

CSE 446/546 Autumn 2025 Final Exam

December 8, 2025

Name _____

UW NetID (not the numbers) _____

Please **wait** to open the exam until you are instructed to begin, and please take out your Husky Card and have it accessible when you turn in your exam.

Instructions: This exam consists of a set of short questions (True/False, multiple choice, short answer).

- NOTE: For multiple choice questions, clearly indicate your answer by filling in the letter(s) associated with your choice. Bubble in your answers. Do not write your answer to the side.
Example:

Not selected answer: ☐ a Selected answer: ☒ a

- Multiple choice questions marked with should only be marked with one answer. All other multiple choice questions are , in which case any number of answers may be selected (**including none, one, or more**).
- For questions, you will receive proportional credit for each option based on whether you get each “option” correct/incorrect. For example, if the question is worth 1 point and there are 4 options, you will receive 0.25 points for each option that matches the solution.
- For each short answer question, please write your answer in the provided space.
- If you need to change an answer or run out of space, please very clearly indicate what your final answer is and what you would like graded. Responses where we cannot determine the selected option will be marked as incorrect.
- Please write your NetID at the top of every page.
- Please remain in your seats for the last 5 minutes of the exam. If you complete the exam before the last 5 minutes, you may turn in your exam to a TA.

ID: _____

Page 2

This image is included only to cover the back of this page. It has no relation to the exam.



1. One Answer 1 point

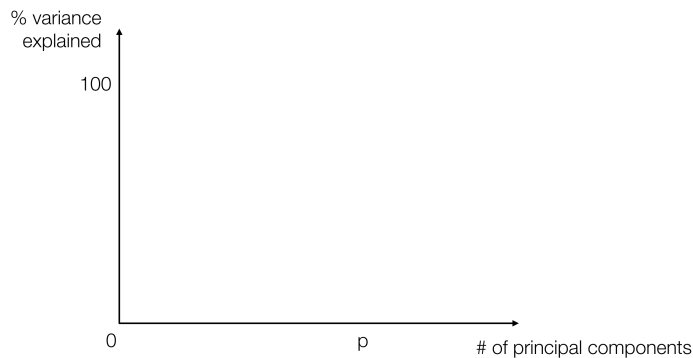
In k-means clustering, we optimize the training objective through alternating minimization. Specifically, we repeatedly alternate between these two steps: first, assign all datapoints in the set to the nearest “centroid”, and then second, update that centroid to be the arithmetic mean of the points assigned to it. What will happen if we update the centroid to be the coordinate-wise **median** of all points assigned to it instead of the **mean**?

- (a) This new clustering algorithm can now draw nonlinear decision boundaries to separate clusters.
- (b) **This new clustering algorithm will be less sensitive to outliers in the dataset.**
- (c) This new clustering algorithm will produce centroids that are always equivalent to *some* datapoint in the dataset.
- (d) There will be no functional change to the clustering algorithm, i.e., it will return the same clusters as k-means given the same initialization.

Solution: A is wrong because changing means to medians does not give this clustering algorithm the power to draw nonlinear boundaries. C is wrong because medians don't have to strictly be points in the dataset (e.g. $\text{median}(\{1,2,3,4\}) = 2.5$). D is wrong since B is correct: the median of a dataset is less sensitive to outliers than the mean and this will be reflected in the cluster centroids update.

2. 1 point

You are performing PCA on an experimentally collected dataset $x_1, x_2, \dots, x_n$ with each $x_i \in \mathbb{R}^d$ to understand its structure. You plot the # of principal components on the x-axis and the % cumulative explained variance on the y-axis. You also know from domain knowledge that there is a low-dimensional subspace of dimension p that the data generally lies on, but not strictly. Sketch out the shape that you expect the plot to take.



Solution: Since the data generally lies on a low-dimensional subspace of dimension p , the first p principal components will capture the vast majority of the variance (the signal). Therefore, the plot should show a steep, rapid increase in cumulative explained variance for the first p components; this should be concave because the first PC explains more variance than the second PC etc. Because the data lies on this subspace "not strictly", the curve will not hit 100% exactly at p . Instead, there will be a distinct "elbow" or "knee" at $x = p$. After this point, the curve will flatten out significantly, growing much more slowly towards 100% as the remaining components ($p + 1$ to d) capture only the residual noise.

3. 2 points Answer the following questions about underfitting and overfitting.

- a. List one way to detect overfitting and one way to detect underfitting in an ML model.

Overfitting: _____

Underfitting: _____

- b. Describe one way to fix overfitting and one way to fix underfitting.

