CS 189/289A Introduction to Machine Learning
Fall 2023 Jennifer Listgarten, Jitendra Malik

Midterm

- Please do not open the exam before you are instructed to do so.
- **Electronic devices are forbidden on your person**, including cell phones, tablets, headphones, and laptops. Leave your cell phone off and in a bag; it should not be visible during the exam.
- The exam is closed book and closed notes except for your one-page 8.5×11 inch cheat sheet.
- You have 1 hour and 50 minutes (unless you are in the DSP program and have a larger time allowance).
- Please write your initials at the top right of each page after this one (e.g., write "JD" if you are John Doe). Finish this by the end of your 1 hour and 50 minutes.
- Mark your answers on the exam itself in the space provided. Do **not** attach any extra sheets.
- For multiple choice questions, fill in the bubble for the single best choice.
- For short and long answer questions, write within the boxes provided. If you run out of space, you may use the last four pages to continue showing your work.
- **The last question is for CS289A students only**. Students enrolled in CS189 will **not** receive any credit for answering this question.

Your Name	
Your SID	
Name and SID of student to your left	
Name and SID of student to your right	
Doing anything fun this weekend?	
Favorite ML algorithm?	

○ CS 189

○ CS 289A

This page intentionally left blank.

initial here

1 Multiple Choice

For the following questions, select the **single best response**. Each question is worth 1.5 points.

1. Which of the following is a problem with the sigmoid activation function, in the context of deep neural networks?

   ◯ Sigmoid is prone to vanishing gradients at extreme values.

   ◯ Sigmoid can take on negative values.

   ◯ Sigmoid is non-linear, which provides less representation power.

   ◯ Sigmoid is numerically unstable when the input is large.

   Solution: The correct answer is A, since the gradient is very small at extreme values, making gradients small. B is wrong, since sigmoid's range is in [0,1]. C is wrong because sigmoid being non-linear gives the neural network *more* representation power, and it's actually an advantage. D is wrong because for large inputs, the denominator approaches 1, and dividing 1 by 1 does not lead to numerical instability.

2. Which of the following is **not** a component/feature of standard Transformer models?

   ◯ Masked decoding, which prevents attention lookups into the future.

   ◯ Transformer model training can be highly parallelized.

   ◯ Multi-head attention, which allows for attending to different parts of the sequence (e.g. long-range vs. short-range dependencies).

   ◯ The runtime complexity of attention is $O(n \log n)$, where n is the input length.

   Solution: The correct answer is D, since the runtime is actually quadratic. A and C are both features of Transformers that augment performance beyond just self-attention. B is also a feature of Transformers, which is why they're so popular in the age of deep learning (where a lot of compute can be thrown at the model to learn complex relationships in language).

3. Consider a multivariate Gaussian distribution with covariance matrix $\Sigma = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$. Which of the following correctly corresponds to the direction of the **major** axis of the Gaussian's isocontours?

   ◯ $\begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$

   ◯ $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$

   ◯ $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$

   ◯ $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Solution: The correct answer is B, as it is the eigenvector corresponding to the largest eigenvalue of Σ.

4. Suppose you train a logistic regression model with **cross-entropy loss**. Assume that you're updating the model using gradient descent with a sufficiently low learning rate. Which of the following are valid ways to initialize the parameters? Assume that numerical stability is not an issue (i.e. when running the model, you don't pass in an invalid argument to a function, such as 0 to log).

   ◯ Initialize all parameters with 0's
   ◯ Initialize all parameters with 1's
   ◯ Randomly initialize parameters
   ◯ All of the above

   Solution: The correct answer is D. You can initialize logistic regression in any way because symmetry is not an issue here. The loss function is convex.

5. Suppose we want to apply dimensionality reduction to a high-dimensional dataset X. Let $X = U\Sigma V^T$ be the singular value decomposition (SVD) of X. Which of the following regarding PCA is **false**?

   ◯ The principal components must be orthogonal to each other.
   ◯ The principal components are given by the eigenvectors of X.
   ◯ Multiplying the first k principal component scores $U_k\Sigma_k$ (where U_k is the first k columns of U and Σ_k is the top-left $k \times k$ entries of Σ) by their corresponding principal axes V_k^T (where V_k is the first k columns of V) gives us a matrix of rank k.
   ◯ The variance along the ith principal component axis is given by σ_i^2.

   Solution: The correct answer is B. The principal components are the eigenvectors of the covariance matrix of X.

6. Which of the following is **not** a feature of a standalone convolution layer? Assume that there is no pooling layer afterwards.

   ◯ Local Connectivity: The convolution layer assumes that local regions in the input are more relevant for learning features.
   ◯ Parameter Sharing: The same set of weights is used across different parts of the input in the convolution layer.
   ◯ Translational Invariance: The convolution layer is designed to produce an output that is insensitive to translations in the input.
   ◯ Channel-wise Feature Learning: In multi-channel inputs (like color images), each filter can learn features that are channel-specific.

   Solution: The correct answer is C. A standalone convolution layer is translationally equivariant, not invariant. This means that if the input is translated, the output will also be translated in the same way. Translational invariance is approximately introduced when max-pooling layers are added after convolution layers.

initial here

7. Consider adding the elastic net regularization term $\lambda_1\|w\|_2^2 + \lambda_2\|w\|_1$ to linear regression. If we want our weights to be smaller, but dislike having many completely zero weights, what is most likely to be the best choice?

   ◯ Increase λ_1

   ◯ Increase λ_2

   ◯ Decrease λ_1

   ◯ Decrease λ_2

   Solution: The correct answer is A. We have to increase one of λ_1, λ_2, since their terms penalize large weights. Furthermore, λ_2 controls how "pointy" our contour is, which would encourage zero weights. Therefore, increasing λ_1 is likely to be the best choice for our scenario.

8. PCA and t-SNE are techniques often used for representing high-dimensional data in a lower-dimensional space. Which of the following is true regarding these two techniques? Assume that PCA is run without a basis expansion (e.g. a polynomial basis expansion).

   ◯ Both PCA and t-SNE can accurately capture non-linear relationships in the data.

   ◯ t-SNE aims to minimize the pairwise distances in the data whereas PCA aims to maximize the variance of the data.

   ◯ Both PCA and t-SNE have convex optimization problems.

   ◯ t-SNE aims to maximize the pairwise distances in the data whereas PCA aims to minimize the variance of the data.

   Solution: The correct answer is B. Choice A is wrong since PCA is linear. Choice C is wrong since t-SNE is non-convex. Choice D is the opposite of what is correct.

