CS 189/289A Introduction to Machine Learning
Fall 2024 Jennifer Listgarten, Saeed Saremi

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. We will not grade any work outside of the space provided.
- 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	

○ CS 189

○ CS 289A

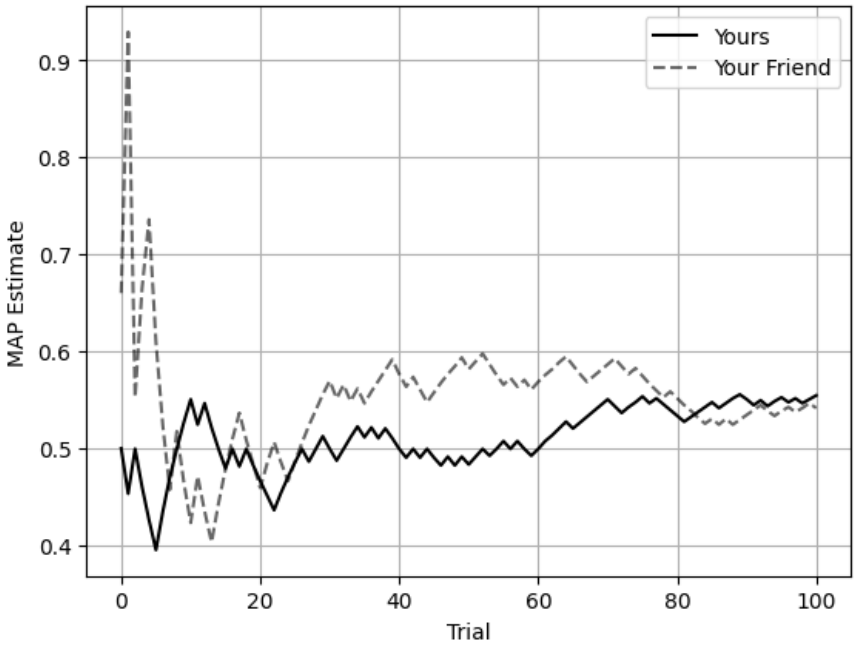
This page intentionally left blank.

initial here

1 Multiple Choice

For the following questions, select the **single best response**.

1. You and your friend are trying to estimate the probability of heads p of a coin. To do this, you both perform a MAP estimate with a binomial likelihood, and you each hand-select a Gaussian prior. You are pretty confident the coin is fair, so you use the prior $p \sim \mathcal{N}(0.5, 0.04^2)$. Your friend does not share your confidence and uses the prior $p \sim \mathcal{N}(\mu, \sigma^2)$. You each flip the coin 100 times and plot your estimate over time starting from $n = 0$ to $n = 100$ trials.



(0.5 points) What is most likely to be true about the mean of your friend's prior μ?

- ◯ $\mu > 0.5$
- ◯ $\mu < 0.5$
- ◯ $\mu = 0.5$

(0.5 points) What is most likely to be true about the variance of your friend's prior σ^2?

- ◯ $\sigma^2 > 0.04^2$
- ◯ $\sigma^2 < 0.04^2$
- ◯ $\sigma^2 = 0.04^2$

(0.5 points) What is most likely to be true about the true probability of heads p?

- ◯ $p = 0.6$
- ◯ $p = 0.55$
- ◯ $p = 0.5$

Solution: The correct answer is A, A, B.

Your friend's initial estimate with no data is higher, so $\mu > 0.5$. Your friend's estimate is much more sensitive to the data in the first few trials, so $\sigma^2 > 0.04^2$. With 100 trials, the contribution of the prior is small, and both you and your friend arrive at roughly at $p = 0.55$, so we can say with high confidence $p = 0.55$.

2. (1.5 points) Suppose we observe the following sequence of values:

$$\mathcal{D} = (2, 2, \pi, 5, \frac{100}{3}, \pi^2, 52, 3, 5, 6)$$

We hypothesize that our data is generated i.i.d. from a continuous uniform distribution $\mathcal{U}[a, b]$. What is the joint MLE of a, b?

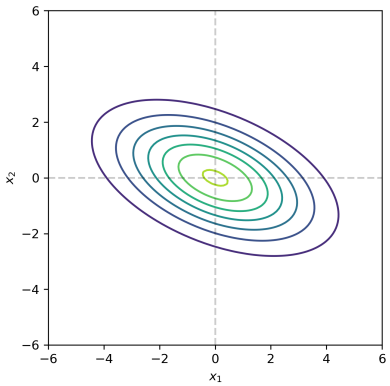
- ◯ $a^* = 2, b^* = 52$.
- ◯ $a^* = b^*$, and a^* is the mean of $\mathcal{D}$.
- ◯ $a^* = -48, b^* = 102$.
- ◯ $a^* = \mu - 2\sigma, b^* = \mu + 2\sigma$, where μ, σ are the mean and variance of $\mathcal{D}$, respectively.

Solution: The correct answer is A.

We compute $\underset{a,b}{\text{argmax}}\ \mathcal{L}(a, b; \mathcal{D}) = \underset{a,b}{\text{argmax}}\ p_{a,b}(\mathcal{D}) = \underset{a,b}{\text{argmax}} \prod_{d \in \mathcal{D}} p_{a,b}(d)$. Note that the value of the uniform PDF is 0 if we observe any value outside the range $[a, b]$, so $a \leq \min \mathcal{D}$ and $b \geq \max \mathcal{D}$. Any point $d \in [a, b]$ has $p(d) = \frac{1}{b-a}$. So we find

$$\begin{aligned}
\underset{a,b}{\text{argmax}}\ \mathcal{L}(a, b; \mathcal{D}) &= \underset{a \leq \min \mathcal{D}, b \geq \max \mathcal{D}}{\text{argmax}} \prod_{d \in \mathcal{D}} p_{a,b}(d) \\
&= \underset{a \leq \min \mathcal{D}, b \geq \max \mathcal{D}}{\text{argmax}} \prod_{d \in \mathcal{D}} \frac{1}{b-a} \\
&= (2, 52)
\end{aligned}$$

3. (1.5 points) Let $X = [x_1, x_2]^T$ be a 2-dimensional Gaussian random variable with mean zero whose probability density function is illustrated by the below contour plot.



