CS 189/289A Introduction to Machine Learning

Fall 2021 Jennifer Listgarten and Jitendra Malik

Final

- Do not open the exam before you are instructed to do so.
- **Electronic devices should be turned off for the entire exam duration**, including cell phones, tablets, headphones, and laptops. Turn your cell phone off, or risk getting a zero on the exam.
- The exam is closed book, closed notes except your **two pages of cheat sheet**. You are allowed two double-sided handwritten 8.5x11 inch cheat sheets.
- You have 2 hours and 50 minutes (unless you are in the DSP program and have an allowance of 150% or 200% time).
- Please write your initials **and your student ID** at the top right of each page after this one. Finish this by the end of your 2 hours and 50 minutes.
- Mark your answers on the exam itself in the space provided. Only the answers written in the provided space will be graded. Do **not** attach any extra sheets.
- For the questions in Section 2, you should mark true or false for each statement. For each question, there may be more than one true option, but there is always at least one true option. Note that there is a penalty for marking the incorrect option, but there is no penalty for not marking either option.
- The last question (Question 8) is for CS289A students only. Students enrolled in CS189 will **not** receive any credit for answering this question.

First name	
Last name	
SID	
First, last name, and SID of student to your left	
First, last name, and SID of student to your right	

○ CS 189

○ CS 289A

1 Multiple Choice (Single Answer)

1. (1 point) Suppose $X_1, X_2, \ldots, X_n$ are drawn i.i.d. from the distribution $p_\theta(x) \propto \frac{1}{2}\exp(-|x-\theta|)$. What is the maximum likelihood estimate of θ?

   A. $\sum_{i=1}^{n} \frac{X_i}{n}$

   B. $\sum_{i=1}^{n} \frac{X_i}{n-1}$

   C. The median of $X_1, X_2, ..., X_n$.

   D. The mode of $X_1, X_2, ..., X_n$.

   Solution: C.

2. (1 point) How does masked self-attention work for transformer decoder models? I.e., ignoring any dropout, when computing self-attention weights for the token at time step t,

   A. All weights are set equal to zero independently at random with some probability.

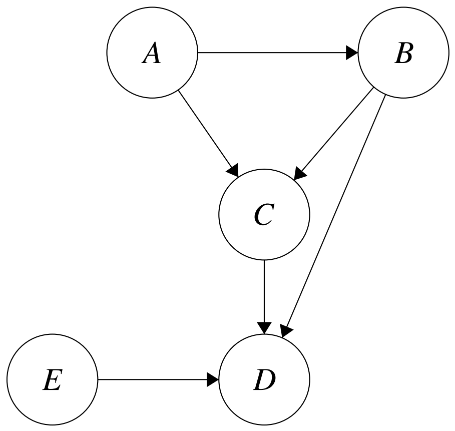
   B. Weights corresponding to time steps greater than t are set equal to zero.

   C. Weights corresponding to time steps less than t are set equal to zero.

   D. Every k-th weight is set to zero, where k is a hyperparameter.

   Solution: B.

3. (1 point) Consider the graphical model below. Which of the following is a joint factorization of the distribution of random variables.



   A. $p(A,B,C)\,p(BC,D)\,p(E)$

   B. $p(A)\,p(B \mid A,C)\,p(C \mid A)\,p(D \mid B,C,E)\,p(E)$

   C. $p(A)\,p(B \mid A)\,p(C \mid A,B)\,p(D \mid B,C,E)\,p(E)$

   D. $p(A \mid B,C)\,p(B \mid C,D)\,p(C \mid D)\,p(D)\,p(E \mid D)$

   Solution: C.

4. (1 point) Suppose that we have a transformer model, and we replace the multi-headed attention mechanism with a single-head attention mechanism (i.e. 1 head). What will be the effect on the model?

   A. The transformer will no longer be able to learn long-range dependencies.

   B. The transformer will have difficulty attending to multiple inputs.

   C. The transformer will not be able to determine the relative word positions in the sequential data.

   D. The transformer will perform computation slower because it has fewer attention heads.

   Solution: B.

5. (1 point) Let $\mathbf{x}$ be a random vector of flattened RGB images (e.g., $\mathbf{x} \in \mathbb{R}^{28\times28\times3}$). Suppose $p(\mathbf{x})$ is a Multivariate Gaussian distribution. If we apply n successive convolutions to the image (only convolutions and no activation functions, pooling or feed-forward layers), i.e. $\mathbf{z} = f(\mathbf{x})$, where $f(\cdot)$ is a function that applies n consecutive convolutions, will $p(\mathbf{z})$ be a Multivariate Gaussian distribution?

   A. Yes

   B. No

   Solution: A. The convolution is a linear operator and the compositions of multiple linear operators is still a linear operator. if $\mathbf{x}$ is a MVG, then $A\mathbf{x}$ is MVG for any linear operator A.

2 Multiple Choice (Multiple Answer)

Fill in either true or false for all statements: there may be more than one true option, but there is always at least one true option. Each question is worth two points, and each statement is worth 0.5 points. Marking the incorrect option will result in a penalty of -0.5 points, whereas not choosing an option does not have a penalty. Your score for each question will be ReLU(points).

1. (2 points) Which of the following statements are true about maximum a posteriori (MAP) estimation? Mark true or false for the statements below.

   ◯ T ◯ F : In MAP estimation, we have a probability distribution on the parameters of the model.

   ◯ T ◯ F : MLE is a special case of MAP where the prior on each parameter is a Bernoulli distribution.

   ◯ T ◯ F : The MAP estimate is equivalent to maximum likelihood estimate.

   ◯ T ◯ F : MAP provides a point estimate but does not provide uncertainty estimation.

   Solution: A and D are true.

   Why B is false: MLE is a special case of MAP where the prior for each parameter is a uniform distribution.

   Why C is false: According to Bayes' rule, $p(\theta \mid X) \propto p(X \mid \theta)p(\theta)$, so the prior also factors in to the MAP estimate.

2. (2 points) What would be an advantage of using lasso regression as compared to using ridge regression? Mark true or false for the statements below.

   ◯ T ◯ F : Lasso regression has a closed-form solution and therefore is less costly computationally.

   ◯ T ◯ F : Lasso regression is more effective at reducing overfitting than ridge regression.

   ◯ T ◯ F : Lasso regression can be used to select out important features.

   ◯ T ◯ F : Lasso regression avoids inverting singular matrices, whereas ridge regression doesn't.

   Solution: C is true. A, B, and D are false.

   Why A is false: Ridge regression has a closed-form solution

   Why B is false: One can turn up the regularization strength in both lasso regression and ridge regression.

   Why D is false: The matrix inversion in the closed form solution for ridge regression is on a non-singular matrix.

3. (2 points) Which of the following statements is true about the bias-variance decomposition? Mark true or false for the statements below.

◯ T ◯ F : Compared to ordinary least squares (OLS), ridge regression generally has higher bias and lower variance.