9. Consider the optimization problem

$$\min_w \ \left\|Xw - y\right\| + g(w)$$

   for some $g(w)$. We assume that $P(y|x; w) = N(w^\top x, \sigma^2)$. The minimizer w^* of this problem can be thought of as a MAP solution with which prior $P(w)$?

   ◯ $\frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-g(w)^2}{2\sigma^2}\right)$

   ◯ $\frac{\exp\left(-g(w)\right)}{\int_{w'} \exp\left(-g(w')\right) dw'}$

   ◯ $g(w)\exp(-\|w\|)$

   ◯ $\frac{1}{2\sigma} \exp\left(\frac{-|g(w)|}{\sigma}\right)$

   Solution: The correct answer is B. We know that

$$\begin{aligned} w_{MAP} &= \operatorname*{argmax}_w \ P(D|w)P(w) \\ &= \operatorname*{argmax}_w \ \left(\prod_{i=1}^{n} P(y_i|x_i; w)\right) P(w) \end{aligned}$$

initial here

$$= \underset{w}{\text{argmax}} \ \log\left[\left(\prod_{i=1}^{n} P(y_i|x_i;w)\right)P(w)\right]$$
$$= \underset{w}{\text{argmax}} \ \left(\sum_{i=1}^{n} \log(P(y_i|x_i;w))\right) + \log(P(w))$$

Because $P(y_i|x_i;w) = N(w^\top x_i, \sigma^2)$, this is

$$\underset{w}{\text{argmax}} \ \left[\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(y_i - w^\top x_i)^2}{2\sigma^2}\right)\right)\right] + \log(P(w))$$
$$= \underset{w}{\text{argmax}} \ \left[\sum_{i=1}^{n} \frac{-(y_i - w^\top x_i)^2}{2\sigma^2}\right] + \log(P(w))$$
$$= \underset{w}{\text{argmin}} \ \left[\sum_{i=1}^{n} \frac{(y_i - w^\top x_i)^2}{2\sigma^2}\right] - \log(P(w))$$
$$= \underset{w}{\text{argmin}} \ \left[\sum_{i=1}^{n} (y_i - w^\top x_i)^2\right] - \log(P(w))$$
$$= \underset{w}{\text{argmin}} \ ||Xw - y|| - \log(P(w))$$

So $-\log(P(w)) = g(w)$. This means that $P(w) = \exp(-g(w))$. However, we are missing a normalization factor because $P(w)$ is a probability distribution. Thus,

$$P(w) = \frac{\exp(-g(w))}{\int_{w'} \exp(-g(w'))\, dw'}$$

The denominator is a constant that does not affect the minimization.

10. When training a neural network with attention blocks, which of the following would most likely cause outputs to become NaNs? Assume numerical instability is not an issue in the inputs.

    ◯ Having large variation in your attention scores. Recall attention scores are dot products between projected keys and queries.

    ◯ Not normalizing keys and queries before taking their dot product. This means keys and queries may not have norm 1.

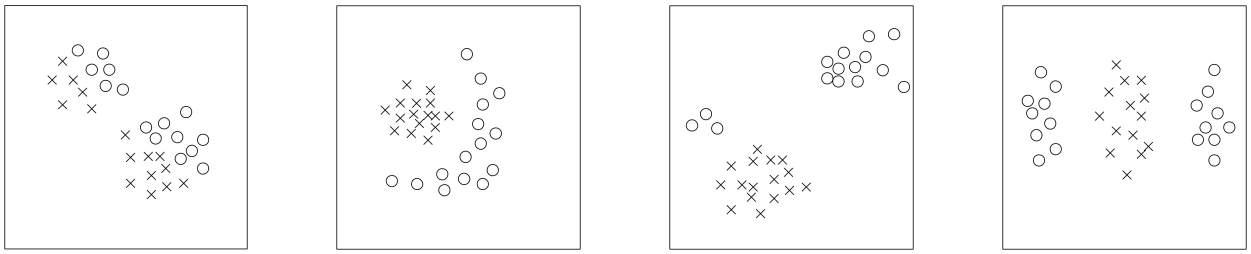
    ◯ Apply a mask to all of our keys. Masking prevents the model from attending to certain keys.

    ◯ Randomly initializing our neural network weights with i.i.d. random variables that have unbounded expectation.

    Solution: The correct answer is C. If we mask out all of our keys, it is impossible to normalize our scaled-dot products into a probability distribution to weight our values with. A and B are fine, but may restrict gradient flow because of the softmax. D doesn't really matter as just because this random variable has unbounded expectation, when sampling it's very unlikely we'll get a sample that causes numeric instability.

initial here

11. Which of the following k-means cluster assignments could be a possible result after running k-means to convergence for 2 clusters?



(a) (b) (c) (d)

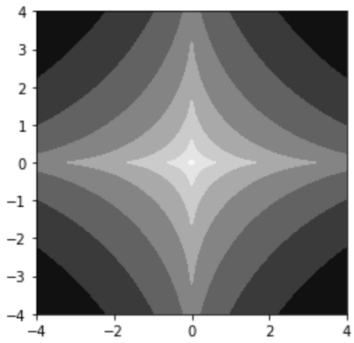
◯ (a)
◯ (b)
◯ (c)
◯ (d)

Solution: The correct answer is A. This could be a result of k-means if it is stuck at a local minimum, as all of the points are closer to their cluster centers than the other cluster's center. Choice B is incorrect because the decision boundary is not linear. Choice C is incorrect because the outliers are clearly closer to the center of x than o. Choice D is incorrect because the decision boundary is, again, not linear.

initial here

2 Short Answer

1. In class you learned about regularizing linear regression with an L_1 penalty (LASSO). Now, instead consider an $L_{0.5}$ penalty on the weights. A plot of the isocontours of the $L_{0.5}$ norm is shown below:



How will the sparsity of the $L_{0.5}$ penalized linear regression compare to that of LASSO? Select the best answer choice **and** explain your answer.

- ◯ $L_{0.5}$ will have more sparse solutions than LASSO.
- ◯ $L_{0.5}$ will have less sparse solutions than LASSO.

Solution: $L_{0.5}$ penalized regression will have sparser solutions, as its "corners" are more pronounced than those in the L_1 contours.

2. We have a dataset with binary labels $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ where $y_n = \{0, 1\}$. Write the formula for the binary cross entropy loss between a model's predicted probabilities for the positive class $\{p_\theta(y = 1|x_n)\}_{n=1}^N$ and the true labels $\{y_n\}_{n=1}^N$. Assume that the loss is for the whole dataset. Please box your final answer.