Overfitting: _____

Underfitting: _____

Solution: It's ok if the answers are short as long as they make sense.

An example:

Detecting overfitting: The train error is way lower than the test error

Detecting underfitting: The train and test errors don't improve much at all over training

Fixing overfitting: Increase regularization

Fixing underfitting: Increase model complexity

4. Select All That Apply 1 point

Consider the kernelized k-NN algorithm using the RBF kernel

$$K(x, x') = \exp(-\gamma \|x - x'\|_2^2),$$

where $\gamma > 0$. When predicting the label or value for a new point x_{new} , the similarity $K(x_{\text{new}}, x_i)$ determines how much each training point x_i influences the prediction. Suppose there are n training points $x_1, x_2, \dots, x_n$ and that the data is not normalized/standardized. Which of these statements are true?

- (a) If we choose to scale some features (i.e., dimensions of x), the absolute value of the similarities $K(x_{\text{new}}, x_i)$ will change, but the relative ranking of $K(x_{\text{new}}, x_1), K(x_{\text{new}}, x_2), \dots, K(x_{\text{new}}, x_n)$ will not.
- (b) If one feature represents "Annual Income" (range 20k-200k) and another represents "Age" (range 0-100), the "Annual Income" feature will likely have a disproportionately large effect on the similarity K , because of its larger range.
- (c) If $\gamma \|x_{\text{new}} - x_i\|_2^2$ is large (i.e., highly positive), then the training point x_i has little influence on the prediction for x_{new} .
- (d) A larger value of γ will make only closer training points influence the prediction, producing more local behavior, whereas a smaller value of γ spreads influence across more points, resulting in smoother predictions.

Solution:

- (a) False. The RBF kernel relies on the Euclidean distance $\|x - x'\|_2$. Scaling specific features changes the geometry of the space and can therefore change the relative distances of the training points from x_{new} .
- (b) True. The distances in the annual income feature will likely be much larger and therefore dominate the calculation.
- (c) True. This follows directly from the formula for K .
- (d) True. The parameter γ (gamma) controls the width of the Gaussian kernel (inverse to variance σ^2). When γ is large, $-\gamma \|\dots\|$ becomes very negative very quickly as you move away from the point, so only points extremely close to x_{new} have any influence. Conversely, when γ is small, the influence drops off slowly as distance increases.

5. 1 point

What is one benefit of using Leave-One-Out Cross-Validation over K-Fold Cross-Validation? What about vice versa?

Solution: LOOCV Pros: has the benefit of a relatively low-bias estimation of performance because only one data point is left out. Good for small datasets LOOCV Con: Highly computationally expensive (have to run through a lot of data points) K-Fold Pro: Faster computation, actually is somewhat of a more pessimistic estimator (but this is less useful for hyperparameter tuning) K-Fold Con: higher bias (model trained on smaller subset of data than LOOCV), doesn't work well if dataset is very small.

6. Select All That Apply 1 point

Consider the standard data matrix $X \in \mathbb{R}^{n \times d}$, where the i -th row is an input data point $x_i^T \in \mathbb{R}^d$. Further assume that the data is *centered* such that $\sum_{i=1}^n x_i = X^T \mathbf{1}_n = 0$. Given the singular value decomposition of the centered data matrix

$$X = U S V^T,$$

which of the following statements are correct?

(a) The columns of V are the eigenvectors of the covariance matrix

$$C = \frac{1}{n} X^T X.$$

(b) The singular values in S are equal to the square roots of the eigenvalues of the covariance matrix C (up to scaling by n).

(c) Assume we want to perform PCA to reduce the data to k dimensions ($k < d$). Then the top k right singular vectors (columns of V) are the principal directions in the feature space; projecting each data point $x_i \in \mathbb{R}^d$ onto these vectors yields the standard k -dimensional PCA representation.

(d) Multiplying X by U gives the principal component *projections*.

Solution: Right singular vectors are the same as the principal component vectors (discussed in section 9) making (a) correct. (b) is also correct as presented in lecture and section. (c) is correct. (d) is wrong since U is not related to PCA in any way.

7. One Answer 1 point

Imagine that you standardize the features in your dataset using the mean and variance computed over the entire dataset *before* you split the dataset into training, validation, and test splits. You then proceed to train on the training set and do hyperparameter tuning on the validation set before evaluating the model's performance on the test data.

Which outcome would you expect?