◯ T ◯ F : Compared to one single decision tree, random forests with decision trees of the same depth generally have higher variance.

◯ T ◯ F : For a soft-margin SVM, decreasing the regularization strength generally leads to lower bias.

◯ T ◯ F : Applying an RBF kernel in kernelized SVM can lead to reductions in model bias on some data sets as compared to a linear kernel.

Solution: A, C, and D are true. B is false.

4. (2 points) River has some high-dimensional data from a genetic sequencing experiment, but he's not sure whether to use PCA or tSNE for dimensionality reduction. Help him by selecting all the true statements here. Mark true or false for the statements below.

◯ T ◯ F : If River wants deterministic dimensionality reduction results, tSNE is a better choice.

◯ T ◯ F : If River is confident that the underlying structure of his data is non-linear, tSNE is probably a better choice.

◯ T ◯ F : If River wants to ensure that local structures are preserved (neighboring points in the original data sets are still close to each other in the new representation), tSNE is a better choice.

◯ T ◯ F : If River wants to use the reduced features in a linear regression model, tSNE is probably a better choice.

Solution: B and C are true. A and D are false.

5. (2 points) Which of the following finds the first principal component, $\mathbf{w}$, of a $n \times d$ data matrix, $X = \begin{bmatrix} X_1 & \dots & X_n \end{bmatrix}^\top$?

◯ T ◯ F : $\max_{\mathbf{w}} \sum_{i=1}^{n} \left\| X_i - \frac{X_i^\top \mathbf{w}}{\|\mathbf{w}\|} \mathbf{w} \right\|^2$

◯ T ◯ F : $\min_{\mathbf{w}} \sum_{i=1}^{n} \left\| X_i - \frac{X_i^\top \mathbf{w}}{\|\mathbf{w}\|} \mathbf{w} \right\|^2$

◯ T ◯ F : $\max_{\substack{\mathbf{w} \\ s.t.\ \|\mathbf{w}\|=1}} \mathbf{w}^\top X^\top X \mathbf{w}$

◯ T ◯ F : $\min_{\substack{\mathbf{w} \\ s.t.\ \|\mathbf{w}\|=1}} \mathbf{w}^\top X^\top X \mathbf{w}$

Solution: B and C are true, sort of. B actually has a typo and should say $\min_{\mathbf{w}} \sum_{i=1}^{n} \left\| X_i - \frac{X_i^\top \mathbf{w}}{\|\mathbf{w}\|^2} \mathbf{w} \right\|^2$. We thus accepted either true or false for B.

6. (2 points) Which of the following statements is true regarding regularized linear regression? Mark true or false for the statements below.

◯ T ◯ F : In the Bayesian MAP interpretation, Ridge regression can be interpreted as linear regression for which the coefficients have a Laplace prior.

◯ T ◯ F : In the Bayesian MAP interpretation, Ridge regression can be interpreted as linear regression for which the coefficients have a Gaussian prior.

◯ T ◯ F : In the Bayesian MAP interpretation, Lasso regression can be interpreted as linear regression for which the coefficients have a Laplace prior.

◯ T ◯ F : In Ridge regression, the regularization parameter, λ, can be fit with MLE on the training set.

Solution: B and C are true.

Why A is false: In the Bayesian MAP interpretation, Ridge regression can be interpreted as linear regression for which the coefficients have a Gaussian prior, not a Laplace Prior.

Why D is false: The regularization parameter should be chosen using cross-validation and should not be fit with MLE on the training set.

7. (2 points) Which of the following statements are true regarding do-interventions? Mark true or false for the statements below.

◯ T ◯ F : X and Y are confounded if $p(Y = y \mid \text{do}(X = x)) = p(Y = y \mid X = x)$

◯ T ◯ F : X and Y are confounded if $p(Y = y \mid \text{do}(X = x)) \neq p(Y = y \mid X = x)$

◯ T ◯ F : $p(Y = y \mid \text{do}(X = x))$ can never be estimated from observational data if there is a confounding variable, Z.

◯ T ◯ F : The adjustment formula provides a way to compute $p(Y = y \mid \text{do}(X = x))$ using standard conditional and marginal probability distributions.

Solution: B and D are true.

Why A is false: Because B is true.

Why C is false: The adjustment formula provides a way to estimate $p(Y = y \mid \text{do}(X = x))$ from observational data in the presence of confounders.

8. (2 points) Which of the following statements are true regarding random forests? Mark true or false for the statements below.

◯ T ◯ F : Random forests fit decision trees to bootstrapped samples of the data.

◯ T ◯ F : Random forests are trained with Stochastic Gradient Descent (SGD).

◯ T ◯ F : Random forests are an example of ensemble methods.

◯ T ◯ F : Random forests can only be used for regression and not classification.

Solution: A and C are true.

Why B is false: Random forests are not trained by SGD.

Why D is false: Random forests can be used for regression or classification.

9. (2 points) Recall that the objective function for Soft-margin SVMs is:

$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{s.t.} \quad y_i\left(\mathbf{w}^\top \mathbf{x}_i - b\right) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

Which of the following statements are true regarding SVMs? Mark true or false for the statements below.

◯ T ◯ F : The objective function above can be rewritten in terms of a squared error loss and a ridge penalty.

◯ T ◯ F : Hard-margin SVM is a special case of Soft-margin SVM that sets C to 0.

◯ T ◯ F : SVM and Neural networks are usually trained using the same optimization technique.

◯ T ◯ F : The corresponding kernelized variant can be derived by analyzing the dual optimization problem.

Solution: D is true.

Why A is false: the objective function above can be rewritten in terms of a hinge loss and a ridge penalty, not a squared error loss.

Why B is false: Hard-margin SVM is a special case of Soft-margin SVM that sets C to infinity.

Why C is false: SVMs are trained using quadratic programming solvers. Neural networks are trained using stochastic gradient descent.

10. (2 points) Which of the following statements are true regarding Multiway Classification (i.e. classification with more than two classes to choose from) **with neural networks**? Mark true or false for the statements below.

◯ T ◯ F : Multiway classification often uses the softmax activation function for computing the predicted probability for each class.

◯ T ◯ F : In multiway classification, a class C_i is predicted only if the predicted probability for the class is greater than 50%.

◯ T ◯ F : The cross entropy loss function derived for multiway classification can be interpreted through the lens of maximum likelihood.

◯ T ◯ F : Multiway classification is handled by performing a series of binary classifications.

Solution: A and C are true.

Why B is false: the predicted class is the one that has the highest predicted probability regardless of whether or not this is greater than 50%

Why D is false: for neural networks, a softmax activation function is used to perform multiway classification. There is no need to perform a series of binary classifications.

11. (2 points) Suppose we have a $2N$-dimensional vector $\mathbf{x}$ with elements, x_i, $i \in \{1, 2, \ldots, 2N\}$. All of the odd indices are *i.i.d* gaussian with mean 0 and variance σ^2. The even indices are defined as: $x_j = x_{j-1}^2$ for even j. Let Σ be the covariance matrix of $\mathbf{x}$. Mark true or false for the statements below.

