CS 189/289A Introduction to Machine Learning
Spring 2025 Jonathan Shewchuk Final

- Please do not open the exam before you are instructed to do so. Fill out the blanks below now.
- **Electronic devices are forbidden on your person**, including phones, laptops, tablet computers, headphones, and calculators. Turn your cell phone off and **leave all electronics at the front of the room**, or **risk getting a zero** on the exam. Exceptions are made for car keys and devices needed because of disabilities.
- When you start, the **first thing you should do** is **check that you have all 13 pages and all 6 questions**. The second thing is to please **write your initials at the top right of every page after this one** (e.g., write "JS" if you are Jonathan Shewchuk).
- The exam is closed book, closed notes except your two cheat sheets.
- You have **180 minutes**. (If you are in the Disabled Students' Program program and have an allowance of 150% or 200% time, that comes to 270 minutes or 360 minutes, respectively.)
- Mark your answers on the exam itself in the space provided. Do **not** attach any extra sheets. If you run out of space for an answer, write a note that your answer is continued on page 6.
- The total number of points is 150. There are 18 multiple choice questions worth 4 points each, and 5 written questions worth a total of 78 points.
- For multiple answer questions, fill in the bubbles for **ALL correct choices:** there may be more than one correct choice, but there is always at least one correct choice. **NO partial credit** on multiple answer questions: the set of all correct answers must be checked.

First name	
Last name	
SID	
Name and SID of student to your left	
Name and SID of student to your right	

Q1. [72 pts] Multiple Answer

For multiple answer questions, fill in the bubbles for **ALL correct choices:** there may be more than one correct choice, but there is always at least one correct choice. **NO partial credit** on multiple answer questions: the set of all correct answers must be checked.

(a) [4 pts] Select the true statements about **entropy and information gain in decision trees.**

○ A: The information gain of a split can be negative.

● B: The information gain of a split is zero only when both children have the same proportions p_C of each class C as the parent (or one child is empty).

● C: The entropy is a strictly concave function of the proportions p_C of each class C in a treenode.

○ D: Entropy works well for two-class classification, but does not apply if there are three or more classes.

- A: False. Information gain is always non-negative; it is zero when the class distribution doesn't change across the split.
- B: True. If the class proportions in both children match the parent, there's no gain—the split didn't help reduce uncertainty.
- C: True. Entropy is strictly concave (see page 79 of the lecture notes), and this property penalizes splits that evenly mix classes, thereby favoring splits that push impurity into fewer branches.
- D: False. Entropy is defined for multi-class classification too.

(b) [4 pts] Select the true statements about **random forests.**

○ A: Each training point for a random forest is stored in every tree in the forest.

● B: To classify a test point with a random forest, we find a leaf node in every tree in the forest.

● C: A single deep decision tree is usually more interpretable than a random forest of 100 shallow decision trees, when trained on the same data.

○ D: In a random forest, we randomly sample features at each treenode to reduce the tree's bias.

- A is False. Random forests use bagging.
- B is True. The decision tree classification algorithm searches down the tree until it reaches a leaf, and the ensemble classification algorithm runs every tree.
- C is True: averaging sacrifices interpretability.
- D is False: randomly sampling features when choosing a split **increases** the bias, as the best features are less frequently selected.

(c) [4 pts] Select the true statements about **ensembles.**

○ A: Averaging the outputs of independently trained models tends to reduce model *bias*.

● B: Averaging the outputs of independently trained models tends to reduce model *variance*.

● C: Random forests often perform better when the individual decision trees are *deep*.

○ D: Random forests often perform better when the individual decision trees are *shallow*.

A: False. Bias isn't significantly reduced by ensembling.

B: True. Ensembling tends to reduce variance.

C: True. Because ensembling tends to reduce variance, but not bias, we prefer deep decision trees, which tend to have low bias but high variance.

D: False. Shallow trees tend to have high bias and low variance, and ensembling does not fix the bias.

(d) [4 pts] Select the true statements about **Bayesian classification.**

● A: Gaussian discriminant analysis assumes that the data of each class is generated from a multivariate normal distribution.

○ B: Neural networks predict a class C for a test point x by estimating posterior probabilities $P(Y = \text{C}|X = x)$ with Bayes' Theorem.

○ C: With the 0-1 loss function, the Bayes decision rule (Bayes classifier) minimizes the training error.

● D: AdaBoost can return an estimate of a posterior probability $P(Y = \text{C}|X = x)$, but this estimation algorithm is discriminative, not generative.