Solution: The binary cross entropy is:

$$-\frac{1}{N}\sum_{n=1}^{N}[y_n \log p_\theta(y = 1|x_n) + (1 - y_n)\log(1 - p_\theta(y = 1|x_n))]$$

3. Suppose you have two coins. Coin A has probability p of landing heads, and coin B has probability $3p$ of landing heads. Suppose you flip both coins and get the following result:
Coin A: T
Coin A: H
Coin B: H
Coin B: H
Coin A: T
Coin B: H

What is the maximum likelihood estimate for p? Please box your final answer.

Solution: The likelihood function is: $L(p) = P(\text{coin A}|T)^2 * P(\text{coin A}|H) * P(\text{coin B}|H)^3 = (1-p)^2 p(3p)^3 = (1-p)^2 p(27p^3) = 27p^4(1-p)^2$.

initial here

The log likelihood function is: $l(p) = \log(27) + 4\log(p) + 2\log(1-p)$.
The maximum likelihood estimate of p is: $\frac{\partial l(p)}{\partial p} = 0 \implies \frac{4}{p} + \frac{-2}{1-p} \implies p = \frac{2}{3}$.

We also accepted $p = \frac{1}{3}$ as a valid answer, since you could get either answer depending on which coin you perform MLE with relation to. As long as your MLE work was correct, we gave full credit.

4. For your latest Transformer model, you have devised this positional encoding scheme:

$$PE_{(x,i)} = \left(\frac{x \pmod{189}}{189}\right)^{i/d_{model}}$$

where x is the position, i is the index of the feature dimension, and d_{model} is the total number of dimensions in an input embedding. This value is added to index i of the x-th token's embedding vector.

Describe in one or two short sentences what problems you may encounter with this encoding, as well as what inputs you would encounter them on.

Solution: If the input has more than 189 tokens, the positional encoding for tokens in the same position $\pmod{189}$ will be seen by the model as the same position.

5. Suppose that $X \in \mathbb{R}^3$ is a random vector with a multivariate Gaussian distribution that can be written as $AZ + \mu$, where $Z \sim N(0, I_3)$, $\mu = [1,1,1]^T$ and $A = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$. Compute the correlation matrix C corresponding to X. In a correlation matrix, entry $C_{ij} = \frac{\text{cov}(X_i, X_j)}{\sqrt{\text{var}(X_i)\text{var}(X_j)}}$, where X_i, X_j are the i and jth entries of X. Please box your final answer.

Solution: Recall that AA^T is the covariance matrix of X, which will give us all of the pairwise covariances. To create the correlation matrix, we also need to compute the variances for each X_i, so that we can find the denominators.

We know that $X_1 = Z_1 + 2Z_2 + 0Z_3 + 1$. Since Z_1, Z_2, Z_3 are all independent and have variance 1, we can compute $\text{var}(X_1) = 1^2 + 2^2 + 0^2 = 5$. Similarly, $\text{var}(X_2) = \text{var}(X_3) = 5$. Therefore the denominators of each entry of the correlation matrix are all the same, and we can calculate it as

$$\frac{1}{\sqrt{5^2}} AA^T = \frac{1}{5}\begin{bmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}\begin{bmatrix} 1 & 2 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} = \frac{1}{5}\begin{bmatrix} 5 & 2 & 2 \\ 2 & 5 & 2 \\ 2 & 2 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0.4 & 0.4 \\ 0.4 & 1 & 0.4 \\ 0.4 & 0.4 & 1 \end{bmatrix}$$

initial here

6. In this problem, we consider a solution to the vanishing gradient problem: residual connections.

   To examine this phenomenon, let's consider a slightly different residual layer than what was presented in lecture. Our layer processes vector inputs $x_{in} \in \mathbb{R}^d$. Let $W \in \mathbb{R}^{d \times d}$ and $b \in \mathbb{R}^d$. Mathematically, a layer is represented as:

   $$x_{out} = x_{in} + \max(Wx_{in} + b, 0)$$

   Compute the Jacobian $\frac{\partial x_{out}}{\partial x_{in}}$ for this residual block and explain why this prevents the gradients from dying at this layer. For this problem, you may assume that $Wx_{in} + b \neq 0$, and therefore we don't need to worry about the derivative at the non-differentiable point. Box your answer for $\frac{\partial x_{out}}{\partial x_{in}}$.

   Hint: Start by deriving the formula for entry (i, j) of the Jacobian: $\frac{\partial x_{out,i}}{\partial x_{in,j}}$. Then, assemble the Jacobian matrix.

   Solution: Let's compute the derivative element-wise. Note: the max function isn't differentiable when both parameters are equal, but we don't worry about this case.

   $$\begin{aligned}\frac{\partial x_{out,i}}{\partial x_{in,j}} &= \frac{\partial}{\partial x_{in,j}}\left(x_{in,i} + \max\left(\sum_k (W_{i,k}x_k) + b, 0\right)\right) \\ &= \mathbf{1}[i = j] + \mathbf{1}\left[\sum_k \left(W_{i,k}x_{in,k}\right) + b_j > 0\right] W_{i,j}\end{aligned}$$

   Composing these into a matrix, we get:

   $$I_d + M_{ind} \circ W^T$$

   Where, M_{ind} is the matrix of indicators. Because we have the identity matrix component of the gradient, even if the max always zero, the gradient can still flow up the network.

7. Say we have four points: (0, 1), (0, -1), (1, 1), (1, -1). Using PCA, find the first principal direction.

   Solution:

   After centering our data matrix, we have $\bar{X} = \begin{bmatrix} -\frac{1}{2} & 1 \\ -\frac{1}{2} & -1 \\ \frac{1}{2} & 1 \\ \frac{1}{2} & -1 \end{bmatrix}$

   Recall that the principal directions are the eigenvectors of $\frac{1}{n}\bar{X}^T\bar{X}$.

   In this case, $\frac{1}{n}\bar{X}^T\bar{X} = \frac{1}{4}\begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & 1 \end{bmatrix}$. Since this is a diagonal matrix, the eigenvectors and thus the principal components would be $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ (with an eigenvalue of $\frac{1}{4}$) and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ (with an eigenvalue of 1). Our final answer would then be the vector $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ since it has the bigger eigenvalue.

initial here

8. Let $f(x) = Wx$ be parameterized by a $d \times d$ matrix W. Suppose a data point $x \in \mathbb{R}^d$ has a label $y \in \mathbb{R}^d$ associated with it. We use the L_2 loss function $L(W) = \|f(x) - y\|_2^2 = \|Wx - y\|_2^2$ to describe the error between our prediction $f(x)$ and the ground truth label y. Recall that a gradient update step for W looks like $W^{(t+1)} = W^t - \alpha * \nabla_W L(W^t)$, where α is our learning rate. Let us now consider the problem of selecting an optimal α. We can do this by solving the following optimization problem:

$$\alpha^\star = \min_\alpha \; L(W - \alpha \nabla_W L(W))$$

What is $\alpha^\star$ in this case? Please box your final answer for $\alpha^\star$.