Hint: Recall that the the third moment of an MVG is 0 (i.e. $\mathbb{E}[x^3] = 0$ for any mean 0 Gaussian random variable, x)

○ T ○ F : $\Sigma_{i,i+1}$ is non-zero.

○ T ○ F : Σ is a diagonal matrix.

○ T ○ F : Σ is PSD.

○ T ○ F : $\mathbf{x}$ is a multivariate gaussian random vector.

Solution: B and C are true.

Why A is false: $\Sigma_{i,i+1} = \mathbb{E}[x \cdot x^2] - \mathbb{E}[x]\mathbb{E}[x^2]$ where $x \sim N(0, \sigma^2)$. $\mathbb{E}[x] = 0 \implies \Sigma_{i,i+1} = \mathbb{E}[x^3]$. Due the symmetry of the Gaussian distribution, all the odd moments are zero, and thus, $\Sigma_{i,i+1} = 0$.

Why D is false: $\mathbf{x}$ is MVG implies that x_j are marginally Gaussian for all j. However, x_j for even j are not Gaussian and thus, $\mathbf{x}$ is not MVG.

3 Stochastic gradient descent

1. (7 points) In this problem, we will walk through a simple example for stochastic gradient descent. Consider the quadratic loss function

$$L(w, \{y_i\}_{i=1}^n) = \frac{1}{n}\sum_{i=1}^{n} L(w, y_i) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(w - y_i)^2 \tag{1}$$

where $y_1, \cdots, y_n$ are given scalars and $w \in \mathbb{R}$ is a single parameter that we wish to estimate.

In order to estimate the optimal w^* that minimizes our loss function we will use stochastic gradient descent (SGD).

Typically, when using SGD, we randomly sample a data point y_i from the set of all data points $\{y_i\}_{i=1}^n$ and compute the gradient update step on the sampled y_i. Rather than sampling with replacement, we consider a variant of SGD where we sample without replacement. Specifically, we will shuffle our dataset and then run SGD where we use y_1 as the sample for the time step $t = 1$, y_2 as the sample for the time step $t = 2$ and so on. For this question you may assume we have a already shuffled dataset $y_1, \cdots, y_n$.

(a) (2 points) What is the stochastic gradient update step for w at time t under the optimization objective to minimize $L(w)$? Assume the step size is η. Write down an expression for $w^{(t+1)}$ using η, $w^{(t)}$ and y_t. Remember that stochastic gradient descent only considers one data point in each update step.

Solution: $\nabla L(w) = w^{(t)} - y_t$ for the update step t, and therefore

$$w^{(t+1)} = w^{(t)} - \eta(w^{(t)} - y_t)$$

(b) (3 points) Now suppose we use the dynamic step size $\eta = \frac{1}{t}$ that changes according to t. Write the expression for $w^{(t)}$ after t steps of SGD. Assume we start with $w^{(1)} = 0$. Your expression for $w^{(t)}$ should only include $y_1, \cdots, y_t$ and t. You may use scratch paper if needed, but ensure that your final answer is in the space provided below on this page.

Solution:

$$w^{(2)} = w^{(1)} - w^{(1)} + y_1 = y_1$$

$$w^{(3)} = w^{(2)} - \frac{1}{2}(w^{(2)} - y_2) = \frac{1}{2}(y_1 + y_2)$$

By induction,

$$w^{(t+1)} = w^{(t)} - \frac{1}{t}(w^{(t)} - y_t) = \frac{1}{t}(y_1 + \cdots + y_t).$$

(c) (2 points) Instead of SGD, we can also determine a closed form solution to our optimization problem

$$w^* = \arg\min_w L(w)$$

What is this closed form solution? You are not required to show the derivation, though you may choose to in order to receive partial credit if your final answer is incorrect.

Is the closed form solution the same as the result of running SGD for n steps (one epoch on the dataset)? If not, how does it differ?

Solution:

$$\frac{d}{dw}L(w)\ |_{w=w^*} = \frac{1}{n}\sum_{i=1}^{n}(w^* - y_i) = 0 \implies w^* = \frac{1}{n}\sum_{i=1}^{n} y_i$$

They are the same.

4 Baking bread on rainy days

1. (10 points) Connor bakes bread every day and experiments with different variables x in bread baking such as the amount of water, the time for rising the dough, and the oven temperature. After baking each bread, Connor measures the quality of the bread and summarizes the bread quality as a variable y. Assuming a linear relationship between the bread quality y and the bread baking variables x, Connor now wants to build a linear regression model for the bread quality.

   In addition to the controlled variables $x \in \mathbb{R}^d$, the bread quality is also influenced by an unobserved random variable $\epsilon \in \mathbb{R}$ that adds "noise" to the linear relationship. For each of the n observations $\{x_i, y_i\}_{i=1}^n$,

$$y_i = x_i^T\beta + \epsilon_i, \quad i = 1, \cdots, n. \tag{2}$$

   Note that all observations are independent from one another.

   Typically, we assume that each error variable ϵ_i is drawn from a Gaussian distribution

$$\epsilon_i \sim N(0, \sigma_i^2). \tag{3}$$

   However, Connor notices that the bread quality is less consistent on rainy days than on sunny days. In other words, the variance σ_i^2 of the error variable ϵ_i is twice as large on rainy days than on sunny days.

   Connor has collected bread data on $2m$ days, where the first m examples $\{x_i, y_i\}_{i=1}^m$ were collected on sunny days and have noise variance σ^2, and the second m examples $\{x_i, y_i\}_{i=m+1}^{2m}$ were collected on rainy days have noise variance $2\sigma^2$. He does not know the noise variance σ^2.

   (a) (2 points) What is the log-likelihood function for the regression coefficient β and the sunny day noise variance σ^2? That is, derive an expression for

$$\ell(\beta, \sigma^2) = \log p(y_1, y_2, \cdots, y_{2m} \mid \beta, \sigma^2)$$

   in terms of β, σ^2, m and the observations $\{x_i, y_i\}_{i=1}^{2m}$.
   Your may use C in your answer to represent a constant that does not depend on β, σ^2 or $\{x_i, y_i\}_{i=1}^{2m}$.