Which of the following is most likely to be the correct covariance matrix for X:

- ◯ $\begin{bmatrix} 2 & 1 \\ -1 & 5 \end{bmatrix}$
- ◯ $\begin{bmatrix} 2 & -1 \\ -1 & 5 \end{bmatrix}$
- ◯ $\begin{bmatrix} 5 & -1 \\ -1 & 2 \end{bmatrix}$
- ◯ $\begin{bmatrix} 5 & 1 \\ -1 & 2 \end{bmatrix}$

Solution: The correct answer is C.

x_1 has a greater variance than x_2, indicating that the top-left entry in the covariance matrix must be greater than the bottom-right entry. Moreover, x_1 and x_2 are anti-correlated so the non-diagonal entries must be negative.

4. (1.5 points) Which of the following statements about regularization is **false**?

   - ◯ The LASSO objective is convex.
   - ◯ You can combine different forms of regularization.
   - ◯ It is not possible to add an L2-regularization term to the cost function of a neural network.
   - ◯ Weight sharing in convolutional neural networks can be viewed as regularization.

   Solution: The correct answer is C.

   A is true. The OLS objective is convex and the L1-penalty is also convex. The sum of two convex functions is also convex. B is true. One example is the ElasticNet objective, which combines L1 and L2 regularization. C is false. You can add an L2-penalty to the cost function of a neural network, and differentiate it with respect to the parameters of said network, just like ridge regression. D is true. Weight sharing in CNNs is a form of regularization because we are imposing the prior that local patterns in an image, irrespective of their spatial location, should have the same representation.

5. (1.5 points) Which of the following statements about generative vs discriminative models for classification is **true**?

   - ◯ Logistic regression is a generative approach.
   - ◯ Consider a method where we use MLE to fit Gaussian distributions to features, conditioned on each class, then choose the most likely class. This approach is generative.
   - ◯ Generative models directly model $p(y \mid x)$, where x denotes input features and y denotes output labels.
   - ◯ Suppose our data is linearly separable, and we choose a separating hyperplane by maximizing the distance from the nearest data points of each (binary) class. This is a generative approach.

initial here

Solution: The correct answer is B.

Generative models model $p(x \mid y)$. (This method is known as Gaussian discriminant analysis.) A is false since logistic regression directly models $p(y \mid x)$. C is false since generative models model the joint (or class-conditional) distribution. D is false as we don't model any probability distribution–we geometrically choose a separating hyperplane.

6. (1.5 points) Which of the following statements about stochastic gradient descent (SGD) is **false**?

   ◯ The computation time it takes for SGD to converge is always greater than that of standard gradient descent.

   ◯ SGD can be used to find approximate solutions to non-convex problems.

   ◯ SGD is more memory efficient than standard gradient descent.

   ◯ Due to its stochastic nature, SGD can escape local minima.

   Solution: The correct answer is A.

   A is false. SGD typically requires more updates to converge, but each update is less computationally expensive than an update in standard gradient descent. For this reason, SGD can be overall faster than standard gradient descent. B is true. There is no reason why SGD could not be used for non-convex problems as long as the problem is differentiable. C is true. SGD performs updates using only a single or a small subset of training samples, which typically requires less memory than standard gradient descent. D is true. The noise introduced by the stochastic updates can help the algorithm jump out of local minima.

7. (1.5 points) Which of the following statements about the backpropagation algorithm is **true**?

   ◯ Backpropagation cannot be used to compute the gradients for self-attention layers.

   ◯ Backpropagation requires neural networks to use activation functions that are differentiable everywhere.

   ◯ Backpropagation requires a forward pass to be run through a neural network before the backward pass can be called.

   ◯ The gradients computed during backpropagation can only be used for gradient descent.

   Solution: The correct answer is C.

   The forward pass through the neural network must be run first so any numerical values required during backpropagation can be cached in memory. A is wrong because backpropagation can be used to compute the gradients for virtually every kind of neural network layer used in practice (also note that a self-attention layer is just a different composition of the same basic layers for which we coded backprop for in homework 3). B is wrong because backpropagation works with networks that use ReLU (you can let the derivative at $x = 0$ be either 0 or 1). D is wrong because backpropagation is just an algorithm for computing gradients, and these gradients can be used for any purpose.

8. (1.5 points) Suppose a convolutional layer has the following specifications:

- Input dimensions: [height = 10, width = 10, channels = 5]
- Output dimensions: [height = 10, width = 10, channels = 20]
- Kernel size: [height = 2, width = 2]
- Each kernel also contains an additional bias term.

How many trainable parameters are there in this convolutional layer?

◯ 80
◯ 100
◯ 400
◯ 420

Solution: The correct answer is D.

Each kernel has $(2 \times 2 \times 5 + 1) = 21$ parameters, where the additional one represents the bias term. There are 20 such kernels, so in total there are $21 \times 20 = 420$ parameters.

9. (1.5 points) Which of the following statements about Transformers is **false**?

◯ Transformers use attention to make the loss convex.
◯ Masked attention is only necessary in the decoder.
◯ A constant positional encoding is equivalent to no positional encoding.
◯ None of the above, they are all true.

Solution: The correct answer is A.

A is false, attention does not guarantee convexity. B is true, the goal of masked attention is to prevent the decoder from attending to values it needs to predict in the future. C is true, if the positional encoding is constant across tokens, then no extra information is added. D is incorrect since A is false.

Note: during the exam we clarified the question should be read as "Which of the first three statements about Transformers is false? If None are false, select the last statement."

10. (1.5 points) The method t-SNE is used to visualize high-dimensional data in a lower dimensional space. Which of the following statements regarding t-SNE is **false**?

◯ The method is stochastic.
◯ The method tries to match neighborhood probabilities in the high-dimensional space and the low-dimensional space.
◯ The method uses total variation distance in its loss function, hence the name "total variation SNE".
◯ The method uses a t-distribution to prevent crowding in the low dimensional space.