- A: True. GDA models $P(x \mid y)$ as a Gaussian for each class.
- B: False. Neural networks don't even try to generate the class-conditional distributions.
- C: False. The Bayes optimal classifier minimizes the expected loss (over infinitely many sample points), but the training error may be higher or lower than this expectation.
- D: True. AdaBoost can estimate a posterior $P(Y = \text{C}|x) \approx \frac{1}{1 + e^{-2M(x)}}$, but it's not generative as it doesn't even try to generate the class-conditional distributions.

(e) [4 pts] Select the true statements about ℓ_2 **regularization in regression.**

● A: Adding ℓ_2 regularization to a regression problem never decreases the training cost (mean loss over the training points) at the global minimum.

○ B: Adding ℓ_2 regularization to a regression problem never decreases the test cost (mean loss over the test points) at the training points' global minimum.

○ C: Ridge regression is a good method for feature selection.

● D: ℓ_2 regularization combined with stochastic gradient descent (SGD) behaves equivalently to SGD with weight decay.

- A: True. The regularization term is nonnegative, so the global minimum can't be lesser after we add shrinkage.
- B: False. The reason we use regularization is because it can sometimes reduce the test cost.
- C: False. Ridge regression shrinks weights, but it almost never sets any exactly to zero.
- D: True. See Lecture 22 and Discussion Section 12.

(f) [4 pts] Select the true statements about the the least-norm solution w^* to the normal equations for **least-squares linear regression.**

- ● A: w^* lies in the row space of X.
- ○ C: w^* lies in the column space of X.
- ● B: $w^* = X^+ y$, where X^+ is the Moore–Penrose pseudoinverse of X.
- ● D: Having the singular value decomposition of X makes it easy to compute w^*.

- A: True. This follows from B, as $\operatorname{col} X^+ = \operatorname{row} X$.
- B: True.
- C: False. Observe that $w^* \in \mathbb{R}^d$ whereas the columns of X have length n.
- D: True. We can easily determine the Moore–Penrose pseudoinverse from the SVD.

(g) [4 pts] Select the true statements about the standard **soft-margin support vector machine (SVM)** optimization problem: to find w, α, and ξ_i that minimizes $\|w\|^2 + C \sum_{i=1}^n \xi_i$ subject to $y_i(X_i \cdot w + \alpha) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all $i \in [1, n]$.

- ○ A: The slack variables allow the SVM to produce nonlinear decision boundaries.
- ● C: If $\xi_i < 1$, the training point X_i is on the *correct* side of the decision boundary.
- ● B: The slack variables allow the SVM to always have a solution (if C is nonnegative and finite).
- ● D: If $\xi_i > 1$, the training point X_i is on the *wrong* side of the decision boundary.

- A: False. Soft-margin SVMs can create non-linear decision boundaries if non-linear features are added to the points.
- B: True. That's why we have the slack variables.
- C: True. If $\xi_i < 1$, then $y_i(X_i \cdot w + \alpha) > 0$, implying a correctly classified point.
- D: True. The optimization problem enforces ξ_i to be the smallest possible while meeting all constraints. If $\xi_i > 1$, then $y_i(X_i \cdot w + \alpha) = 1 - \xi_i < 0$, implying a misclassified point.

(h) [4 pts] Select the true statements about ***k*-means clustering.** Assume no two sample points are equal.

- ○ A: Lloyd's Algorithm finds a global minimum of its cost function.
- ○ C: Lloyd's Algorithm finds the same clusters, regardless of how you initialize it.
- ○ B: Lloyd's Algorithm uses the average linkage to assign a distance between two clusters.
- ● D: Increasing the hyperparameter k always decreases the global minimum of the cost function (assuming there are more points than clusters).

- A: k-means clustering is NP-hard. Lloyd's Algorithm is a heuristic that finds a "local" minimum (of the alternating minimization procedure), but not always the global minimum.
- B: No, but sometimes hierarchical clustering does that.
- C: k-means is very sensitive to initialization. Different initializations can end in drastically different clusterings.
- D: True. See Discussion 12 Question 3.

(i) [4 pts] Select the true statements about the hyperparameter k in the two-class ***k*-nearest neighbor classifier.**

○ A: The test accuracy of the model tends to increase as we increase the value of k.

○ B: The approximation of the Bayes optimal decision boundary tends to improve as we increase the value of k.

● C: We tend to obtain a "smoother" (less jagged) decision boundary as we increase the value of k.