Hint 1: Recall from Homework 1 that $\nabla_W L(W) = 2(Wx - y)x^\top$.

Hint 2: Performing a change of variables $z = Wx - y$ may help to simplify algebra.

Solution: Using the value for $\nabla_W L(W)$, the objective function is

$$\min_\alpha \; L(W - 2\alpha(Wx - y)x^\top)$$

Plugging into $L(W)$, we have

$$\min_\alpha \; \|(W - 2\alpha(Wx - y)x^\top)x - y\|_2^2 = \min_\alpha \; \|Wx - y - 2\alpha(Wx - y)x^\top x\|_2^2$$

Let us write $\epsilon(W) = Wx - y$. Then this is

$$\min_\alpha \; \|\epsilon(W) - (2\alpha x^\top x)\epsilon(W)\|_2^2 = \min_\alpha \; (1 - 2\alpha x^\top x)^2 \|\epsilon(W)\|_2^2$$

We take the derivative of the objective with respect to α and set it to 0:

$$\frac{d}{d\alpha}(1 - 2\alpha x^\top x)^2 \|\epsilon(W)\|_2^2 = 2\|\epsilon(W)\|_2^2 (1 - 2\alpha x^\top x) \cdot (-2x^\top x)$$

So

$$\begin{aligned} & -4\|\epsilon(W)\|_2^2 \|x\|_2^2 (1 - 2\alpha \|x\|_2^2) = 0 \\ \implies \alpha^\star = & \frac{4\|\epsilon(W)\|_2^2 \|x\|_2^2}{-4\|\epsilon(W)\|_2^2 \|x\|_2^2 * -2\|x\|_2^2} = \frac{1}{2\|x\|_2^2} \end{aligned}$$

9. Consider the function $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ whose components are given by

$$f_i(x) = \frac{x_i^2}{\sum_{k=1}^d x_k^2}$$

for $i = 1, \ldots, d$ and $x \in \mathbb{R}^d$.

(a) Find an expression for $\frac{\partial f_i}{\partial x_j}$. *Hint: It may help to consider two cases: $i = j$ and $i \neq j$.*

Solution: Following the hint, we divide the computation into two cases:

initial here

- $i = j$

  In this case, we have

$$\frac{\partial f_i}{\partial x_i} = \frac{\partial}{\partial x_i} \frac{x_i^2}{\sum_{k=1}^d x_k^2} = \frac{2x_i(\sum_{k=1}^d x_k^2) - x_i^2(2x_i)}{(\sum_{k=1}^d x_k^2)^2} = \frac{2x_i}{\sum_{k=1}^d x_k^2}\left(1 - \frac{x_i^2}{\sum_{k=1}^d x_k^2}\right)$$

- $i \neq j$

  In this case, we have

$$\frac{\partial f_i}{\partial x_j} = \frac{\partial}{\partial x_j} \frac{x_i^2}{\sum_{k=1}^d x_k^2} = \frac{2x_j}{\sum_{k=1}^d x_k^2}\left(-\frac{x_i^2}{\sum_{k=1}^d x_k^2}\right)$$

Thus,

$$\frac{\partial f_i}{\partial x_j} = \frac{2x_j}{\sum_{k=1}^d x_k^2}(1[i = j] - f_i)$$

(b) We can see that $0 \leq f_i \leq 1$ for each $i = 1, \ldots, d$ and $\sum_{i=1}^d f_i = 1$. Thus, f normalizes the input vector x to a probability distribution, just like Softmax does! That said, what is one way in which our function f graphically differs from the Softmax function?

Solution: Let $\sigma : \mathbb{R}^d \mapsto \mathbb{R}^d$ be the softmax function. We can see that softmax preserves order, i.e., $x_j > x_i \Leftrightarrow \sigma_j(x) > \sigma_i(x)$ since $\exp(x)$ is monotonically increasing. On the other hand, our new function f is only monotonic in magnitude, i.e., $|x|_j > |x|_i \Leftrightarrow f_j(x) > f_i(x)$ (it's possible for $x_j < x_i$ but $f_j(x) > f_i(x)$). This is because x^2 is an even function that is monotonically decreasing for $x < 0$ and (symmetrically) monotonically increasing for $x > 0$.

initial here

3 PCA Fundamentals

Principal Component Analysis (PCA) is a commonly used technique for dimensionality reduction. Suppose we are given a set of data points $\mathcal{D} = \{x_n \in \mathbb{R}^D : n = 1, ..., N\}$. PCA can be viewed as a change of basis that represents each data point x_n as a weighted combination of a new set of basis functions $w_1, ..., w_L \in \mathbb{R}^D$, where the weights are given by $z_n \in \mathbb{R}^L$. That is, each x_n is approximated by $\sum_{k=1}^{L} z_{nk} w_k$. Here we explore one of the many aspects of this change of basis operation.

(a) As a warm up exercise, show that the sample covariance matrix Σ is PSD, where

$$\Sigma = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T$$

and $\mu = \frac{1}{N} \sum_{n=1}^{N} x_n$ is the sample mean.

Solution: Given any non-zero vector $v \in \mathbb{R}^D$, we have:

$$\begin{aligned} v^T \Sigma v &= \frac{1}{N} \sum_{n=1}^{N} v^T (x_n - \mu)(x_n - \mu)^T v \\ &= \frac{1}{N} \sum_{n=1}^{N} [(x_n - \mu)^T v]^2 \geq 0 \end{aligned}$$

(b) One way to view the change of basis that PCA performs is that it is the optimal solution that minimizes the average reconstruction error of the data:

$$\mathcal{L}(W, Z) = \frac{1}{N} \left\| X - ZW^T \right\|_F^2 = \frac{1}{N} \sum_{n=1}^{N} \|x_n - W z_n\|^2$$

where $X \in \mathbb{R}^{N \times D}$ is the data matrix, $W \in R^{D \times L}$ is the matrix of "latent factors", and $Z \in \mathbb{R}^{N \times L}$ is the matrix of "latent vectors".

Suppose we wish to minimize the error, subject to the constraint that W is an orthogonal matrix. We will first find the optimal Z and then find the optimal W when $L = 1$. Express and expand out the reconstruction error as a function of w_1 and its associated coefficients $z_1 = [z_{11}, ..., z_{N1}] \in \mathbb{R}^N$, and show the optimal solution for each entry of z_1, z_{n1}, is $z_{n_1} = w_1^T x_n$.