   Solution: For each y_i, the likelihood is

$$p(y_i \mid \beta, \sigma^2) = \frac{1}{\sigma_i\sqrt{2\pi}} \exp\{-\frac{(y_i - x_i^T\beta)^2}{2\sigma_i^2}\},$$

   and the log-likelihood is

$$\log p(y_i \mid \beta, \sigma^2) = -\frac{1}{2}\log\sigma_i^2 - \frac{(y_i - x_i^T\beta)^2}{2\sigma_i^2} - \frac{1}{2}\log(2\pi).$$

   The overall log-likelihood is

$$\ell(\beta, \sigma^2) = \log p_\beta(y_1, y_2, \cdots, y_n) = -\frac{1}{2}\sum_{i=1}^n \log\sigma_i^2 - \sum_{i=1}^n \frac{(y_i - x_i^T\beta)^2}{2\sigma_i^2} - \frac{n}{2}\log(2\pi).$$

When we substitute in the given noise variance values,

$$\ell(\beta, \sigma^2) = -m \log \sigma^2 - \frac{m}{2} \log 2 - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y_i - x_i^T \beta)^2 - \frac{1}{4\sigma^2} \sum_{i=m+1}^{2m} (y_i - x_i^T \beta)^2 - \frac{n}{2} \log(2\pi).$$

(b) (2 points) Next we will work towards finding the maximum likelihood estimate (MLE) of the linear regression model parameters. To simplify the calculations, we want to use the vector and matrix notation instead of the summation notation. Let $X \in \mathbb{R}^{2m \times d}$ be the feature matrix and $y \in \mathbb{R}^{2m}$ be the bread quality vector. For example, the loss for ordinary least squares regression (not applied here) could be written as $\left\|y - X\beta\right\|^2$.

Let Λ denote the $2m$ by $2m$ diagonal matrix

$$\Lambda = \begin{bmatrix} I_m & 0 \\ 0 & 2I_m \end{bmatrix}.$$

Derive an expression $\ell(\beta, \sigma^2)$ in the vector and matrix notation in terms of m, β, σ^2, X, y, and Λ.

Hint: To sanity check your solution, noisier examples on the rainy days ought to be weighed less in the regression.

Solution:

The above log-likelihood $\ell(\beta, \sigma^2)$ can then be reformatted as

$$\ell(\beta, \sigma^2) = -m \log \sigma^2 - \frac{1}{2\sigma^2} \left\| \sqrt{\Lambda^{-1}} (y - X\beta) \right\|^2 - m \log(2\pi)$$

$$\ell(\beta, \sigma^2) = -m \log \sigma^2 - \frac{1}{2\sigma^2} (y - X\beta)^T \Lambda^{-1} (y - X\beta) - m \log(2\pi).$$

Note that there is an inverse to the Λ matrix. You should not be double penalized for missing the inverse if it propagates from part (b) to parts (c) and (d).

(c) (3 points) What is the maximum likelihood estimate (MLE) of the linear regression model parameters β?

Please use the vector and matrix notation in writing your answer. Derive an expression of $\hat{\beta}_{MLE}$ in terms of m, X, y, and Λ.

Solution: Using the log-likelihood above in the vector and matrix notation, the gradient with respect to β is

$$\nabla_\beta \ell(\beta, \sigma^2) = -\frac{1}{2\sigma^2} \nabla_\beta \left(\Lambda^{-1} (y - X\beta)^T (y - X\beta) \right) = -\frac{1}{2\sigma^2} \left(2X^T \Lambda^{-1} X\beta - 2X^T \Lambda^{-1} y \right)$$

Solving for $\nabla_\beta \ell(\beta, \sigma^2) = 0$ at $\beta = \hat{\beta}$ yields

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

(d) (3 points) What is the maximum likelihood estimate (MLE) of the sunny day noise variance σ^2? Derive an expression of $\hat{\sigma}^2_{MLE}$ in terms of $\hat{\beta}_{MLE}, m, X, y$, and Λ.

Solution: The gradient with respect to σ^2 is

$$\nabla^2_{\sigma}\ell(\beta, \sigma^2) = -\frac{m}{\sigma^2} + \frac{1}{2\sigma^4}(y - X\hat{\beta})^T\Lambda^{-1}(y - X\hat{\beta}).$$

Solving for $\nabla^2_{\sigma}\ell(\beta, \sigma^2) = 0$ at $\beta = \hat{\beta}$ and $\sigma^2 = \hat{\sigma}^2$ yields

$$\hat{\sigma}^2 = \frac{1}{2m}(y - X\hat{\beta})^T\Lambda^{-1}(y - X\hat{\beta})$$

5 Multi-query attention

1. (8 points) Multi-head attention layers, as used in the Transformer neural sequence model, are powerful for moving information across and between sequences. While training these layers is generally fast and simple, due to parallelizability across the length of the sequence, incremental inference at test time for Transformer decoders (where such parallelization is impossible) is often slow, due to the memory-bandwidth cost of repeatedly loading the large "keys" and "values" tensors. In this problem, we will look into the memory usage of multi-head attention.

   (a) (1 point) In the multi-head scaled dot-product attention, recall that there are multiple attention "heads". Each head takes as input a query matrix $Q \in \mathbb{R}^{T \times d_k}$, a key matrix $K \in \mathbb{R}^{S \times d_k}$ and a value matrix $V \in \mathbb{R}^{S \times d_v}$. S and T may differ in general, for example in cross-attention where T is the target sequence length and S is the source sequence length. d_k and d_v can also differ in general. However, in this problem, for simplicity, assume $S = T = L$ (sequence length) and $d_k = d_v = d_h$ (head dimension). The attention weights in each head are:

   $$\text{Attention weights}(Q, K) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right),$$

   and the attention output from each head is:

   $$\text{Attention}(Q, K, V) = \text{Attention weights}(Q, K)\ V = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V.$$

   What is the dimension of the attention weight matrix Attention weights(Q, K) and the attention output matrix Attention(Q, K, V) for each head?

   Solution: The attention weight matrix Attention weights(Q, K) has dimensions $T \times S = L \times L$. The attention output matrix Attention(Q, K, V) has dimensions $T \times d_v = L \times d_h$.

   (b) (2 points) What is the total size of memory needed when computing Attention(Q, K, V) for each head? Write your answer using big-O notation $O(\cdot)$ in terms of L and d_h.

   Solution: For each head,

   - Computing the matrix product QK^T requires $O(T \times S)$ additional memory.
   - Computing the softmax requires $O(T \times S)$ additional memory.
   - Computing the attention output requires $O(T \times d_h)$ additional memory.

   Adding these parts together, the overall attention operation for each head requires $O(L^2 + d_h L)$ memory.

   (c) (1 point) In the multi-head self-attention layers in Transformer, the query $Q \in \mathbb{R}^{L \times d_h}$, key $K \in \mathbb{R}^{L \times d_h}$, and value $V \in \mathbb{R}^{L \times d_h}$ matrices **for each attention head** are linearly projected from the same embedding matrix X of dimension $L \times e$, where e is the embedding dimension. Note that each attention head is associated with three projection matrices: one projection matrix from the embeddings to the attention query, one to the attention key, and one to the attention value.

   What is the dimension of each of these three linear projection matrices (for one head)? Write your answer in terms of L, d_h, and e.

   Solution: Each projection matrix is of shape $d_h \times e$.

(d) (2 points) What is the total size of memory needed when computing the linear projections and the attention operations **for H heads**? **Include** the memory required for storing the projection weights and the intermediate query, key, and value matrices.