Solution: The correct answer is C.

t-SNE uses the KL-divergence, not total variation distance, in its loss function. The method stands for "t-distributed SNE" because it switches from the Gaussian used by SNE to a t-distribution in the low dimensional space.

initial here

2 Short Answer

1. (2 points) Let Y be a multivariate Gaussian that can be expressed as AX, where $A \in \mathbb{R}^{3\times 3}$ is a matrix and $X \in \mathbb{R}^3$ is a collection of i.i.d. standard normal RVs. Assume that Y has zero mean. What is Σ_Y, the covariance of Y?

   Solution: We know that $E[Y] = 0$. Since X is a collection of i.i.d. standard normal RVs, $\Sigma_X = I_3$. Therefore, we can rewrite Σ_Y as

   $$\Sigma_Y = \mathbb{E}\left[(AX - 0)(AX - 0)^\top\right] = A\mathbb{E}\left[XX^\top\right]A^\top = A\Sigma_X A^\top$$

   This simplifies to $\Sigma_Y = AIA^\top = AA^\top$.

2. Suppose we have a dataset of independent and identically distributed data points

   $$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

   where $x_i \in \mathbb{R}$ and $y_i \in \{0, 1\}$. Further assume that that data from the two classes fall on different sides of the origin (i.e. $y_i = 0$ when $x_i < 0$ and $y_i = 1$ when $x_i > 0$). We train a logistic regression model (without an intercept term) defined by

   $$p(y_i = 1 \mid x_i) = \frac{1}{1 + e^{-\beta x_i}}.$$

   (a) (1 point) For what values of $\beta \in \mathbb{R}$ will the model obtain a perfect accuracy on the dataset? Recall that for binary-classification, perfect accuracy is achieved if $p(y_i = 1 \mid x_i) > 0.5$ when $y_i = 1$ and $p(y_i = 1 \mid x_i) < 0.5$ when $y_i = 0$.

   (b) (1 point) Say that there is some $\hat{\beta}$ that obtains perfect accuracy. Find a β^* that also obtains perfect accuracy but increases the log-likelihood (or equivalently, lowers the cross-entropy loss) compared to $\hat{\beta}$.

   Solution: Any $\beta > 0$ will achieve perfect accuracy on this dataset, and increasing the β will increase the log-likelihood.

3. (2 points) The SiLU (sigmoid linear unit) function is defined as follows

   $$\text{SiLU}(x) = x \cdot \sigma(x)$$

   where $\sigma(x)$ is the sigmoid function. For this question, we will only consider scalar inputs $x \in \mathbb{R}$. The derivative of the SiLU function can be written as

   $$\frac{d}{dx}\text{SiLU}(x) = b[1 + a \cdot (1 - b)]$$

   What are the missing terms a and b?

   Solution: The derivative of the sigmoid function is $\sigma(x) \cdot (1 - \sigma(x))$.

   For the SiLU function, we again apply the chain rule where $u = x$, $v = u \cdot \sigma(u)$

   $$\begin{aligned}\frac{d}{dx}\text{SiLU}(x) &= 1 \cdot \sigma(x) + x \cdot \sigma(x) \cdot (1 - \sigma(x)) \\ &= \sigma(x)[1 + x \cdot (1 - \sigma(x))]\end{aligned}$$

   Therefore $a = x$ and $b = \sigma(x)$.

initial here

4. (2 points) Suppose we have a convolutional layer composed of a single filter where the bias parameter is zero and the weight parameter is

$$W = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

Consider the following input X containing a single channel

$$X = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 3 & 4 \end{bmatrix}$$

Calculate the (pre-activation) output of the convolutional layer applied on X, with no padding and a stride of 1.

Solution: Since we only have one filter, we know that our output will be a single channel. Let

$$Z_{11} = \langle X[0:2, 0:2], W \rangle_F$$
$$Z_{12} = \langle X[1:3, 0:2], W \rangle_F$$
$$Z_{21} = \langle X[0:2, 1:3], W \rangle_F$$
$$Z_{22} = \langle X[1:3, 1:3], W \rangle_F$$

Then,

$$Z = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 5 \end{bmatrix}$$

5. (2 points) Consider a convolutional layer with the following description:

- Number of filters: 128
- Kernel size: 4x4
- Stride: 2
- Padding size: 1

Calculate the output shape $[n, c, h, w]$ after applying the convolutional layer to an input of shape $[32, 3, 28, 28]$. The shapes are ordered as [batch size, number of channels, height, and width].

Solution: The output height and width is given by

$$\left\lfloor \frac{\text{input} - \text{kernel size} + 2 \cdot \text{padding size}}{\text{stride}} \right\rfloor + 1.$$

Plugging in we get that the output height and width are both 14. The output number of channels is the number of filters, 128, and the output batch size is the same as the input batch size, 32. Our final answer is [32, 128, 14, 14].

initial here

6. (2 points) Suppose we are applying self-attention (with one head) on n tokens. Let $q_1, \ldots, q_n$ be the query vectors, $k_1, \ldots, k_n$ be the key vectors, and $v_1, \ldots, v_n$ be the value vectors.

   Assume that q_i is orthogonal to the key vectors for all tokens (including that for token i). What will the output of self-attention be for token i?

   Solution: The attention weight between token i and any other token j is equal to the following:

$$\begin{aligned}\text{Attention}(q_i, k_j) &= \frac{\exp\{q_i^T k_j\}}{\sum_{\ell=1}^{n} \exp\{q_i^T k_\ell\}} \\ &= \frac{e^0}{\sum_{\ell=1}^{n} e^0} \\ &= \frac{1}{n}\end{aligned}$$

   Therefore, the self-attention output will be $\frac{1}{n}\sum_{i=1}^{n} v_i$.

   Note that the result is the same for scaled dot-product attention with the $\frac{1}{\sqrt{d}}$ scaling term, where d is the dimension of the queries and keys.

   Note: during the exam we clarified that in this question we are using unmasked self-attention.

7. (2 points) Consider the following design matrix containing sample points $X_i \in R^2$.

$$X = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -2 & 0 \end{bmatrix}$$

   Using PCA, what is the first principal component and its explained variance?

   Solution: X is already centered, so $X = \bar{X}$.