● D: As the size of our training set increases, we should generally choose to increase the value of k.

A: False. While increasing k reduces variance and may help prevent overfitting, too large a k introduces high bias and can degrade accuracy.

B: False. Increasing k does not guarantee a better approximation of the Bayes decision boundary when working with a finite dataset.

C: True. A larger k leads to majority voting over a broader neighborhood, making the boundary between classes less sensitive to local noise and thus smoother.

D: True. Denser data allows for a larger k without pulling in too many points from other classes, making the classification more stable and less sensitive to noise.

(j) [4 pts] Consider a **neural network** and a fully connected layer that takes an input vector x, then computes a linear preactivation vector $a = Wx$ and a vector of hidden units $h = f(a)$. Let J be the network's cost function. Which of these derivatives **cannot** be computed during the forward pass of backpropagation, but only during the backward pass?

● A: $\nabla_x J$.

● B: $\dfrac{\partial J}{\partial W_{ij}}$.

○ C: The Jacobian $\dfrac{\partial h}{\partial a}$ if f is the identity function.

○ D: $\dfrac{\partial h}{\partial x}$ if f is the element-wise sigmoid function.

(k) [4 pts] Select the true statements about **the vanishing gradient problem for sigmoid units.**

○ A: Decreasing the learning rate often reduces its impact.

● B: Batch normalization and residual connections often reduce its impact.

● C: The partial derivative $\dfrac{\partial J}{\partial w_i}$ for a weight w_i entering a sigmoid unit becomes very small when the sigmoid's output is very close to 0 or 1.

● D: The vanishing gradient problem is more likely to cause deep networks to fail than shallow ones.

A: False. A different learning rate does not change the derivative, and a smaller rate makes the stuck units change even more slowly.

B: True. Batch normalization keeps inputs in the linear region and residual connections permit gradients to backpropagate around layers, both alleviating the issue.

C: True. In the saturated regions ($\sigma(z) \approx 0$ or 1) the derivative $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ is nearly 0, so back-propagated gradients shrink.

D: True. Through multiple layers of backprop, the gradients can become extremely tiny.

(l) [4 pts] Select the true statements about **output units and initialization in neural networks.**

● A: Initializing all the weights to the same value can create symmetries that prevent some hidden units from differentiating from each other.

○ B: Using a linear activation at the output units guarantees lower training loss than using logistic (sigmoid) output units.

● C: When a neural network has multiple output units, hidden units may learn features that are useful for more than one output.

○ D: The cost function of a neural network is convex if all the activation functions are convex.

- A: True. Page 92 explains that identical initial weights cause symmetry in hidden units that gradient descent cannot break.
- B: False. Logistic outputs are often used because they model probabilities and help constrain outputs—linear isn't always better.
- C: True. Page 91: "some [classifiers] come out better because they can take advantage of particularly useful hidden units..."
- D: False. ReLUs have convex activation functions (the ramp function), but the cost function is nevertheless not even close to convex.

(m) [4 pts] Select the true statements about **shared weights in convolutional neural networks.**

● A: If a filter learns to detect edges, edge detection will be applied to every patch in the image.

○ B: The idea of shared weights was inspired by an axon that giant squids use for fast propulsion by expelling jets of water.

● C: Shared weights makes it less likely that any one weight will become unreasonably large.

○ D: In practice, most convolutional neural networks use manually designed filters such as Sobel or Gabor filters to maximize test accuracy.

- A: True. The convolution operation slides the filter over all parts of the image, so an edge detector will be applied on each patch.
- B: False. Shared synapse weights are not known to occur in nature. Squid axons were used to characterize the mathematics of action potentials, however.
- C: True. It's unlikely that a weight will become spuriously large if it's used in many places.
- D: False. Typical ConvNets *learn* all the filters rather than using hand-crafted ones.

(n) [4 pts] Select the true statements about **layers of edges in convolutional neural networks.**

○ A: Increasing the stride of a convolutional layer decreases the number of weights in the layer.

○ B: An average-pooling layer that reduces $4j$ hidden units to j hidden units has more weights than a max-pooling layer that reduces $4j$ hidden units to j hidden units. (Assume $j \geq 4$.)

● C: Increasing the stride of a convolutional layer decreases the number of hidden units in the activation maps computed by the layer.

● D: A fully-connected layer of edges that connects j hidden units to j hidden units has more weights than a batch normalization layer that connects j hidden units to j hidden units. (Assume $j \geq 4$.)