Typically in Transformer models the embedding dimension $e = \sum_{j=1}^{H} d_h = Hd_h$ is equal to the sum of head dimensions across all attention heads. You may use this assumption to simplify your answer. You are not required to write down the generic formula without the simplifying assumption, though you may choose to in order to receive partial credit if your final answer is incorrect.

Use the big-O notation $O(\cdot)$ in terms of the embedding dimension e, the sequence length L, and the number of heads H.

Solution: For each head the total memory is $O(ed_h + L^2 + d_h L)$.

For H heads it's $O(Hed_h + HL^2 + Hd_h L) = O(e^2 + HL^2 + eL)$.

(e) (1 point) For long sequence length ($L >> e$), what is the majority of the memory consumption used for?

Solution: When $L >> e$, the $O(HL^2)$ part of the memory consumption dominates, going towards the attention weight matrices.

(f) (1 point) For models with "wide" embedding dimension on short sequence length ($e >> L$), what is the majority of the memory consumption used for?

Solution: When $e >> L$, the $O(e^2)$ part of the memory consumption dominates, going towards the linear projection matrices that project the embedding to Q, K, V.

6 Ocean wave patterns

1. (10 points) Stella watches ocean waves every day and notices a daily periodic pattern. She is curious to study the sea surface elevation as a function of the time of the day. Stella uniformly sampled n observation times $x_1, \cdots, x_n$ during the day. For simplicity, we will denote the time of the day to be a value in the range of $[-\pi, \pi]$, with $-\pi$ representing midnight and π representing the midnight of the next day. In other words, the x values of the data points are randomly sampled from the uniform distribution on the domain $[-\pi, \pi]$.

   After collecting the sea surface elevation data at these randomly sampled time points, Stella wishes to estimate the sea surface elevation at noon (corresponding to $x = 0$), and decides to apply k-nearest neighbors (kNN) regression. Stella's version of kNN regression takes the **unweighted average** of the k nearest neighbors.

   Even though Stella doesn't know this, the true sea surface elevation level at time x is $cos(x)$. Stella is able to measure the sea surface level perfectly, so there is no noise in her measurements.

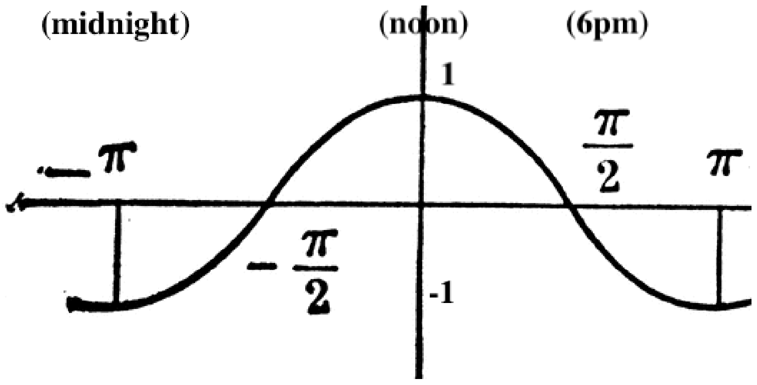


Figure 1: The true sea surfacel elevation level function $f(x) = cos(x)$ during the interval $[-\pi, \pi]$.

   (a) (2 points) Among the n observation times $x_1, \cdots, x_n \in [-\pi, \pi]$ that Stella uniformly randomly sampled, what is the probability that at least one point will be within the interval $[-\frac{\pi}{3}, \frac{\pi}{3}]$?

   Solution: For each random sample x_i, the probability that x_i is in the interval $[-\frac{\pi}{3}, \frac{\pi}{3}]$ is

$$P(x_i \in [-\frac{\pi}{3}, \frac{\pi}{3}]) = \frac{1}{3}.$$

   The probability that none of the points fall in this interval is

$$P(x_1, \cdots, x_n \notin [-\frac{\pi}{3}, \frac{\pi}{3}]) = \prod_{i=1}^{n} P(x_i \notin [-\frac{\pi}{3}, \frac{\pi}{3}]) = (1 - P(x_i \in [-\frac{\pi}{3}, \frac{\pi}{3}]))^n = (\frac{2}{3})^n.$$

   Therefore,

$$P(\text{at least one } x_i \in [-\frac{\pi}{3}, \frac{\pi}{3}]) = 1 - P(x_1, \cdots, x_n \notin [-\frac{\pi}{3}, \frac{\pi}{3}]) = 1 - (\frac{2}{3})^n$$

(b) (2 points) With $k = 1$ (1NN regression), let $\hat{f}(x)$ be the 1NN estimate of the sea level at time $x \in [-\pi, \pi]$. Note that $\hat{f}(x)$ depends on the observations that Stella sampled.

What is the probability that Stella's estimate $\hat{f}(x = 0)$ for the noon sea level will be greater than or equal to $\frac{1}{2}$? Your answer should depend on n.

Hint: $\cos(\pi/3) = \frac{1}{2}$.

Solution: For $k = 1$, the estimate $\hat{f}(x = 0)$ will be determined by the closest observed point to $x = 0$. This estimate will be greater than or equal to $\frac{1}{2}$ if and only if there is at least one observed x_i that falls in $x_i \in [-\frac{\pi}{3}, \frac{\pi}{3}]$. Therefore, based on part (a),

$$P(\hat{f}(x = 0) \geq \frac{1}{2}) = 1 - (\frac{2}{3})^n$$

(c) (2 points) Still with $k = 1$ (1NN regression), what is the probability that Stella's estimate $\hat{f}(x = 0)$ for the noon sea level will be within δ to the true sea level at noon? Recall that the true sea level is 1 at noon according to the function $f = \cos(x)$. Here, $\delta \in (0, 1)$ is a given constant.

Your answer should depend on n and δ, and you may use the arccos function. Recall that the arccos function is the inverse of the cosine function. For example, $\arccos(1) = 0$, $\arccos(0) = \pi/2$, and $\arccos(-1) = \pi$. Recall that the arccos function has $[-1, 1]$ as its domain and $[0, \pi]$ as its output range.

Solution: Similarly, Stella's estimate $\hat{f}(x = 0)$ will be greater than or equal to $1 - \delta$ if and only if there is at least one observed x_i that falls in $x_i \in [-\arccos(1 - \delta), \arccos(1 - \delta)]$. Stella's 1NN estimate would never to greater than 1 since 1 is the maximum true sea level that she can observe. Therefore,

$$P(|\hat{f}(x = 0) - 1| \leq \delta) = P(\hat{f}(x = 0) \geq 1 - \delta) = 1 - (1 - \frac{\arccos(1 - \delta)}{\pi})^n$$

Note the answer could also be written as $1 - (\frac{\arccos(\delta - 1)}{\pi})^n$ due to the symmetry in arccos.