- (a) The test accuracy will be an unbiased estimate of the model's true generalization performance because standardization affects all samples equally.
- (b) The test accuracy will be an overestimate of the model's true generalization performance because information from the test set leaked during preprocessing.**
- (c) The test accuracy will be lower than the model's true generalization performance because using the statistics of the entire dataset for standardization compromises accuracy.

Solution: The answer is b because the test set was used to determine the statistics that standardized the features, which leaked information from the test set and can artificially boost generalization performance.

8. 2 points

Answer the following questions about K-nearest neighbor classification models. Assume there are n data points in the training set, which are each in $\mathbb{R}^d$, and that the training and test data are independently and identically distributed.

- a. What is the time complexity for classifying a new data point $x_{new} \in \mathbb{R}^d$ using a naive brute-force implementation? Assume that it takes $O(n \log n)$ time to sort n numbers and pick out the top k . Give your answer in terms of n and d , ignoring terms in k .

$O(\quad)$

b. Select All That Apply

Which of the following is true?

- Ⓐ For any fixed dimensionality d , as the number of training points n grows to infinity the test error of a K -nearest-neighbor model will be at most twice of the Bayes error.
- Ⓑ Normalizing features is important for k -Nearest Neighbors, because otherwise features with larger ranges will tend to have more influence.
- Ⓒ k -Nearest Neighbors is parametric because k is a parameter.
- Ⓓ Smaller k tends to lead to underfitting.

Solution: a. The time complexity is $O(n \cdot d + n \log n)$. In a naive approach, we must calculate the distance between x_{new} and all n training points. Each distance calculation (e.g., Euclidean) takes $O(d)$ time. Finding the top k neighbors after computing distances takes $O(n)$ (using quickselect) or $O(n \log n)$ (sorting), though the distance computation $O(nd)$ is often considered the dominant term for high-dimensional data.

b. We covered the Bayes error statement in class.

Normalization is crucial because distance metrics (like Euclidean distance) are sensitive to the scale of features; a feature with a large range (e.g., salary) would otherwise dominate the distance calculation over a feature with a small range (e.g., age).

Regarding the distractors:

- k -NN is *non-parametric* because the "parameters" (the stored training data) grow with the sample size n , rather than being fixed. k is a hyperparameter, not a learned parameter.
- Smaller k leads to a more complex, jagged decision boundary that captures noise, which leads to **overfitting**, not underfitting. Larger k smoothes the boundary, leading to lower variance but higher bias (underfitting).

9. Select All That Apply 1 point

Which of the following are true about autoencoders?

- Ⓐ **An autoencoder is trained without labeled data.**
- Ⓑ Autoencoders are primarily used for regression tasks.
- Ⓒ **The input dimension of the encoder and the output dimension of the decoder must be the same.**
- Ⓓ **The primary objective is to learn a compressed representation of the input data.**

Solution:

1. They are unsupervised (or self-supervised) models because they do not require external labels

2. Autoencoders are trained by learning to minimize the reconstruction loss, *not* for regression.
3. Since the goal is to reproduce the input x , the output layer $\hat{x}$ must have the exact same dimensionality as the input layer.
4. The hidden layer (bottleneck) is typically smaller than the input dimension. This constraint forces the network to learn a compressed, efficient representation (encoding) of the data to successfully reconstruct it.

10. One Answer 1 point

You have a dataset of 10,000 labeled examples and want to (1) tune hyperparameters and (2) obtain an approximately unbiased estimate of the generalization performance of your final model. Which of the following strategies is most appropriate?

- (a) **Randomly split the data into 80% training and 20% test. Use only the 80% training set in a 5-fold cross-validation loop to choose hyperparameters, then retrain the model with the chosen hyperparameters on the full 80% training set and evaluate it once on the 20% test set.**
- (b) Perform 10-fold cross-validation on the full dataset. Choose the hyperparameters that minimize the average cross-validation error, and then report this same average cross-validation error as the final test performance of the model.
- (c) Randomly split the data into 60% training, 20% validation, and 20% test. Use both the validation set and the test set to try different hyperparameters and pick the one that gives the lowest average error across these two sets.
- (d) Randomly split the data into 70% training and 30% validation. Use the validation set to tune hyperparameters and report the final validation error as the test performance.

Solution: The answer is A. A proper train/validation/test protocol requires that:

- The training (or training folds) are used to *fit* the model.
- A validation mechanism (e.g., cross-validation) within the training data is used to *tune hyperparameters*.
- A separate test set, never used in training or hyperparameter tuning, is used *once* to estimate generalization error.