Solution: The reconstruction error is given by:

$$\begin{aligned} \mathcal{L}(w_1, z_1) &= \frac{1}{N} \sum_{n=1}^{N} \|x_n - z_{n1} w_1\|^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} (x_n - z_{n1} w_1)^T (x_n - z_{n1} w_1) \end{aligned}$$

$$= \frac{1}{N}\sum_{n=1}^{N}(x_n^T x_n - 2z_{n1}w_1^T x_n + z_{n1}^2 w_1^T w_1)$$
$$= \frac{1}{N}\sum_{n=1}^{N}(x_n^T x_n - 2z_{n1}w_1^T x_n + z_{n1}^2)$$

where the $w_1^T w_1 = 1$ since W is orthogonal. Take the derivative with respect to each entry of z_1 and set to zero:

$$\frac{\partial}{\partial z_{n1}}\mathcal{L}(w_1, z_1) = \frac{1}{N}(-2w_1^T x_n + 2z_{n1}) = 0$$

Solving for z_{n1}, we obtain $z_{n_1} = w_1^T x_n$, or in matrix form $z_1 = Xw_1$.

(c) Given the optimal solution of z_n, show the optimal solution for w_1 corresponds to finding the largest eigenvalue of some matrix. What is the matrix in the case in which the data is centered, i.e. the sample mean $\mu = 0$? **Hint**: This will require using constrained optimization.

Solution: Substituting $w_1^T x_n = z_{n1}$ back to the objective, we have:

$$\mathcal{L}(w_1) = \frac{1}{N}\sum_{n=1}^{N}(x_n^T x_n - z_{n1}^2) = C - \frac{1}{N}\sum_{n=1}^{N} z_{n1}^2$$

where C is a constant with respect to w_1. Dropping the constant term, we have:

$$-\frac{1}{N}\sum_{n=1}^{N} z_{n1}^2 = -\frac{1}{N}\sum_{n=1}^{N} w_1^T x_n x_n^T w_1 = -w_1^T S w_1$$

where $S = \frac{1}{N}\sum_{n=1}^{N} x_n x_n^T$.
We optimize for w_1 subject to the constraint that $\|w_1\|_2 = 1$. With Lagrange multiplier, the objective can be expressed as:

$$\underset{w_1}{\text{argmin}}\ -w_1^T S w_1 + \lambda(w_1^T w_1 - 1)$$

Take derivative of the objective and set to zero:

$$\frac{\partial}{\partial w_1}\mathcal{L}(w_1) = -2Sw_1 + 2\lambda w_1 = 0$$
$$Sw_1 = \lambda w_1$$

The optimal solution of w_1 corresponds to finding an eigenvector of S with an associated eigenvalue λ. Since we want to minimize the loss, we want to maximize $w_1^T S w_1$. Left multiply by w_1, we see that

$$w_1^T S w_1 = \lambda$$

since $w_1^T w_1 = 1$. So the optimal solution corresponds to finding the eigenvector of S with the largest eigenvalue. In the case in which the data is centered, S corresponds to the sample covariance matrix Σ.

initial here

(d) It is often said that the optimal solution for w_1 that minimizes the reconstruction error corresponds to maximizing the sample variance of the projected data. Show that this is the case when the data is centered. Give a brief explanation for what might go wrong when the data is not centered.

Solution: When the data is centered, we have, for any n:

$$\mathbb{E}[z_{n1}] = \mathbb{E}[x_n^T w_1] = \mathbb{E}[x_n]^T w_1 = 0$$

So the sample variance is:

$$\text{Var}[z_1] = \frac{1}{n}\sum_{n=1}^{N}(z_{n1} - \mathbb{E}[z_{n1}])^2 = \frac{1}{n}\sum_{n=1}^{N} z_{n1}^2 = -\mathcal{L}(w_1) + C$$

Hence, minimizing $\mathcal{L}(w_1)$ corresponds to maximizing $\text{Var}[z_1]$.

Regardless of whether the data is centered, the objective corresponds to maximizing $\frac{1}{N}\sum_{n=1}^{N} z_{n1}^2$, i.e. maximizing the sum-of-squared deviation of the projected data from the origin. When the data is not mean centered, then the first principal component may capture the mean of the data, making the statistics obtained from PCA misleading.

initial here

4 Motivating Logistic Regression

In class, you learned that a common justification for logistic regression is the fact that Gaussian class-conditional probability densities result in a posterior probability $p(y|x)$ that takes the form of a logistic function with a linear argument, assuming the Gaussians all have the same variances. In this question, we will explore the case when this assumption does not hold true.

Consider the case in which we perform binary classification for two classes $y = 0$ and $y = 1$. Suppose that we know both classes have Gaussian class-conditionals, i.e. $P(X \mid Y = 0) \sim N(\mu_1, \sigma_1)$ and $P(X \mid Y = 1) \sim N(\mu_2, \sigma_2)$. For simplicity, we assume that the covariance matrix for each class is diagonal with the same variance along each dimension (but still with differing variances between the two classes). Recall the Gaussian PDF formula:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

We also know the two classes have prior probabilities $P(Y = 0) = \pi_1$ and $P(Y = 1) = 1 - \pi_1$.

(a) Show that the posterior distribution $P(Y = 0 \mid X)$ is a logistic function with a quadratic argument $\alpha\|x\|^2 + \beta^T x + \gamma$. What are α, β, and γ? Box your answers for each.

Solution: First apply Bayes rule:

$$P(Y = 0|X) = \frac{P(X|Y = 0)\pi_1}{P(X|Y = 0)\pi_1 + P(X|Y = 1)(1 - \pi_1)}$$

$$P(Y = 0|X) = \frac{\exp\left(\frac{-(x-\mu_1)^2}{2\sigma_1^2}\right)}{\exp\left(\frac{-(x-\mu_1)^2}{2\sigma_1^2}\right) + \frac{\sigma_1(1-\pi_1)}{\sigma_2\pi_1}\exp\left(\frac{-(x-\mu_2)^2}{2\sigma_2^2}\right)}$$

$$P(Y = 0|X) = \frac{1}{1 + \frac{\sigma_1(1-\pi_1)}{\sigma_2\pi_1}\exp\left(\frac{-(x-\mu_2)^2}{2\sigma_2^2} + \frac{(x-\mu_1)^2}{2\sigma_1^2}\right)}$$

$$P(Y = 0|X) = \frac{1}{1 + \exp\left(-\left(\left(\frac{1}{\sigma_2^2} - \frac{1}{\sigma_1^2}\right)x^2 + 2\left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right)^T x + \left(\frac{\mu_2^2}{\sigma_2^2} - \frac{\mu_1^2}{\sigma_1^2}\right) + \log\frac{\sigma_2\pi_1}{\sigma_1(1-\pi_1)}\right)\right)}$$

Reading off the constants, we have that $\alpha = \left(\frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}\right), \beta = \left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right)$ and $\gamma = \left(\frac{\mu_2^2}{\sigma_2^2} - \frac{\mu_1^2}{\sigma_1^2}\right) + \log\frac{\sigma_2\pi_1}{\sigma_1(1-\pi_1)}$.