(d) (1 point) What happens to the above probability when $n \to \infty$? (The above probability refers to the probability that Stella's estimate $\hat{f}(x = 0)$ for the noon sea level will be within δ to the true sea level at noon.)

A. The probability goes to $1 - \delta$ as $n \to \infty$.

B. The probability goes to $1/2$ as $n \to \infty$.

C. The probability goes to 1 as $n \to \infty$.

D. The probability goes to δ as $n \to \infty$.

Solution: C.

As $n \to \infty$, $P(|\hat{f}(x = 0) - 1| \leq \delta) \to 1$. In other words, as n becomes large, Stella almost always has her estimate within error δ for any fixed $\delta \in (0, 1)$.

(e) (2 points) Now consider kNN regression with $k > 1$. What is the probability that **at least** k of the n observation points will be within the interval $[-\frac{\pi}{3}, \frac{\pi}{3}]$?

Write your answer as a function of n and k, and you may use binomial terms in the form of $\binom{a}{b}$ in your answers and summation over such binomial terms.

Solution: The number of points $N_{[-\frac{\pi}{3},\frac{\pi}{3}]}$ within the interval $[-\frac{\pi}{3},\frac{\pi}{3}]$ follows a binomial distribution $N_{[-\frac{\pi}{3},\frac{\pi}{3}]} \sim Binom(n, p = \frac{1}{3})$.

Therefore,

$$P(N_{[-\frac{\pi}{3},\frac{\pi}{3}]} \geq k) = 1 - \sum_{i=0}^{k-1} P(N_{[-\frac{\pi}{3},\frac{\pi}{3}]} = i) = 1 - \sum_{i=0}^{k-1} \binom{n}{i} (\frac{1}{3})^i (\frac{2}{3})^{n-i} = 1 - \sum_{i=0}^{k-1} \binom{n}{i} \frac{2^{n-i}}{3^n}$$

Note that the answer is also equivalent to

$$\sum_{i=k}^{n} \binom{n}{i} \frac{2^{n-i}}{3^n}$$

(f) (1 point) Which of the following statements is true about estimating the noon time sea level ($x = 0$) and estimating the 6pm sea level ($x = \frac{\pi}{2}$))? There is only one correct choice.

A. Stella's estimate for the noon time sea level with $k = 3$ is always at least as accurate as her estimate with $k = 1$.

B. Stella's estimate for the noon time sea level with $k = 1$ is always at least as accurate as her estimate with $k = 3$.

C. Stella's estimate for the 6pm sea level with $k = 3$ is always at least as accurate as her estimate with $k = 1$.

D. Stella's estimate for the 6pm sea level with $k = 1$ is always at least as accurate as her estimate with $k = 3$.

Solution: B.

Stella is always better off with $k = 1$ for estimating the noon time sea level, since the noon time sea level is at the maximum and using $k > 1$ will lower the estimated value.

However, for estimating the 6pm sea level (at $x = \pi/2$), using $k > 1$ could be beneficial in some cases since more samples could average out the overestimate and the underestimate.

7 Residual connections

1. (11 points) Residual connections are a very important innovation in the practice of deep learning because they allow more effective training of very deep neural networks. In the previous homework assignments, we have seen that residual connections are useful in Transformer models. In this problem, we will examine why residual connections help with training deep networks. Throughout this problem, we will be working with **fully-connected neural networks**.

   Recall that in a fully-connected neural network of depth k, we have k fully-connected hidden layers. In the t-th layer ($t = 1, \cdots, k$), the input $\mathbf{s}_t \in \mathbb{R}^d$ to the t-th layer is the output of the previous layer, and the output $\mathbf{s}_{t+1} \in \mathbb{R}^d$ from the t-th layer will be the input to the next layer. Assume for simplicity that all hidden layers have the same output dimensionality d, and this is equal to the input dimensionality. First, a linear transformation W_t is applied to the activation $\mathbf{s}_t$, and then an activation function A is applied on the input $\mathbf{s}_t$. Mathematically,

$$\mathbf{s}_{t+1} = \mathrm{A}(W_t\mathbf{s}_t) \tag{4}$$

   where $W_t \in \mathbb{R}^{d\times d}$ is the weight matrix for the t-th layer. Assume all the hidden dimensions are the same d. The initial input to the first layer s^1 is the input x to the neural network. At the end of the k hidden layers, there is a linear output projection such that the output of the neural network is $y = W_{k+1}\mathbf{s}_{k+1}$. Overall, the output y is a function of x and all the weight matrices W_t for $t = 1, \cdots, k+1$.

   For simplicity throughout this question, we will assume that **the activation function is the identity** ($A(s) := s$), but the intuition from this exercise also applies more generally.

   (a) (1 point) What is the gradient of $\mathbf{s}_{t+1}$ with respect to $\mathbf{s}_t$? In other words, what is $\nabla_{\mathbf{s}_t}\mathbf{s}_{t+1}$? Note that this gradient will be a matrix of dimension $d \times d$, since both $\mathbf{s}_t$ and $\mathbf{s}_{t+1}$ are d-dimensional vectors.

   Solution: We seek the Jacobian of the function $W_t\mathbf{s}_t$. One way to do this is component-wise. $(\nabla_{\mathbf{s}_t} W_t\mathbf{s}_t)_{ij} = \frac{\partial \mathbf{s}_{t+1i}}{\partial \mathbf{s}_{tj}}$. Note that the $\mathbf{s}_i^{t+1} = \sum \mathbf{s}_{k=1}^{n} W_{tik}\mathbf{s}_{tk}$. Thus $(\nabla_{\mathbf{s}_{t+1}} W_t\mathbf{s}_t)_{ij} = W_{tij}$ which implies that the $\nabla_{\mathbf{s}_t}\mathbf{s}_{t+1} = \nabla_{\mathbf{s}_t} W_t\mathbf{s}_t = W_t$.

   (b) (1 point) Now let's consider a loss function $L(x, y)$. During backpropagation, if we already calculated the gradient $\nabla_{\mathbf{s}_{t+1}} L$ (this is a d-dimensional vector), what is the gradient $\nabla_{\mathbf{s}_t} L$? Write down an expression for $\nabla_{\mathbf{s}_t} L$ using W_t and $\nabla_{\mathbf{s}_{t+1}} L$.

   Solution: This can also be done component-wise. Note that $\nabla_{\mathbf{s}_t} L = \left[\frac{\partial L}{\partial \mathbf{s}_{ti}}\right]$. Concentrating on the partial $\frac{\partial L}{\partial \mathbf{s}_{ti}} = \sum \mathbf{s}_{k=1}^{d} \frac{\partial L}{\partial \mathbf{s}_k^{t+1}} \frac{\partial \mathbf{s}_k^{t+1}}{\partial \mathbf{s}_i^{t}}$, we see that the overall gradient vector can be written as the matrix-vector product $\nabla_{\mathbf{s}_t}\mathbf{s}_{t+1}\ \nabla_{\mathbf{s}_{t+1}} L = W_t\ \nabla_{\mathbf{s}_{t+1}} L$.