   The covariance matrix $\frac{1}{n}\bar{X}^T\bar{X} = \frac{1}{3}\begin{bmatrix} 6 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & \frac{2}{3} \end{bmatrix}$. Since the matrix is diagonal, the eigenvectors and eigenvalues are

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \lambda_1 = 2$$

$$\mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \lambda_2 = \frac{2}{3}$$

   Therefore, the first principal component and its explained variance are $\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \lambda_1 = 2$.

8. (2 points) Describe one advantage of t-SNE over PCA and one advantage of PCA over t-SNE.

   Solution: Advantage of t-SNE:

   - Can capture complex, non-linear patterns in the data

initial here

Advantages of PCA:

- t-SNE has no explicit projection mapping (making it less interpretable)
- SVD provides an efficient way to compute the global minima (t-SNE requires less efficient optimization algorithms that can get stuck in local minima)
- Can embed new points into the low dimensional space without updating the embedding space

initial here

3 A Classifier for Count Data

We consider a binary classification problem in which the observed features are counts. Specifically, for the i-th observation, let $Y_i \in \{0, 1\}$ be the class label, and $X_i \in \mathbb{N}$ be the count features.

We assume that the Y_i are i.i.d. Bernoulli random variables with parameter θ. We also assume that the X_i are i.i.d. random variables with the following distribution

$$\begin{cases} X_i \sim \text{Poisson}(\lambda_1) & \text{if} \quad Y_i = 1 \\ X_i \sim \text{Poisson}(\lambda_0) & \text{if} \quad Y_i = 0. \end{cases}$$

Recall if Z is a Poisson distribution of parameter λ, its probability mass function is given by

$$P(Z = k) = \frac{\exp\{-\lambda\}\lambda^k}{k!}.$$

We hope to learn $(\theta, \lambda_0, \lambda_1)$ using maximum likelihood estimation as follows:

$$(\hat{\theta}, \hat{\lambda}_0, \hat{\lambda}_1) = \underset{\theta, \lambda_0, \lambda_1}{\text{argmax}} \ \mathcal{L}(\theta, \lambda_0, \lambda_1),$$

where $\mathcal{L}(\theta, \lambda_0, \lambda_1) = \sum_{i=1}^n \log P(Y_i, X_i; \theta, \lambda_0, \lambda_1)$ is the log-likelihood of the data. Here, $P(Y_i, X_i; \theta, \lambda_0, \lambda_1)$ is the joint probability of observing Y_i and X_i.

(a) (3 points) Show that log-likelihood $\mathcal{L}(\theta, \lambda_0, \lambda_1)$ can be written, up to a constant, as

$$\sum_{i=1}^n Y_i \left[X_i \log \lambda_1 - \lambda_1 + \log \theta\right] + (1 - Y_i)\left[X_i \log \lambda_0 - \lambda_0 + \log(1 - \theta)\right]$$

Solution: We know that $P(X_i \mid Y_i = k)$ follows a Poisson distribution. For any $k \in \{0, 1\}$

$$P(X_i \mid Y_i = k) = \frac{\exp\{-\lambda_k\}\lambda_k^{X_i}}{X_i!}.$$

Similarly, we also know that $P(Y_i)$ follows a Bernoulli distribution

$$P(Y_i) = \theta^{Y_i}(1 - \theta)^{1 - Y_i}$$

We can then re-write the joint likelihood using these terms

$$\begin{aligned} \mathcal{L}(\theta, \lambda_0, \lambda_1) &= \sum_{i=1}^n \log P(Y_i, X_i) \\ &= \sum_{i=1}^n \log P(Y_i) P(X_i \mid Y_i) \end{aligned}$$

initial here

$$= \sum_{i=1}^{n} \log \theta^{Y_i}(1-\theta)^{1-Y_i} \left(\frac{e^{-\lambda_1}\lambda_1^{X_i}}{X_i!}\right)^{Y_i} \left(\frac{e^{-\lambda_0}\lambda_0^{X_i}}{X_i!}\right)^{1-Y_i}$$
$$= \sum_{i=1}^{n} Y_i \left[X_i \log \lambda_1 - \lambda_1 + \log \theta\right] + (1-Y_i)\left[X_i \log \lambda_0 - \lambda_0 + \log(1-\theta)\right]$$

(b) (3 points) Is $\theta, \lambda_0, \lambda_1 \mapsto \mathcal{L}(\theta, \lambda_0, \lambda_1)$ concave? Justify your answer by computing $\nabla^2 \mathcal{L}(\theta, \lambda_0, \lambda_1)$. Assume the Y_i are not all the same value.

Solution: We define $f_i(\theta, \lambda_0, \lambda_1)$ to be our expression from (b). For any i, let $f_i(\theta, \lambda_0, \lambda_1) := Y_i\left[X_i \log \lambda_1 - \lambda_1\right] + (1-Y_i)\left[X_i \log \lambda_0 - \lambda_0\right] + Y_i \log \theta + (1-Y_i)\log(1-\theta) - C$ Consequently, the Hessian can be expressed as follows

$$\nabla^2 \mathcal{L}(\theta, \lambda_0, \lambda_1) = \sum_{i=1}^{n} \nabla^2 f_i(\theta, \lambda_0, \lambda_1)$$
$$= \begin{pmatrix} -\sum_{i=1}^n Y_i/\theta^2 + (1-Y_i)/(1-\theta)^2 & 0 & 0 \\ 0 & -\sum_{i=1}^n (1-Y_i)X_i/\lambda_0^2 & 0 \\ 0 & 0 & -\sum_{i=1}^n Y_i X_i/\lambda_1^2 \end{pmatrix}$$

Yes, the function is concave. To test for concavity, we need to show that the Hessian is negative semi-definite. Since the Hessian is a diagonal matrix, its eigenvalues are the elements on the diagonal. Since $X_i \geq 0$ and $Y_i \in \{0, 1\}$, from our expression for the Hessian we know all the eigenvalues are less than or equal to zero. Consequently, the maxima of the log-likelihood are global and can be found using first-order conditions.

(c) (3 points) Compute the maximum likelihood estimates $\hat{\theta}, \hat{\lambda}_0, \hat{\lambda}_1$.