Option A respects this: cross-validation happens only on the training portion, and the test set is held out until the very end. Options B, C, and D all reuse data intended for evaluation in the tuning procedure or fail to maintain a dedicated test set, leading to optimistically biased performance estimates.

11. One Answer 1 point

You are clustering a dataset in $\mathbb{R}^d$ using either k -means or a Gaussian Mixture Model (GMM)

with K components and parameters learned by EM. Which of the following statements about the relationship between k -means and GMMs is most accurate?

- Ⓐ **k -means can be viewed as a special case of a GMM with K Gaussian components that are all spherical with equal variance, equal mixing weights, and where we take the limit as the variance $\sigma^2 \rightarrow 0$ and use hard cluster assignments instead of soft responsibilities.**
- Ⓑ A GMM with K components is exactly equivalent to k -means with K clusters for any choice of covariance matrices; they optimize the same objective and always produce identical cluster assignments.
- Ⓒ k -means is more general than GMMs, because k -means can learn arbitrary cluster shapes while GMMs are restricted to Gaussian clusters.
- Ⓓ A GMM with diagonal covariance matrices is equivalent to running k -means independently on each feature dimension and then combining the 1D clusterings.

Solution: The answer is A.

A Gaussian Mixture Model with K components defines a density

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k),$$

and EM learns soft assignments (responsibilities) of each data point to each component and updates both the means and covariances.

If we restrict the GMM so that:

- All covariances are spherical and equal: $\Sigma_k = \sigma^2 I$ for all k ,
- All mixing weights are equal: $\pi_k = 1/K$,
- And we take the limit $\sigma^2 \rightarrow 0$ while using hard assignments (each point assigned entirely to the component with highest responsibility),

then maximizing the GMM log-likelihood becomes equivalent to minimizing the sum of squared distances between points and their assigned cluster centers, which is exactly the k -means objective. In this sense, k -means can be seen as a limiting special case of a GMM with strong constraints.

Option B is incorrect because general GMMs and k -means optimize different objectives and may produce very different clusterings. Option C is backwards: k -means assumes spherical, equal-variance clusters in the Euclidean metric, while GMMs with full covariances can model much richer (elliptical) cluster shapes. Option D is also incorrect: running k -means independently per feature dimension ignores cross-feature correlations and is not equivalent to fitting a diagonal-covariance GMM.

12. In this question, let us study the curse of dimensionality. Imagine that we discretize the feature space by dividing each dimension into 10 equally sized bins.

a. 1 point

Calculate the total number of grid cells (hypercubes) created in a:

A. 1D line ($d = 1$): _____B. 2D plane ($d = 2$): _____C. 3D cube ($d = 3$): _____b. 1 pointSuppose you have a fixed dataset of $N = 500$ points. Using your answer to part *a*, explain why the data becomes “sparse” as the dimension increases.

c. 1 point

How does Principal Component Analysis (PCA) help mitigate the curse of dimensionality?

Solution: Part a:

1. 10 bins
2. 100 bins
3. 1000 bins

Part b: In 1D, the average density of the bins is 50 points/bin; the space is dense. In 2D, the average density of the bins is 5 points/bin; the space is moderately populated. In 3d, the average is 0.5 points/bin; half of the bins are empty.

As dimension d increases, the volume of the space grows exponentially (10^d). With a fixed number of data points, the density drops to near zero, making the data extremely sparse.

Part c: The "Curse" implies that we need exponentially more data to generalize correctly in high dimensions. PCA helps by finding a lower-dimensional subspace (the top k principal components) that captures the majority of the variance. By projecting the data onto this subspace (reducing d to k), we increase the density of the data, making it easier for models to learn patterns without overfitting.

13. One Answer 1 point

Which of the following statements correctly distinguishes **random forests** from **gradient-boosted trees**?

- (a) **Random forests can be trained in parallel because each tree is independent, whereas gradient-boosted trees must be trained sequentially because each new tree attempts to correct the errors of the previous ones.**
- (b) Random forests utilize "stumps" (trees with only one split) to prevent overfitting, while gradient boosted trees require deep, fully grown trees to capture complex patterns.
- (c) Random forests use all available features at every split to ensure the strongest predictors are used, whereas gradient boosting adds randomness by randomly ignoring some strong predictors when constructing splits.
- (d) Random forests focus on reducing bias by increasing the depth of trees, while gradient boosted trees focus on reducing variance by aggregating across multiple trees.