   (c) (2 points) As described above, let $k+1$ be the index of the final output layer in the neural network after k hidden layers, and suppose we have the vector $\nabla_{\mathbf{s}_{k+1}} L$ (the gradient of the loss with respect to the final $(k+1)$-th layer input). For any $t = 1, \ldots, k$, write the gradient $\nabla_{\mathbf{s}_t} L$. Your answer may only contain the derivative term $\nabla_{\mathbf{s}_{k+1}} L$, all other derivatives need to be explicitly written. *Hint: You can use the product notation* $\prod$ *for matrix product to help you simplify your answer.*

Solution:

$$\nabla_{\mathbf{s}_t} L = W_t \, \nabla_{\mathbf{s}_{t+1}} L = W_t \, W_{k+1} \, \nabla_{\mathbf{s}_{k+2}} L = \cdots = (\prod_{j=t}^{k} W^j) \nabla_{\mathbf{s}_{k+1}} L$$

(d) (3 points) Suppose we have a symmetric matrix $W \in \mathbb{R}^{d \times d}$ whose largest eigenvalue $\lambda_1(W)$ satisfies $|\lambda_1(W)| < 1$, and let $\mathbf{x} \in \mathbb{R}^d$ be a d-dimensional vector.

Now consider taking the m-th power of the matrix for some integer $m \geq 1$. Derive an upper bound for the norm $||W^m \mathbf{x}||_2$ in terms of m, $||\mathbf{x}||_2$, and the largest eigenvalue $\lambda_1(W)$. As m becomes large, what happens to the norm of the m-th power of the matrix multiplied by the vector, $||W^m \mathbf{x}||_2$?

Solution: Since W is a real symmetric matrix, we can use the spectral decomposition $W = UDU^T$ where D is diagonal and U is orthogonal. The m-th power of W is then

$$W^m = (UDU^T)^m = UD^m U^T,$$

and therefore $W^k \mathbf{x} = UD^m U^T \mathbf{x}$. Since U is an orthogonal matrix, $\left\|U^T \mathbf{x}\right\|_2 = ||\mathbf{x}||_2$. Since the largest entry of D is $\lambda_1(W)$, the largest entry of D^m has magnitude $|\lambda_1(W)|^m$. Therefore, $\left\|DU^T \mathbf{x}\right\|_2 \leq |\lambda_1(W)|^m \left\|U^T \mathbf{x}\right\|_2 = |\lambda_1(W)|^m ||\mathbf{x}||_2$. Finally, again since U is orthogonal,

$$\left\|W^k \mathbf{x}\right\|_2 = \left\|UD^m U^T \mathbf{x}\right\|_2 = \left\|D^m U^T \mathbf{x}\right\|_2 \leq |\lambda_1(W)|^m \left\|U^T \mathbf{x}\right\|_2 = |\lambda_1(W)|^m ||\mathbf{x}||_2.$$

(e) (2 points) Suppose that for all t, W_t is symmetric and such that the largest eigenvalue $|\lambda_1(W_t)| < 1$. For any $t = 1, \cdots, k$, what is an upper bound for $\nabla_{\mathbf{s}_t} L$ in terms of the norm of the gradient at the last layer $\left\|\nabla_{\mathbf{s}_{k+1}} L\right\|_2$ and the largest eigenvalues of each weight matrix $\lambda_1(W_i)$?

Solution:

$$\left\|\nabla_{\mathbf{s}_t} L\right\|_2 = \left\|(\prod_{j=t}^{k} W^j) \nabla_{\mathbf{s}_{k+1}} L\right\|_2 \leq \left(\prod_{j=t}^{k} \left|\lambda_1(W^j)\right|\right) \left\|\nabla_{\mathbf{s}_{k+1}} L\right\|_2$$

(f) (1 point) From the previous parts of this problem, you should have observed a potentially problematic situation that can arise when training deep neural networks. What is the problem?

Solution: As the network becomes deeper, the gradient $\nabla_{\mathbf{s}_t} L$ at layer t becomes exponentially smaller according to $k - t$ where k is the network depth. This makes gradient descent on the network parameters inefficient.

(g) (1 point) Residual connections seeks to address this problem. Residual connections add the untransformed input into the output of the layer. Explicitly:

$$\mathbf{s}_{t+1} = \mathrm{A}(W_t \mathbf{s}_t) + \mathbf{s}_t. \tag{5}$$

Similar to part (a), compute $\nabla_{\mathbf{s}_t}\mathbf{s}_{t+1}$ for a residual network. What are the implications for the previously problematic issue with training very deep neural networks?

Solution:

$$\nabla_{\mathbf{s}_t}\mathbf{s}_{t+1} = W_t + I,$$

where I is the identity matrix.

Now even when the largest eigenvalue $\lambda_1(W_t)$ has very small magnitude $|\lambda_1(W_t)|$, the gradient $\nabla_{\mathbf{s}_t}\mathbf{s}_{t+1}$ has eigenvalue $1 + \lambda_1(W_t)$. Overall, the backpropagated gradient $\nabla_{\mathbf{s}_t}L$ is less prone to vanishing for the beginning layers. (Note that $\lambda_1(W_t)$ could be negative so there is still a chance that the gradient has eigenvalues smaller than 1, but the eigenvalues are more likely to be close to 1 if the weight matrices are initialized to be small.)

8 Theory of neural network generalization (CS289A only)

1. (8 points) Many empirical phenomena, well-known to deep learning practitioners, remain mysteries to theoreticians. One such mystery has been the question of generalization: why do the functions learned by neural networks generalize so well to unseen data? From the perspective of classical bias-variance trade-off in machine learning, the generalization performance of neural networks is a surprise given that they are so overparameterized that they could easily represent countless poorly-generalizing functions.

   In this problem, we will explore the following problem: given a network architecture, a target function f, and a training set of n random examples, can we efficiently predict the generalization performance of the network's learned function $\hat{f}$? This would help explain why neural networks generalize well on certain functions.

   (a) (2 points) Consider a feedforward neural network representing a function $\hat{f}_\theta : \mathcal{X} \to \mathbb{R}$, where θ is a d-dimensional parameter vector. Further consider one training example x with target value y and one test point x'. Suppose we perform one step of gradient descent with a small learning rate η with respect to the MSE loss

   $$\ell(\theta) = (\hat{f}_\theta(x) - y)^2.$$

   What is the parameter update on θ from this one step of gradient descent?

   You can write the update rule in the form of $\theta \leftarrow \theta + \delta$ and give an expression of δ in terms of the learning rate η, $\hat{f}_\theta$ at the previous parameter θ, and the training example (x, y). You may need the gradient $\nabla_\theta \hat{f}_\theta(x)$ in your expression.