Solution: Taking the gradient with respect to θ gives the following condition

$$\sum_{i=1}^{n} \frac{Y_i}{\theta} - \frac{1-Y_i}{1-\theta} = 0.$$

We multiply both terms by $\theta(1-\theta)$ and rearrange the terms to get

$$\hat{\theta} = \frac{\sum_i Y_i}{n}$$

Using the same strategy, we obtain

$$\hat{\lambda}_1 = \frac{\sum_i Y_i X_i}{\sum_i Y_i} \quad \text{and} \quad \hat{\lambda}_0 = \frac{\sum_i (1-Y_i) X_i}{\sum_i (1-Y_i)}.$$

(d) (1 point) Is this model generative or discriminative? Why?

Solution: The model is generative, since it models the joint distribution of features and labels, and not labels conditioned on features only like discriminative models.

initial here

4 Matrix Decomposition on Ridge Regression

Consider the following common set-up: we have a dataset $X \in \mathbb{R}^{n \times d}$ that contains n data points and d features per data point, and a target dataset $Y \in \mathbb{R}^n$ of target outputs. We parameterize a linear regression model $f(x) = w^T x$, where w is a learned weight vector and x is a data point.

To train our model we use ridge regression, where we solve the following optimization problem for some fixed value of λ

$$w_{\text{opt}} = \underset{w}{\text{argmin}} \ ||Xw - Y||^2 + \lambda ||w||^2$$

The optimal solution to this problem is

$$w_{\text{opt}} = (X^T X + \lambda I)^{-1} X^T Y$$

(a) (1 point) As λ approaches ∞, what does w_{opt} approach? Provide a brief justification.

Solution: The solution w_{opt} approaches 0 as λ approaches ∞. This is because the second term $\lambda ||w||_2^2$ dominates the expression, i.e., for any non-zero value of w, there exists a value λ such that $\lambda ||w||_2^2 > ||Xw - Y||_2^2$.

(b) (2 points) Suppose we know the SVD of X, where $X = U \Sigma V^T$. Recall that $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{d \times d}$ are orthonormal matrices, and $\Sigma \in \mathbb{R}^{n \times d}$ is defined as $\text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_d)$.

Consider the matrix $(X^T X + \lambda I)$ that is computed to produce w_{opt}. Write the spectral decomposition of $(X^T X + \lambda I)$.

Solution: Recall that for a symmetric matrix A, its spectral decomposition is written in the form QDQ^T. We can write $(X^T X + \lambda I)$ in terms of the components from the SVD of X, which we can see simplifies to its spectral decomposition

$$X^T X + \lambda I = (V \Sigma^T \Sigma V^T + V \lambda V^T I) = V \Sigma' V^T$$

where $\Sigma' = \text{diag}(\sigma_1^2 + \lambda, \sigma_2^2 + \lambda, \ldots, \sigma_d^2 + \lambda)$.

(c) (3 points) When $\lambda \gg \sigma_1^2$, what matrix does $(X^T X + \lambda I)^{-1}$ approach? You may find your answer from part (b) useful.

Solution:

From part (b), we know that $\Sigma' = \text{diag}(\sigma_1^2 + \lambda, \sigma_2^2 + \lambda, \ldots, \sigma_d^2 + \lambda)$.
In the case that $\lambda \gg \sigma_1^2 \geq \sigma_i^2$, its inverse Σ'^{-1} approaches $\text{diag}(\frac{1}{\lambda}, \frac{1}{\lambda}, \ldots, \frac{1}{\lambda}) = \frac{1}{\lambda} I$.
Therefore, the entire expression $(X^T X + \lambda I)^{-1}$ approaches

$$(X^T X + \lambda I)^{-1} = (V \Sigma' V^T)^{-1} = V \Sigma'^{-1} V^T \approx V(\frac{1}{\lambda} I) V^T = \frac{1}{\lambda} V V^T = \frac{1}{\lambda} I$$

where we used the fact that $V^{-1} = V^T$ and $VV^T = I$ since V is an orthonormal matrix.

(d) (3 points) Now, we want to consider how the direction of w_{opt} changes as we vary λ. To do this, we define $w_{\text{norm}} = \frac{w_{\text{opt}}}{||w_{\text{opt}}||}$. Write an expression for w_{norm} in terms of X, Y, λ as λ approaches ∞. You may find your answer from part (c) useful.

initial here

Solution: As λ approaches ∞, $\lambda \gg \sigma_1^2$. Using part (c), in this case

$$w_{\text{opt}} = (X^T X + \lambda I)^{-1} X^T Y \approx \frac{1}{\lambda} X^T Y$$

Therefore,

$$w_{\text{norm}} \approx \frac{\frac{1}{\lambda} X^T Y}{\|\frac{1}{\lambda} X^T Y\|_2} = \frac{X^T Y}{\|X^T Y\|_2}.$$

The interpretation here is that, as λ increases, ridge regression increasingly assumes that $(X^T X + \lambda I)^{-1} \approx \frac{1}{\lambda} I$, or that the covariance between different features is zero, making the direction of the solution w_{norm} depend more on the covariance between the features and labels instead.

initial here

5 Residuals vs Errors in Linear Regression

In linear regression, we hold the following hypotheses:

i. $Y = Xw + e$

ii. $e \sim \mathcal{N}(0, \sigma^2 I)$

where $Y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times d}$, $w \in \mathbb{R}^d$, and $e \in \mathbb{R}^n$. We call e the vector of "errors."

The errors are a theoretical quantity that we don't know as they depend on knowledge of the true (often unobserved) value. What we can do is look at the residuals defined by $\epsilon = Y - \hat{Y}$, the difference between our observed Y and our predicted $\hat{Y}$. In this problem, you will show that the residuals do not share the same distribution as the errors.

(a) (3 points) From the above assumptions, we can derive that $Y \mid X \sim \mathcal{N}(Xw, \sigma^2 I)$. You may use this fact without proof.