Solution:

- (a) is correct. Random forests build multiple decision trees independently using bootstrap samples of the data. Since the construction of one tree does not depend on the state of another, the training process can be fully parallelized. On the other hand, gradient-boosted trees are built sequentially, as each tree is trained to predict the errors of the previous ensemble.
- (b) is incorrect: Random forests typically use deep, fully grown trees (low bias, high variance) and reduce variance through averaging. Gradient boosting typically uses shallow trees (weak learners, often with high bias) and reduces bias through iterative correction.
- (c) is incorrect: Random forests choose a random subset of features at each split, not

all available features. This decorrelates the trees, improving the ensemble's performance.

- (d) is incorrect: Random forests primarily focus on reducing variance by averaging independent trees, while Gradient Boosting focuses on reducing bias by sequentially fixing errors.

14. Select All That Apply 1 point

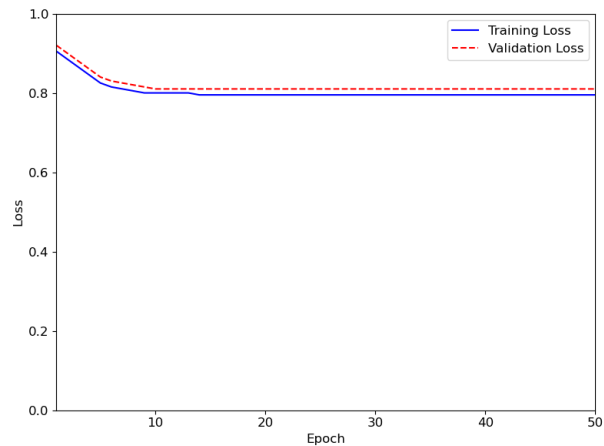
Select all statements that are true about the bootstrap.

- ☒ **(a) Bootstrapping is usually used to estimate the sampling distribution and standard error of a statistic, especially when a closed-form solution is not available.**
- ☐ **(b) In bootstrapping, each resampled dataset is sampled without replacement, so each observation appears at most once (unless there were duplicates in the underlying sample data).**
- ☒ **(c) Bootstrapping assumes that the empirical distribution of the sample data is a good approximation of the true, unknown population distribution.**
- ☒ **(d) Bootstrapping is also used when constructing random forests. Specifically, each tree in the random forests is trained on a different bootstrapped sample of the training data.**

Solution: Option A is true since it is the exact use case for bootstrap: we use it to get statistical estimates of a sampling distribution, and it's especially useful when a closed-form solution isn't available. Option B is false, because bootstrapping uses sampling with replacement; each observation can appear more than once. Option C is true, because this assumption is what allows us to use bootstrap to get estimates about a sampling distribution; if this assumption doesn't hold, our estimates will be useless. Option D is true, as we covered in class.

15. 2 points

You are training a neural network and get the following plot after plotting your training and validation loss.



- (a) Give two possible reasons for this training behavior, and give a brief explanation for each reason.

- (b) Suppose you were using sigmoid activations in your original network. Propose an alternative activation function you could use, and explain why it might be preferable over sigmoid in this situation.

Solution: For part a, valid reasons include:

1. Too simple of a model (underfitting) - would hinder learning since the model isn't able to learn the features
2. Bad weight initialization - as discussed in section, bad weight init can result in a lack of learning

3. Vanishing gradients - can prevent the model from learning since gradients would be too small and layers won't get updated
4. Bad hyperparameters (e.g. too high of a learning rate) - would prevent learning
5. Stuck in a local minimum - leads to suboptimal solution

For part b, the Sigmoid activation function "kills" gradients (as discussed in section), meaning it could result in vanishing gradients and thus poor learning; an example of an activation function that doesn't have this issue is ReLU (or a variant like Leaky ReLU).

16.

Suppose a professor at UW is developing a model to predict whether or not an undergraduate CSE student will attend graduate school ($y \in \{0, 1\}$), based on a variety of input features x . They would like to formulate this as a Maximum Likelihood Estimation problem and train a logistic regression model accordingly.

(a) 1 point

What is $P_{\theta}(y|x)$, i.e., the form of the conditional probability distribution used to model the binary labels as a function of the model weights θ ?

(b) 1 point

Given a set of independent and identically distributed training data $\{(x_1, y_1), \dots, (x_n, y_n)\}$, what is the likelihood function?

(c) 1 point

Given the likelihood function, how do we get the trained weights $\hat{\theta}$?

Solution:

- (a) Since y is binary, this follows a Bernoulli distribution:

$$P_{\theta}(y|x) = (\sigma(\theta^T x))^y (1 - \sigma(\theta^T x))^{1-y}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function.