(b) Consider the decision boundary, i.e. the point at which $P(Y = 1 \mid X) = P(Y = 0 \mid X)$. Assume that $\pi_1 = 0.5$, i.e. the prior probabilities of the two classes are the same. How does the shape of the decision boundary compare qualitatively when the two class variances are the same vs. not the same? 1-2 sentences is enough.

Solution: When the classes have the same variance, the decision boundary is a linear surface. When the classes have a different variance, the decision boundary is a quadratic.

initial here

5 Laplacian Discriminant Analysis

You are playing a fun game with Alice and Bob: in each turn, either Alice or Bob sends you a number, and you have to guess who sent it!

You know the following about Alice and Bob's behavior:

- The probability that Alice sends the number is π_A and the probability that Bob sends the number is $\pi_B = 1 - \pi_A$.
- Each person generates numbers from an independent Laplace distribution, with parameters (μ_A, b) for Alice, and (μ_B, b) for Bob. (Note that the second parameter is the same for both.)
- $\mu_A > \mu_B$.

The Laplace distribution is a continuous distribution. It is parameterized with two values (μ, b), and its probability density function is as follows:

$$f(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$

(a) You receive a number x and you want to determine if it's more likely to be from Alice or Bob. What kind of classifier are you trying to build?

- ◯ Generative
- ◯ Discriminative

Solution: Generative, because you are modeling a probability distribution for both classes and learning parameters that best fit this distribution.

(b) Compute the probability that the number is from Alice. Box your final answer.

Your answer should be expressed in the form $\frac{1}{1+\exp(-z)}$, where z is some expression that may (or may not) be in terms of $\pi_A, \pi_B, \mu_A, \mu_B, x, b$, and constants.

Solution: We know the prior probabilities, now we just have to compute the posterior using Bayes rule.

$$\begin{aligned}
p(A|x) &= \frac{p(x \cap A)}{p(x)} \\
&= \frac{f_A(x)p(A)}{f_A(x)p(A) + f_B(x)p(B)} \\
&= \frac{\frac{1}{2b}\exp\left(-\frac{|x-\mu_A|}{b}\right) \times \pi_A}{\frac{1}{2b}\exp\left(-\frac{|x-\mu_A|}{b}\right) \times \pi_A + \frac{1}{2b}\exp\left(-\frac{|x-\mu_B|}{b}\right) \times (1-\pi_A)} \\
&= \frac{1}{1 + \frac{1-\pi_A}{\pi_A}\exp\left(-\frac{|x-\mu_B|-|x-\mu_A|}{b}\right)} \\
&= \frac{1}{1 + \exp\left(-\left(\frac{|x-\mu_B|-|x-\mu_A|}{b} - \ln\left(\frac{1-\pi_A}{\pi_A}\right)\right)\right)}
\end{aligned}$$

initial here

Thus,

$$z = \frac{|x - \mu_B| - |x - \mu_A|}{b} - \ln\left(\frac{1 - \pi_A}{\pi_A}\right)$$

(c) You decide to play a new game. All the conditions stay the same, except this time, Alice and Bob both send you a number, and you have to match the two numbers to their senders! You receive two numbers: $x_1 > x_2 > \mu_A$. To maximize the probability of being correct, which one should you guess to be Alice's number? Justify your answer.

Solution: When $x \geq \mu_A$, $z = \frac{\mu_A - \mu_B}{b} - \ln\left(\frac{1-\pi_A}{\pi_A}\right)$, which turns out to be a constant value. Thus, $p(A|x_1) = p(A|x_2) = \frac{1}{1+\exp(-z)}$ is the same for both x_1 and x_2, making neither of them more likely to be from Alice.

initial here

6 Machine Unlearning: Linear Regression

We often care about validating our model on data that wasn't used to train it. We've seen that K-fold cross validation can be a powerful way to ensure that our models are robust to dropping out chunks of the data.

In this question, we're going to engage with a version of K-fold validation called *leave-one-out* validation. Let's say we have a design matrix $X \in \mathbb{R}^{n \times d}$ and a corresponding set of labels $Y \in \mathbb{R}^n$. As the name suggests, we train n different models leaving out a single data point each time and then test our model on this held out datapoint.

Typically, we denote the model fit on all the data except the ith point with a subscript $[i]$, so the model trained without (X_i, Y_i) is $f_{[i]}$. The leave-one-out error is therefore defined as

$$J_{loo}(X, Y, f) = \frac{1}{n} \sum_i \text{loss}(f_{[i]}(X_i), Y_i)$$

This kind of validation is quite cumbersome for most models, but we'll see for linear models and the squared error loss it can actually be computed efficiently. We'll then explore the consequences of this fact for updating linear models with new data.

(a) First compute $(X_{[i]}^T X_{[i]})^{-1}$ in terms of $(X^T X)^{-1}$ and X_i, the ith row of X. You may use the Sherman-Morrison identity without proof, which is: Let $A \in \mathbb{R}^{k \times k}$ and $u, v \in \mathbb{R}^k$, then

$$\left(A + uv^\intercal\right)^{-1} = A^{-1} - \frac{A^{-1}uv^\intercal A^{-1}}{1 + v^\intercal A^{-1}u}.$$

Hint: Try writing $X^T X$ in outer product form.

Solution: We start out following the hint:

$$X^T X = \sum_j X_j X_j^T$$

Then using this decomposition, we can see that removing the ith datapoint is equivalent to subtracting $X_i X_i^T$ from $X^T X$. We then plug this in, and use Sherman-Morrison

$$\begin{aligned}(X_{[i]}^T X_{[i]})^{-1} &= (X^T X - X_i X_i^T)^{-1} \\ &= (X^T X)^{-1} + \frac{(X^T X)^{-1} X_i X_i^T (X^T X)^{-1}}{1 - X_i^T (X^T X)^{-1} X_i}\end{aligned}$$

(b) Write out the closed form solution for $\beta_{[i]}$, the coefficients of a linear model trained without the ith datapoint. Your answer should be in terms of β (the coefficients fitted on all the data), X_i, $e_i = X_i^T (X^T X)^{-1} X^T Y - Y_i$, $(X^T X)^{-1}$, and $h_i = X_i^T (X^T X)^{-1} X_i$.

Hint: Begin by writing out the closed form solution for $\beta_{[i]}$, then plugging in part (a).