Show that the MLE of w (keeping σ fixed) is the same as the minimizer of the sum of squared residuals. Precisely, demonstrate the following:

$$\operatorname*{argmax}_w \mathcal{L}(w; Y \mid X) = \operatorname*{argmin}_w \|Y - Xw\|_2^2.$$

Solution:

$$\begin{aligned}
\operatorname*{argmax}_w \mathcal{L}(w; Y \mid X) &= \operatorname*{argmax}_w \prod_{i=1}^n p_w(Y_i \mid X_i) \\
&= \operatorname*{argmax}_w \sum_{i=1}^n \log p_w(Y_i \mid X_i) \\
&= \operatorname*{argmax}_w \sum_{i=1}^n \left(-\frac{(Y_i - (Xw)_i)^2}{2\sigma^2} - \log\left(\sigma\sqrt{2\pi}\right) \right) \\
&= \operatorname*{argmin}_w \sum_{i=1}^n (Y_i - (Xw)_i)^2 \\
&= \operatorname*{argmin}_w \|Y - Xw\|_2^2
\end{aligned}$$

(b) (3 points) Suppose we minimize the above loss function (sum of squared residuals) and attain our optimal weight vector w^*. Prove that, when our data matrix contains a **bias term** (that is, one column of X is the vector with every entry being 1), then the sum of residuals is 0. Precisely, prove that

$$\sum_{i=1}^n \epsilon_i = \sum_{i=1}^n (Y_i - \hat{Y}_i) = 0,$$

initial here

where ϵ is our vector of residuals defined as $Y - \hat{Y}$, and $\hat{Y} = Xw^*$. You do **not** need to prove convexity of $\|Y - Xw\|_2^2$.

Hint: It may be helpful to rewrite $Y_i - \hat{Y}_i$ as $Y_i - (X'w')_i - w_0$, where X' is the data matrix with the bias column removed, w' is the weight vector with the bias weight removed, and w_0 is the bias weight.

Solution: Since the sum of squared residuals is differentiable, convex, and we are optimizing over an open set, we know that the first-order conditions must hold at the optimum w^*, so $\nabla_w \|Y - Xw\|_2^2|_{w=w^*} = 0$. Hence, $\frac{\partial}{\partial w_0}\|Y - Xw\|_2^2|_{w=w^*} = 0$. So we have

$$\begin{aligned}
\frac{\partial}{\partial w_0}\|Y - Xw\|_2^2|_{w=w^*} &= \frac{\partial}{\partial w_0}\sum_{i=0}^{n}(Y_i - (X'w')_i - w_0)^2 \\
&= \sum_{i=0}^{n} -2(Y_i - (X'w')_i - w_0) = 0 \\
&= \sum_{i=0}^{n}(Y_i - (Xw)_i) = 0 \\
&\implies \sum_{i=0}^{n}(Y_i - \hat{Y}_i) = \sum_{i=0}^{n}\epsilon_i = 0
\end{aligned}$$

An alternate solution using orthogonality is also valid. Since OLS gives us a projection onto the range of X, the residuals must be orthogonal to each column of X. This includes the column of all ones. Hence, $\mathbf{1} \cdot \epsilon = 0 \implies \sum_i \epsilon_i = 0$.

(c) (3 points) Using the result of the previous part, prove that the residuals ϵ do not share the same distribution as the errors e. Precisely, prove that

$$\epsilon \nsim \mathcal{N}(0, \sigma^2 I).$$

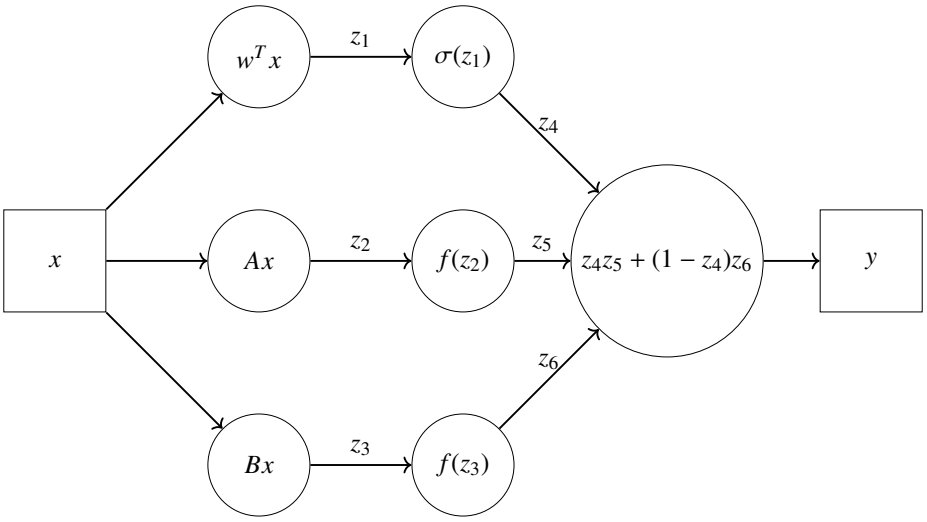
Hint: A multivariate Gaussian with a diagonal covariance matrix has (mutually) independent components. You may use this fact without proof.

Solution: Using the hint, we note that our assumption indicates that the components of ϵ should be independent, but, in the previous part, we showed that the sum of residuals (components of epsilon) sum to 0. This indicates that the value of ϵ_i is completely determined by the values of all ϵ_j, $j \neq i$ since $\epsilon_i = -\sum_{j\neq i}\epsilon_j$. So, under the assumption of independence we should have $p(\epsilon_i \mid (\epsilon_j)_{i\neq j}) = p(\epsilon_i)$, which is our normal PDF centered at 0 with variance σ^2, but from the result in part (b), we find $\epsilon_i \mid (\epsilon_j)_{i\neq j}$ is completely determined with mean $-\sum_{j\neq i}\epsilon_j$ and zero variance.

initial here

6 Computational Graph Analysis

You are building a small neural network which takes in a vector $x \in \mathbb{R}^d$ and outputs another vector $y \in \mathbb{R}^d$. The computational graph of your network looks like this



where $w \in \mathbb{R}^d$, $A, B \in \mathbb{R}^{d \times d}$ are the parameters of your layers, σ is the sigmoid function, and $f : \mathbb{R}^d \to \mathbb{R}^d$ is some activation function.