- (b) The likelihood function $L(\theta)$ is the product of the individual probabilities for all n data points:

$$L(\theta) = \prod_{i=1}^n P_{\theta}(y_i|x_i) = \prod_{i=1}^n (\sigma(\theta^T x_i))^{y_i} (1 - \sigma(\theta^T x_i))^{1-y_i}$$

- (c) We find $\hat{\theta}$ through maximum likelihood estimation, i.e., finding θ that maximizes the likelihood function; more specifically, we typically do this through gradient descent on the negative log-likelihood.

17. Consider a binary logistic regression model that predicts $Y = 1$ whenever $\mathbb{P}(Y = 1|X) \geq \mathbb{P}(Y = 0|X)$. Each data point $x \in \mathbb{R}^2$ has two dimensions, x_1 and x_2 , that are mapped to features

using $\phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ 1 \end{bmatrix}$.

The model parameters are weights w_1, w_2 and a bias term b .

- (a) 1 point

Write the equation for the decision boundary.

(b) 1 point

If $w_1 = w_2 > 0$ and $b < 0$, what is the geometric shape of the resulting decision boundary?

Solution:

1. $w_1x_1^2 + w_2x_2^2 + b = 0$
2. A circle.

18. 1 point

Is this claim True or False? Briefly explain your answer.

“ $w_j = 0$ means that feature j has no predictive power for the true label y ”

Solution: A weight $w_j = 0$ means that feature j contributes no *additional* information for predicting y **given the other features in the model**.

However, the feature itself might still be highly predictive. For example, if you have two identical features x_1 and x_2 (perfectly correlated), the model might assign w_1 a non-zero value and $w_2 = 0$. In this case, x_2 has the same predictive power as x_1 , but its weight is zero because it is redundant.

19. Consider a simple neural network for binary classification with one hidden layer. For an input $x \in \mathbb{R}$, let the hidden layer be $a = \text{ReLU}(w_1x)$ and the output layer be $\hat{y} = \sigma(w_2a)$, where

$w_1, w_2 \in \mathbb{R}$ are the weights and $\sigma(\cdot)$ is the sigmoid activation function. Assume we use the cross-entropy loss function $L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$, where $y \in \{0, 1\}$ and the log is the natural log.

Recall that $\text{ReLU}(x) = \max(0, x)$.

(a) 1 point

Show that the loss function is not convex. Hint: Fix a single training point (x, y) to be $(1, 1)$ and give examples of (w_1, w_2) assignments where the convexity property does not hold.

(b) 1 point

ReLU is non-differentiable at $x = 0$. What method can we use to optimize non-differentiable, non-convex loss functions?

Solution:

(a) Give a counter example of (w_1, w_2) where

$$f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y)$$

does not hold. Fix a training point $(x, y) = (1, 1)$. Compute three separate cases with $\theta = (w_1, w_2)$ equal to $(1, 1)$, $(0, 0)$, and $(-1, -1)$. Case 1, $L(\theta_1) = -\log \sigma(1)$. Case 2, $L(\theta_{mid}) = -\log \sigma(0) = \log 2$. Case 3, $L(\theta_3) = -\log \sigma(0) = \log 2$. Assume $\lambda = 0.5$, we show that

$$L(\theta_{mid}) = \log 2 > 0.5L(\theta_1) + 0.5L(\theta_3) = 0.5(-\log \sigma(1) + \log 2)$$

(b) Sub-gradient descent; though we will give full points for just ‘gradient descent’.

20. 1 point

Recall that the trace of a matrix is the sum of its eigenvalues. Consider matrix A . Let $USV^\top$ be the SVD decomposition of A such that

$$S = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}.$$

Suppose L is the best rank-2 approximation of A . What is the trace of $L^\top L$?

Trace $(L^\top L) =$ _____

Solution: 13. Recall that the best rank-2 approximation of A is $L = U_{1:2}S_{1:2,1:2}V_{1:2}^\top$ (lecture 15 slide 7). Note that

$$S_{1:2,1:2} = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix},$$

the diagonal of which are the roots of the eigenvalues of $L^\top L$ (lecture 15 slide 6). So

$$\begin{aligned} \text{Trace}(L^\top L) &= 3^2 + 2^2 \\ &= 13. \end{aligned}$$

21. One Answer 1 point

Consider the data matrix

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d},$$

and feature mapping $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^p$. Let $K \in \mathbb{R}^{n \times n}$ be the corresponding kernel matrix, where $K_{ij} = \phi(x_i)^\top \phi(x_j)$. For K to be invertible, which of the following must be true?