   Solution:

   $$\delta = -\eta \nabla_\theta \ell_\theta = -2\eta(\hat{f}_\theta(x) - y)\nabla_\theta \hat{f}_\theta(x).$$

   This question is adapted from the following recent paper:

   Simon, James B., Madeline Dickens, and Michael R. DeWeese. "Neural Tangent Kernel Eigenvalues Accurately Predict Generalization." arXiv preprint arXiv:2110.03922 (2021).

   Learn more at the blog post: A First-Principles Theory of Neural Network Generalization.

   (b) (2 points) Now let's look at how this parameter update changes the output $\hat{f}(x')$ on the test example x'. As an approximation, we linearize $\hat{f}_{\theta+\delta}$ about θ with Taylor expansion:

   $$\hat{f}_{\theta+\delta}(x') = \hat{f}_\theta(x') + \nabla_\theta \hat{f}_\theta(x') \cdot \delta + O(\delta^2).$$

   This will give us an approximation in the form of

   $$\hat{f}_{\theta+\delta}(x') - \hat{f}_\theta(x') = -2\eta(\hat{f}_\theta(x) - y)K(x, x') + O(\delta^2).$$

   This kernel approximation approximates how the training data point (x, y) changes the prediction for the test data point x' through the kernel $K(x, x')$.

   What is the kernel $K(x, x')$? Note that $K(x, x')$ is a scalar that measures the "similarity" between x and x'.

Hint: The expression for $K(x, x')$ will depend on $\nabla_\theta \hat{f}_\theta$.

Solution:

$$\begin{aligned}\hat{f}_{\theta+\delta}(x') &= \hat{f}_\theta(x') + \nabla_\theta \hat{f}_\theta(x') \cdot \delta + O(\delta^2) && (6)\\ &= \hat{f}_\theta(x') + \nabla_\theta \hat{f}_\theta(x') \cdot \left(-2\eta(\hat{f}_\theta(x) - y)\nabla_\theta \hat{f}_\theta(x)\right) + O(\delta^2) && (7)\\ &= \hat{f}_\theta(x') - 2\eta(\hat{f}_\theta(x) - y)\nabla_\theta \hat{f}_\theta(x') \cdot \nabla_\theta \hat{f}_\theta(x) + O(\delta^2) && (8)\end{aligned}$$

Therefore,

$$K(x, x') = \nabla_\theta \hat{f}_\theta(x') \cdot \nabla_\theta \hat{f}_\theta(x).$$

(c) (1 point) In the special case where the network is only one linear layer ($\hat{f}_\theta(x) = x^T\theta$), what is the kernel $K(x, x')$?

Solution: In this case,

$$\nabla_\theta \hat{f}_\theta(x') = x' \text{ and } \nabla_\theta \hat{f}_\theta(x) = x.$$

Thus,

$$K(x, x') = \nabla_\theta \hat{f}_\theta(x') \cdot \nabla_\theta \hat{f}_\theta(x) = x' \cdot x = \langle x', x\rangle.$$

(d) (3 points) In deriving the generalization of kernel regression, we get a lot of mileage from a simple trick: we look at the learning problem in the eigenbasis of the kernel. Viewed as a linear operator, the kernel has eigenvalue and eigenfunction pairs (λ_i, ϕ_i) defined by the condition that for all $x \in \mathcal{X}$ (recall that $\mathcal{X}$ is the input domain of the neural network function $\hat{f}$),

$$\int_{x' \in \mathcal{X}} K(x, x')\phi_i(x')dx' = \lambda_i \phi_i(x).$$

The eigenvalue and eigenfunction definitions here are the generalization of the finite-dimensional definitions of eigenvalue and vector.

Suppose the input space is the one-dimensional interval $\mathcal{X} = [-1, 1]$. Consider the polynomial kernel $K(x, x') = xx' + x^2x'^2$. (This polynomial kernel arises from featurizing x as $[x, x^2]$.)

Show that $\phi_1(x) = x$ and $\phi_2(x) = x^2$ are the eigenfunctions for this kernel. What are the eigenvalues λ_1 and λ_2 for these two eigenfunctions?

Recent theoretical work tells us that higher-eigenvalue eigenfunctions are easier to learn. Comment on the implications of this result in light of the eigenvalues you computed.

Solution: Grading note: the solution proves that x and x^2 are the only two eigenfunctions (up to a constant), but the students just need show that they are eigenfunctions and get the correct eigenvalues.

With the given kernel,

$$\int_{x' \in \mathcal{X}} K(x, x')\phi_i(x')dx' = \int_{x' \in \mathcal{X}} (xx' + x^2x'^2)\phi_i(x')dx'$$

$$= \qquad x\left(\int_{x'\in\mathcal{X}} x'\phi_i(x')dx'\right) + x^2\left(\int_{x'\in\mathcal{X}} x'^2\phi_i(x')dx'\right)$$

For this to be equal to $\lambda_i\phi_i(x)$ on every x means that $\phi_i(x)$ must of the form $\phi_i(x) = ax+bx^2$. Therefore, substituting in $\phi_i(x) = ax + bx^2$, we then have that for all $x \in \mathcal{X}$,

$$x\left(\int_{x'\in\mathcal{X}} x'\phi_i(x')dx'\right) + x^2\left(\int_{x'\in\mathcal{X}} x'^2\phi_i(x')dx'\right) = \lambda_i ax + \lambda_i bx^2.$$

Since this holds for all $x \in \mathcal{X}$, we can then deduce

$$\int_{x'\in\mathcal{X}} x'\phi_i(x')dx' = \lambda_i a \text{ and } \int_{x'\in\mathcal{X}} x'^2\phi_i(x')dx' = \lambda_i b.$$

Substituting in $\phi_i(x') = ax' + bx'^2$,

$$\int_{x'\in\mathcal{X}} x'(ax' + bx'^2)dx' = \lambda_i a \text{ and } \int_{x'\in\mathcal{X}} x'^2(ax' + bx'^2)dx' = \lambda_i b$$

Now we compute the integrals for $\mathcal{X} = [-1, 1]$:

$$\int_{x'\in\mathcal{X}} x'(ax' + bx'^2)dx' = a \text{ and } \int_{x'\in\mathcal{X}} x'^2(ax' + bx'^2)dx' = \frac{2}{5}b$$

Putting together the above, we must have

$$a = \lambda_i\frac{2}{3}a \text{ and } \frac{2}{5}b = \lambda_i b \implies \lambda_i = 1 \text{ and } b = 0; \text{ or } \lambda_i = \frac{2}{5} \text{ and } a = 0$$

Therefore the eigenvalues are $\lambda_1 = \frac{2}{3}$ and $\lambda_2 = \frac{2}{5}$, and the corresponding eigenfunctions are $\phi_1(x) = x$ and $\phi_2(x) = x^2$ (the eigenfunctions are up to scaling by a constant).

According to the theory, the higher-eigenvalue linear eigenfunction $\phi_1(x) = x$ is easier to learn than the quadratic eigenfunction.