(a) Suppose $f(x) = x$ is the identity function. Compute the following derivatives in terms of the variables defined in the computational graph.

i. (1.5 points) $\partial y / \partial z_1$

Solution:

$$\frac{\partial y}{\partial z_1} = \frac{\partial y}{\partial z_4}\frac{\partial z_4}{\partial z_1} = (z_5 - z_6)\sigma(z_1)(1 - \sigma(z_1))$$
$$= \sigma(z_1)(1 - \sigma(z_1))(z_5 - z_6).$$

ii. (1 point) $\partial y / \partial z_2$

Solution:

$$\frac{\partial y}{\partial z_2} = \frac{\partial y}{\partial z_5}\frac{\partial z_5}{\partial z_2} = z_4 I$$
$$= z_4 I.$$

iii. (1 point) $\partial y / \partial A_{*j}$, where A_{*j} is the j-th column of A

Solution:

$$z_2 = Ax = \sum_{k=1}^{d} A_{*k} x_k$$

initial here

$$\frac{\partial z_2}{\partial A_{*j}} = Ix_i = x_i I$$
$$\frac{\partial y}{\partial A_{*j}} = \frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial A_{*j}} = z_4 x_i I.$$

iv. (1.5 points) $\partial y/\partial x$

Solution:

$$\begin{aligned}\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial z_1}\frac{\partial z_1}{\partial x} + \frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial x} + \frac{\partial y}{\partial z_3}\frac{\partial z_3}{\partial x} \\ &= \sigma(z_1)(1-\sigma(z_1))(z_5 - z_6)w^T + z_4 A - z_4 B.\end{aligned}$$

(b) (2 points) Suppose $f(x) = \text{ReLU}(x)$. Briefly identify when and how each term of $\partial y/\partial x$ would change from your answer in part (a).

Solution:

When $z_2 = (Ax)_i < 0$, i-th row of the $z_4 A$ term will be zeroed out. This is because previously for the identity activation $(\partial z_5/\partial z_2) = I$, but for ReLU $(\partial z_5/\partial z_2) = \text{diag}(\mathbf{1}[z_2 > 0])$, where $\mathbf{1}[z_2 > 0]$ is an indicator vector containing 1 where $z_2 > 0$ and 0 elsewhere.

The same logic holds when $z_3 = (Bx)_i < 0$, and the i-th row of the $-z_4 B$ term will be zeroed out.

Solutions which describe how the derivative of ReLU becomes 0 when the input is less than 0 will receive partial credit. Solutions must identify the terms which change to receive full credit.

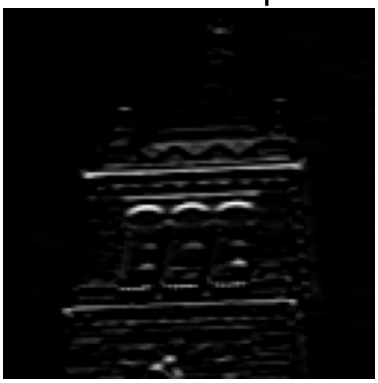
initial here

7 Reverse-Engineering CNN Filters

You have an image and the outputs from two different convolutional filters, and you want to reverse-engineer what filters were applied. You know that the filters are 3x3 with no bias term, applied with no padding and a stride of 1, followed by a ReLU activation. The image and the two outputs are normalized such that all values are $\in [0, 1]$, where 0 is black and 1 is white.

Input Image

Filter 1 Output



Filter 2 Output

(a) (3 points) Fill in weights for each filter. All weights are a value in the set $\{-1, 0, 1\}$. Some values have already been filled in for you.

Filter 1 $\begin{bmatrix} 1 & \_\_ & \_\_ \\ \_\_ & 0 & \_\_ \\ \_\_ & \_\_ & -1 \end{bmatrix}$ Filter 2 $\begin{bmatrix} 1 & \_\_ & \_\_ \\ \_\_ & 0 & \_\_ \\ \_\_ & \_\_ & -1 \end{bmatrix}$

Solution: Filter 1 is $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$ and Filter 2 is $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$.

(b) (1 point) Qualitatively, what types of features are being identified by each filter?

Solution: Filter 1 detects horizontal edges, and Filter 2 detects vertical edges.

initial here

8 Gradient Descent for Linear Regression (CS289A Only)

Only complete this problem if you are enrolled in CS289A.
Do **not** complete this problem if you are enrolled in CS189.

Suppose we have a dataset of n samples $D = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We can stack the data into an $n \times d$ design matrix X and the labels into a vector $y \in \mathbb{R}^n$. Recall that the objective of the ordinary least squares problem is to find a weight vector $w \in \mathbb{R}^d$ that minimizes the squared error $J(w) = \|Xw - y\|_2^2$. You may use without proof that $J(w)$ is convex.

(a) (2 points) Show that any $w \in \mathbb{R}^d$ that satisfies the *normal equation* $X^\top Xw = X^\top y$ will be a solution to the OLS problem. When does a unique solution exist, and what is that solution?

Solution: Note that $J(w) = w^\top X^\top Xw - 2w^\top X^\top y + y^\top y$. Since $J(w)$ is convex, we minimize it by setting its gradient to 0:

$$\nabla_w J(w) = 0 \implies 2X^\top Xw - 2X^\top y = 0 \implies X^\top Xw = X^\top y$$

When X is full-rank, that is $X^\top X$ is full rank, it follows that $X^\top X$ will be invertible. Then, the normal equation will have a unique solution given by $w = (X^\top X)^{-1} X^\top y$.

(b) (2 points) In Big-O notation, what's the computational cost of finding a solution to the normal equation $X^\top Xw = X^\top y$? We will measure the cost in FLOPs (floating point operations), i.e., the total number of additions, divisions, subtractions and multiplications between two scalar numbers required to perform a computation.
Hint: Solving a linear system, represented by an $m \times m$ matrix, using Gaussian elimination takes $O(m^3)$ FLOPs.

Solution: The computational complexity of finding this solution can be broken down into several parts:

- Computing $X^\top X$ will take $O(nd^2)$ FLOPs.
- Computing $X^\top y$ will take $O(nd)$ FLOPs.
- Performing Gaussian elimination to solve $X^\top Xw = X^\top y$ will take $O(d^3)$ FLOPs.