Ⓐ $n \geq d$

Ⓑ $n \geq p$

Ⓒ $p \geq d$

Ⓓ $p \geq n$

Solution: Define

$$\phi(X) = \begin{bmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{bmatrix} \in \mathbb{R}^{n \times p}.$$

Note that $K = \phi(X)\phi(X)^\top$. Recall that for $X \in \mathbb{R}^{n \times d}$, $X^\top X$ is invertible only if $n \geq d$. Analogously, since $\phi(X)^\top \in \mathbb{R}^{p \times n}$, $\phi(X)\phi(X)^\top$ is invertible only if $p \geq n$.

22. 1 point

Let $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$, and $\lambda > 0$. Show that the function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ defined by

$$f(w) = \|Xw - y\|_2^2 + \lambda \|w\|_2^2$$

is convex.

Solution: Second derivative test:

$$\nabla f(w) = 2X^\top(Xw - y) + 2\lambda w \Rightarrow \nabla^2 f(w) = 2X^\top X + 2\lambda I.$$

Note that $2(X^\top X + \lambda I)$ is PSD, so $f(w)$ is convex.

23. 2.5 points

The following are common machine learning tasks for neural networks.

- a) Standard regression
- b) Multi-class classification
- c) Binary classification
- d) Multi-label classification (binary labels that are non-exclusive)
- e) Regression of a non-negative value

For each type of output layer for a neural network, write the letter corresponding to the task it is best used for. Assume $k > 1$.

1. 1-wide ReLU layer: _____

2. 1-wide linear layer: _____

3. k -wide softmax layer: _____

4. 1-wide sigmoid layer: _____

5. k -wide sigmoid layer: _____

Solution:

1. e. ReLU is not typically the final layer of a neural network because it clips values to only be positive; this is not typically a desired behavior, unless we are performing regression on a target that we know must be non-negative.

2. a. Regression is typically done with a linear output because the task is to predict a continuous value.
3. b. The output of softmax looks like a probability distribution across k options. This can be used for classifying an input into 1 of k classes.
4. c. You only need 1 neuron to do binary classification. Labels can be 0 and 1. With sigmoid as the final layer, it will output 0 for large magnitude negative numbers, and 1 for large magnitude positive numbers.
5. d. In multi-label classification, classes are non-exclusive (e.g., an image can contain both a cat and a dog). A k -wide sigmoid layer produces k independent probabilities between 0 and 1, effectively treating each output as a separate binary classifier. This allows multiple classes to be predicted as "present" simultaneously, unlike Softmax which forces the probabilities to sum to 1.

24. 2 points

Consider a 2-layer neural network for regression without bias terms:

- Input dimension: 2.
- Hidden layer: 2 units with ReLU activation,

$$h = \text{ReLU}(Wx), \quad W \in \mathbb{R}^{2 \times 2}.$$

- Output: scalar prediction

$$\hat{y} = v^\top h, \quad v \in \mathbb{R}^2.$$

- Loss (for a single example): squared error

$$L = \frac{1}{2}(\hat{y} - y)^2.$$

For a single training example, suppose

$$x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad y = 2, \quad W = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix}, \quad v = \begin{bmatrix} 1 \\ -2 \end{bmatrix}.$$

Assume $\text{ReLU}(z) = \max(0, z)$ is applied elementwise.

- (a) Compute the hidden activations h and the prediction $\hat{y}$.

(b) Compute the loss L .

(c) Compute the gradient $\nabla_v L$.

(d) Compute the gradient $\nabla_W L$.

Solution: First compute the pre-activations of the hidden layer:

$$Wx = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 - 2 \\ -1 - 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}.$$

Applying ReLU elementwise,

$$h = \text{ReLU}(Wx) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Then

$$\hat{y} = v^\top h = \begin{bmatrix} 1 & -2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0.$$

The loss is

$$L = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(0 - 2)^2 = 2.$$

For the gradients, note

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = -2.$$

So

$$\frac{\partial L}{\partial v} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial v} = (\hat{y} - y) h = -2 \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

For W , the gradient flows through ReLU. Since both pre-activations are negative, the ReLU derivative is 0 for both hidden units, so

$$\frac{\partial L}{\partial W} = 0_{2 \times 2}.$$

This example illustrates a “dead ReLU” situation where all gradients are zero.