Solution: We can use a similar trick to the outer product for $X^T Y$ to get $X_{[i]}^T Y_{[i]}$.

$$X_{[i]}^T Y_{[i]} = (X^T y - X_i y_i)$$

initial here

Next, let $h_i = X_i^T (X^T X)^{-1} X_i$. In statistics, this h_i is called a leverage score. We then compute:

$$\begin{aligned}
\beta_{[i]} &= (X_{[i]}^T X_{[i]})^{-1} X_{[i]}^T Y_{[i]} \\
&= \left[(X^T X)^{-1} + \frac{(X^T X)^{-1} X_i X_i^T (X^T X)^{-1}}{1 - h_i} \right] (X^T y - X_i y_i) \\
&= (X^T X)^{-1} X^T y - (X^T X)^{-1} X_i y_i + \frac{(X^T X)^{-1} X_i X_i^T (X^T X)^{-1}}{1 - h_i} (X^T y - X_i y_i) \\
&= \beta - (X^T X)^{-1} X_i y_i + \frac{(X^T X)^{-1} X_i X_i^T (X^T X)^{-1}}{1 - h_i} (X^T y - X_i y_i)
\end{aligned}$$

We then note that the prediction on the ith data point (which we are removing) is $\hat{y}_i = X_i^T (X^T X)^{-1} X^T y$ and the training error on this point $e_i = \hat{y}_i - y_i$.

$$\begin{aligned}
\beta_{[i]} &= \beta - (X^T X)^{-1} X_i y_i + \frac{1}{1 - h_i} (X^T X)^{-1} X_i \hat{y}_i - \frac{h_i}{1 - h_i} (X^T X)^{-1} X_i y_i \\
&= \beta - \frac{1}{1 - h_i} (X^T X)^{-1} X_i \hat{y}_i - \frac{1}{1 - h_i} (X^T X)^{-1} X_i y_i \\
&= \beta - (X^T X)^{-1} X_i \frac{\hat{y}_i - y_i}{1 - h_i} \\
&= \beta - (X^T X)^{-1} X_i \frac{e_i}{1 - h_i}
\end{aligned}$$

(c) Finish by computing the leave-one-out error $e_{[i]}$ in terms of the original training error $e_i = \hat{Y}_i - Y_i$ and h_i.

Solution:

Plugging in the previous part, we get

$$\begin{aligned}
e_{[i]} &= X_i^T \beta_{[i]} - y_i \\
&= X_i^T \left(\beta - (X^T X)^{-1} X_i \frac{e_i}{1 - h_i} \right) - y_i \\
&= \hat{y}_i - y_i - \frac{h_i}{1 - h_i} e_i \\
&= \frac{e_i}{1 - h_i}
\end{aligned}$$

(d) Use parts (a) and (b) to show how you could add a single datapoint to an existing OLS fit given just the new point X_{n+1}, an updated covariance matrix $(X_{n+1}^T X_{n+1})^{-1} = (X^T X + X_{n+1} X_{n+1}^T)^{-1}$, and the original fit β.

Solution: Using parts (a) and (b), we know that $\beta_{[i]} = \beta - (X^T X)^{-1} X_i \frac{e_i}{1 - h_i} = \beta - (X^T X)^{-1} X_i e_{[i]}$. Rearranging terms, we get

$$\beta_{[i]} + (X^T X)^{-1} X_i e_{[i]} = \beta$$

initial here

We can then rename our variables to suggest that we are now adding a new point rather than removing one.

$$\beta_n + (X_{n+1}^T X_{n+1})^{-1} X_{n+1} e^{[i]} = \beta_{n+1}$$

This formula is known as recursive least squares. You may also wonder how we can efficiently compute $(X_{n+1}^T X_{n+1})^{-1}$. As it turns out, this is another application of the Sherman-Morrison identity.

initial here

7 Xavier Initialization for Neural Networks (CS 289A Only)

When you optimize a neural network's weights using Gradient Descent, you must start with an initial guess for your model parameters that is then iteratively updated during training. Before the advent of modern deep learning techniques like Batch Normalization, the way in which these parameters were initialized could significantly impact model training: more often than not, poor model performance was attributed to poor weight initialization. In this problem, we will think about a scheme for initializing model weights that leads to stable training.

Consider a feed forward neural network with L layers. Assume that we use the activation function $\sigma(\cdot)$ after each layer. Suppose the input to layer l is the $n^{[l-1]}$-dimensional vector $x^{[l-1]}$ and let this layer have an $n^{[l]} \times n^{[l-1]}$ weight matrix $W^{[l]}$ and $n^{[l]}$-dimensional bias vector $b^{[l]}$. Then, the forward propagation equations for layer l are

$$z^{[l]} = W^{[l]} x^{[l-1]} + b^{[l]}$$
$$x^{[l]} = \sigma(z^{[l]})$$

By convention, we let $x^{[0]}$ and $x^{[L]}$ be the input to and the output of the whole network, respectively. We denote $x_k^{[l]}$ (consequently $b_k^{[l]}$ and $z_k^{[l]}$) to be the kth component of $x^{[l]}$ (consequently $b^{[l]}$ and $z^{[l]}$). Similarly, we denote $W_{ij}^{[l]}$ to be the ijth component of $W^{[l]}$.

(a) Suppose $\sigma(x) = x$, i.e., the identity activation function, is used throughout the neural network. For the sake of simplicity, assume that each $b_k^{[l]}$ is initialized to 0. Furthermore, assume that

- Each $x_k^{[0]} \sim X$ is independent and identically distributed, where X is a distribution with $\mathbb{E}[X] = 0$.
- Each $W_{ij}^{[l]} \sim W_l$ is independent and identically distributed, where W_l is a distribution with $\mathbb{E}[W_l] = 0$.
- All $x_k^{[0]}$ and $W_{ij}^{[l]}$ are mutually independent.

Find an expression for $\text{Var}(x_k^{[1]})$ in terms of $n^{[0]}$, $\text{Var}(W_1)$ and $\text{Var}(X)$.

Hint: you may use the following fact without proof: if Y and Z are independent, zero-mean random variables, then $\text{Var}(YZ) = \text{Var}(Y)\,\text{Var}(Z)$.