Thus, the overall cost is $O(d^3 + nd^2)$ FLOPs.

(c) (1 point) We learned about an iterative algorithm called gradient descent in lecture for training logistic regression models and neural networks. However, gradient descent is a very general algorithm that can be applied to many optimization problems, including least squares. Write down the gradient descent update for w. Denote the learning rate by η.

Solution: We already have the gradient from the first part. Thus, the gradient descent update will be given by

$$w \leftarrow w - 2\eta X^\top (Xw - y)$$

initial here

(d) (2 points) In Big-O notation, what is the computational cost of performing gradient descent for L iterations? Once again, we will measure the cost in FLOPs. Different sequences of computations will yield different costs. Pick the lowest cost.

Solution: It takes $O(nd)$ FLOPs to compute $Xw - y$, and $O(nd)$ further FLOPs to compute $2\eta X^\top (Xw - y)$. Once the gradient is computed, the actual parameter update only takes $O(d)$ FLOPs. Thus, the computational cost of a single gradient descent update is $O(nd)$ FLOPs. The cost of running gradient descent for L iterations is, then, $O(Lnd)$ FLOPs.

(e) (1 point) For an appropriately chosen learning rate, gradient descent will converge as $L \to \infty$. However, we can only run a finite number of gradient descent iterations on a real computer so this solution ends up being an approximate minimizer of the OLS objective, unlike the solution returned by the normal equation, which will be an exact minimizer.

Regardless, it is still used as one of the main methods for solving OLS in practice. In what situations might the gradient descent approach be preferable over attempting to solve the normal equation directly?

Solution: Suppose $d \gg n$, i.e., the number of features far exceeds the number of samples. In such cases, the $O(d^3)$ complexity of attempting to solve the normal equation will dominate everything else and using gradient descent to get an approximate solution will be far more efficient (and sometimes, the only way to get *any* kind of solution).

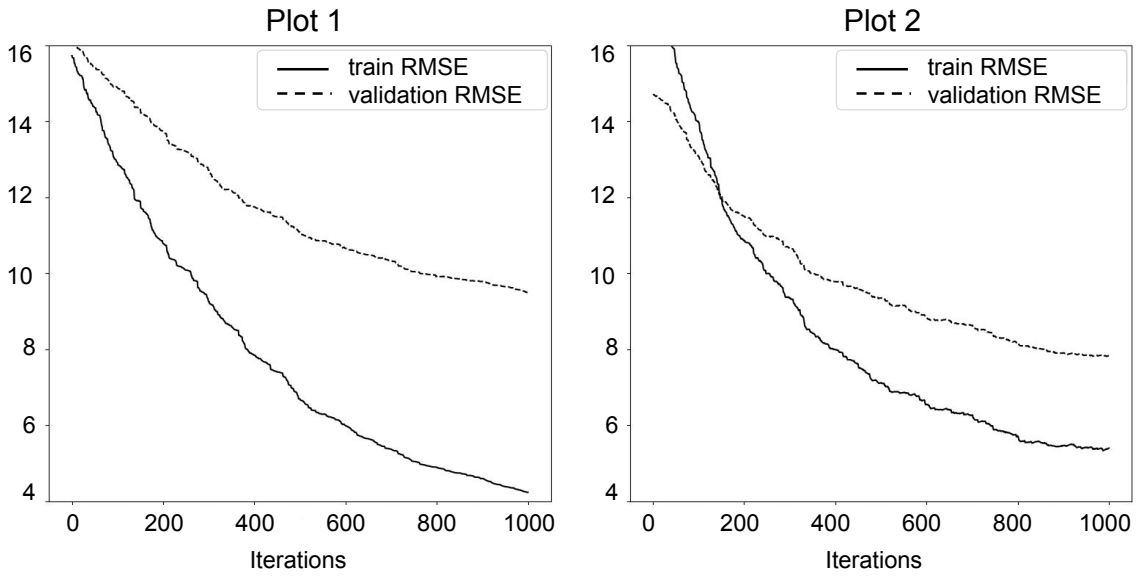
(f) (2 points) Let $J_\lambda(w) = \|Xw - y\|_2^2 + \lambda \|w\|_2^2$, for $\lambda > 0$, be the objective function for ordinary least squares with an l2-penalty, i.e., ridge regression.

Suppose we split the dataset into a training and validation split, and run stochastic gradient descent (SGD), using only the training set, for 1000 iterations to optimize $J(w)$ versus $J_\lambda(w)$, for some $\lambda > 0$. We also plot the *root mean squared error* (RMSE) of the parameter vector $w^{(t)}$ at iteration t for each $t = 0, \ldots, 1000$, on the entire training and validation splits, for each experiment. The root mean squared error is defined as

$$\text{RMSE}(X, y, w) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (x_i^\top w - y_i)^2}$$

for $X \in \mathbb{R}^{m \times d}$ and $y \in \mathbb{R}^m$, where (X, y) is either the train or validation split with m samples.

initial here



We have plotted the training and validation RMSE curves from running SGD to minimize $J(w)$ and $J_\lambda(w)$, but we don't know which plot corresponds to which objective! Based on just the figure above, identify this correspondence and briefly justify your reasoning (1-2 sentences is enough).

Solution: Plot 1 corresponds to $J(w)$ and plot 2 corresponds to $J_\lambda(w)$.

Note that the l2-penalty term in the ridge regression objective has a regularizing effect on the model, and we trade off a lower training RMSE for a lower validation RMSE, i.e., better generalization. This can also be seen when you compare the gaps between the training and validation curves in each plot: the gap in the second plot is smaller, which indicates that the l2-regularized model generalizes better to unseen data.

We would intuitively also expect the training RMSE for $J(w)$ to be lower than that of $J_\lambda(w)$ since $[\text{RMSE}(X, y, w)]^2$ is just $J(w)$ scaled down by a scalar factor. In other words, SGD is directly optimizing the RMSE in the case of $J(w)$, but its minimizing a slightly different objective in the case of $J_\lambda(w)$.

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.