25. 2 points

Consider the 1D quadratic loss

$$f(w) = \frac{1}{2}aw^2, \quad a > 0,$$

and gradient descent with fixed step size η :

$$w_{t+1} = w_t - \eta \nabla_w f(w_t), \quad w_0 \neq 0.$$

- (a) Derive the gradient descent update rule explicitly, and solve the recurrence to obtain a closed-form expression for w_t in terms of t , w_0 , a , and η .

- (b) Using your expression from part (a), give a **necessary and sufficient condition** on the step size η for gradient descent to converge to the minimizer $w^* = 0$ (i.e., $w_t \rightarrow 0$ as $t \rightarrow \infty$).

Solution: The gradient is $f'(w) = aw$, so the update is

$$w_{t+1} = w_t - \eta aw_t = (1 - \eta a)w_t.$$

Solving the recurrence gives

$$w_t = (1 - \eta a)^t w_0.$$

Convergence requires $|1 - \eta a| < 1$, which is equivalent to

$$0 < \eta < \frac{2}{a}.$$

The three cases follow from the sign and magnitude of the factor $r = 1 - \eta a$:

- **Monotone convergence:** $0 < 1 - \eta a < 1 \Rightarrow 0 < \eta < \frac{1}{a}$.
- **Oscillatory convergence:** $-1 < 1 - \eta a < 0 \Rightarrow \frac{1}{a} < \eta < \frac{2}{a}$.
- **No convergence:** $\eta \leq 0$ or $\eta \geq \frac{2}{a}$.

26. 2 points

For each of the following statements about regularized linear regression, circle **True** or **False** and briefly justify your answer in one sentence.

- (a) **True / False:** For any feature matrix $X \in \mathbb{R}^{n \times d}$, the solution to

$$\min_w \|y - Xw\|_2^2 + \lambda \|w\|_0$$

is guaranteed to be unique for all $\lambda > 0$.

- (b) **True / False:** For an orthonormal feature matrix X (columns orthonormal), lasso and ridge generally select exactly the same non-zero coordinates, just with different magnitudes.

- (c) **True / False:** For any feature matrix X , increasing λ in lasso regression can never make a coefficient increase in magnitude.

- (d) **True / False:** For any feature matrix X , increasing λ in ridge regression can never make a coefficient increase in magnitude.

Solution: (a) False. The ℓ_0 -penalized objective is non-convex, so multiple distinct sparse minimizers can exist.

(b) False. With orthonormal columns, lasso performs coordinate-wise soft-thresholding and can set some coefficients to zero, unlike ridge.

(c) False. While the overall trend is shrinkage, strong collinearity can cause individual coordinates to change non-monotonically with λ .

(d) False. Similar to (c), while the overall trend is shrinkage, strong collinearity can cause individual coordinates to change non-monotonically with λ .

27. 1 point [Extra credit.]

In this question, we will study the representer theorem for logistic regression, as a special case of a representer theorem for kernelized models. It gets its name because it shows that the solution can be directly represented as a combination of the training data.

Assume that we have training data $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$, and are training a L_2 -regularized logistic regression model by optimizing the MLE training objective

$$\hat{w} = \arg \min_w \frac{1}{n} \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i)) + \lambda \|w\|_2^2.$$

Assume that $\lambda > 0$ and that d can be larger than n . Prove that the MLE solution $\hat{w}$ can always be expressed as a linear combination of the training points $x_1, x_2, \dots, x_n$.

Hint: Decompose $\hat{w}$ into $\hat{w}_\parallel$ and $\hat{w}_\perp$, where the former is in the span of the training data and the latter is in the orthogonal complement of the span.

Solution: This is similar to what we covered in section 7. Note that $\hat{w}_\perp$ does not affect predictions so it doesn't affect the first term in the loss, and it strictly increases the second term, so it has to be 0.

28. 1 point [Extra credit.] Surprisingly, we can show that not only does the representer theorem hold at convergence, but that it also holds for every gradient descent iterate (given the right initialization).

Continuing from the previous question, now assume that we use stochastic gradient descent (with batch size 1) to train our model, initialized with $w_0 = 0$. Prove that every iterate $w_1, w_2, \dots$ obtained by running stochastic gradient descent can always be expressed as a linear

combination of the training points $x_1, x_2, \dots, x_n$.

Hint: Consider a proof by induction. At time 0, $w_0 = 0$ is in the span of the training data.

Solution: Deriving the SGD update rule shows that if the weights at the previous time step is in the span of the training points, then the weights at the next time step is still in the span of the training points.

ID: _____

Page 28

END OF EXAM

ID: _____

Page 29

Extra page for scratch paper

ID: _____

Page 30

Extra page for scratch paper