.
- Alexander Tsigler.
- Anant Sahai.
- Jane Yu.
- Philipp Moritz.
- Soroush Nasiriany.
- Ashwin Vangipuram.
- Jichan Chung.
- Druv Pai.
- Josh Sanz.