Solution: Since $\sigma(\cdot)$ is the identity and $b_k^{[1]} = 0$ for each $k = 1, \ldots, n^{[1]}$, from the forward propagation equations, we know that

$$x_k^{[1]} = \sum_{i=1}^{n^{[0]}} W_{ki}^{[1]} x_i^{[0]}$$

Since each $x_i^{[0]}$ and $W_{ki}^{[1]}$ is IID and mutually independent, it follows that $W_{ki}^{[1]} x_i^{[0]}$ is also IID. Then,

$$\text{Var}(x_k^{[1]}) = \text{Var}\left(\sum_{i=1}^{n^{[0]}} W_{ki}^{[1]} x_i^{[0]}\right) = \sum_{i=1}^{n^{[0]}} \text{Var}(W_{ki}^{[1]} x_i^{[0]})$$

initial here

The last equality follows because each term in the sum above is independent. Letting $Y = W_{ki}^{[1]}$ and $Z = x_i^{[0]}$, we can apply the hint to get

$$\begin{aligned}\text{Var}(x_k^{[1]}) &= \sum_{i=1}^{n^{[0]}} \text{Var}(W_{ki}^{[1]})\,\text{Var}(x_i^{[0]}) \\ &= \sum_{i=1}^{n^{[0]}} \text{Var}(W_1)\,\text{Var}(X) \\ &= n^{[0]}\,\text{Var}(W_1)\,\text{Var}(X)\end{aligned}$$

(b) Show that $\mathbb{E}[x_k^{[1]}] = 0$.

Solution: We go back to our forward propagation equation from the previous part and let

$$\mathbb{E}[x_k^{[1]}] = \mathbb{E}\left[\sum_{i=1}^{n^{[0]}} W_{ki}^{[1]} x_i^{[0]}\right]$$

Again, since each $W_{ki}^{[1]}$ and $x_i^{[0]}$ is mutually independent, it follows that

$$\begin{aligned}\mathbb{E}[x_k^{[1]}] &= \sum_{i=1}^{n^{[0]}} \mathbb{E}[W_{ki}^{[1]}]\,\mathbb{E}[x_i^{[0]}] \\ &= \sum_{i=1}^{n^{[0]}} 0 \cdot 0 \\ &= 0\end{aligned}$$

as desired.

(c) It turns out each $x_k^{[1]}$ (for $1 \le k \le n^{[1]}$) has the same zero-mean distribution. In fact, by induction, one can show that each $x_k^{[l]}$ (for $1 \le k \le n^{[l]}$) has the same zero-mean distribution, which we denote by X_l, and that each $x_k^{[l]}$ is mutually independent of each $W_{ij}^{[l]}$. Then, operating under the same assumptions as part (a), find an expression for $\text{Var}(X_l)$ in terms in terms of $n^{[l-1]}$, $\text{Var}(W_l)$ and $\text{Var}(X_{l-1})$.

Hint: you may use the following fact without proof: if Y and Z are independent, zero-mean random variables, then $\text{Var}(YZ) = \text{Var}(Y)\,\text{Var}(Z)$.

Solution: We follow the exact same approach as part (a). Since the identity activation is still used in layer l, the forward propagation equation tells us that

$$x_k^{[l]} = \sum_{i=1}^{n^{[l-1]}} W_{ki}^{[l]} x_i^{[l-1]}$$

Since each $x_i^{[l-1]}$ and $W_{ki}^{[l]}$ is IID and mutually independent, it follows that $W_{ki}^{[l]} x_i^{[l-1]}$ is also IID. Then,

$$\text{Var}(x_k^{[l]}) = \text{Var}\left(\sum_{i=1}^{n^{[l-1]}} W_{ki}^{[l]} x_i^{[l-1]}\right) = \sum_{i=1}^{n^{[l-1]}} \text{Var}(W_{ki}^{[l]} x_i^{[l-1]})$$

initial here

The last equality follows because each term in the sum above is independent. Letting $Y = W_{ki}^{[l]}$ and $Z = x_i^{[l-1]}$, we can apply the hint to get

$$\begin{aligned}\mathrm{Var}(X_l) = \mathrm{Var}(x_k^{[l]}) &= \sum_{i=1}^{n^{[l-1]}} \mathrm{Var}(W_{ki}^{[l]})\,\mathrm{Var}(x_i^{[l-1]}) \\ &= \sum_{i=1}^{n^{[l-1]}} \mathrm{Var}(W_l)\,\mathrm{Var}(X_{l-1}) \\ &= n^{[l-1]}\,\mathrm{Var}(W_l)\,\mathrm{Var}(X_{l-1})\end{aligned}$$

(d) Show that setting $\mathrm{Var}(W_l) = \frac{1}{n^{[l-1]}}$ will yield $\mathrm{Var}(X_L) = \mathrm{Var}(X)$, i.e., each component of the network input and output will have the same variance. This weight initialization where

$$W_{ij}^{[l]} \sim \mathcal{N}\left(0, \frac{1}{n^{[l-1]}}\right)$$

is called the *Xavier* initialization.

Solution: Unrolling the recursion in the previous part reveals that

$$\mathrm{Var}(X_L) = \mathrm{Var}(X) \prod_{i=1}^{L} n^{[l-1]}\,\mathrm{Var}(W_l)$$

When $\mathrm{Var}(W_l) = \frac{1}{n^{[l-1]}}$,

$$\mathrm{Var}(X_L) = \mathrm{Var}(X) \prod_{i=1}^{L} n^{[l-1]} \frac{1}{n^{[l-1]}} = \mathrm{Var}(X) \prod_{i=1}^{n} 1 = \mathrm{Var}(X)$$

(e) What happens when $\mathrm{Var}(W_l) > \frac{1}{n^{[l-1]}}$ instead? What about $\mathrm{Var}(W_l) < \frac{1}{n^{[l-1]}}$?

Solution: When $\mathrm{Var}(W_l) > \frac{1}{n^{[l-1]}}$, we see that

$$\mathrm{Var}(X_L) = \mathrm{Var}(X) \prod_{i=1}^{L} \underbrace{n^{[l-1]}\,\mathrm{Var}(W_l)}_{>1} \implies \mathrm{Var}(X_L) \gg \mathrm{Var}(X)$$

This indicates a possibly exploding forward pass signal.

On the other hand, when $\mathrm{Var}(W_l) < \frac{1}{n^{[l-1]}}$, we see that

$$\mathrm{Var}(X_L) = \mathrm{Var}(X) \prod_{i=1}^{L} \underbrace{n^{[l-1]}\,\mathrm{Var}(W_l)}_{<1} \implies \mathrm{Var}(X_L) \ll \mathrm{Var}(X)$$

This indicates a possibly vanishing forward pass signal.

Thus, the Xavier initialization avoids the problem of vanishing and exploding forward passes, which also yields better conditioned gradients during back-propagation. This ensures that our training process does not diverge right away at the beginning.

As an aside, the identity activation is a decent approximation for $\sigma(x) = \tanh(x)$ so we expect the Xavier initialization to work well with layers that use the tanh activation function.

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.

initial here

You may use this page to show extra work. Clearly mark your work with the problem number here, and also mention in the problem-specific box that your work is continued here.