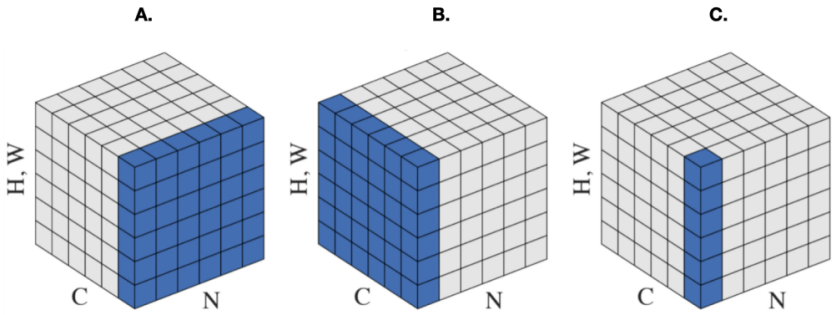
A: False. The number of parameters in a convolutional layer depends only on the size of the filter and the number of filters.

B: False. Both types of pooling layers have no weights at all.

C: True. The filter will be applied over fewer patches of the input. Strides greater than one are used to downsample images and activation maps.

D: True. The fully-connected layer has $j^2 + j$ weights (including j bias terms), whereas the batch normalization layer has just $2j$ trainable weights (including the bias terms).

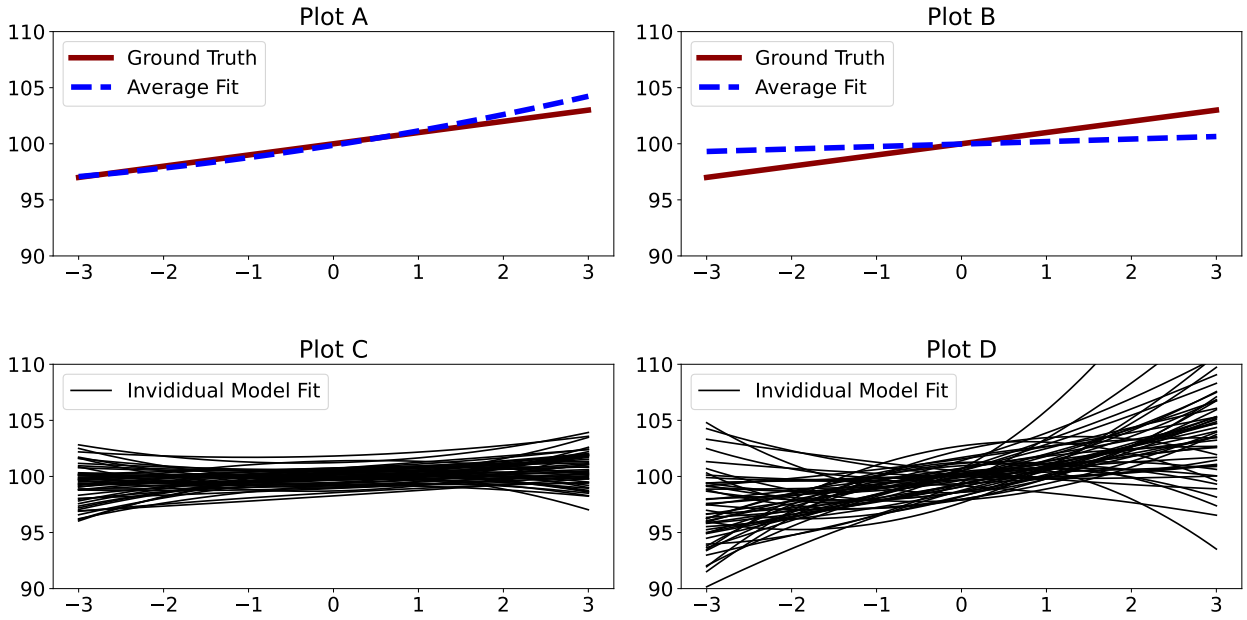
(o) [4 pts] In the following diagram, the dimension labeled C represents the different channels, the dimension labeled N represents the training points (images) in the minibatch, and the dimension labeled H, W (height and width) represents the pixels or the units in an activation map. The small shaded cubes in each big cube represent a subset of units that contribute to a single mean (and a single variance) computed by **batch normalization or layer normalization for images.** Select the true statements.



- ● A: Figure A correctly depicts batch normalization.
- ○ B: Figure B correctly depicts batch normalization.
- ○ C: Figure C correctly depicts batch normalization.
- ○ D: Figure A correctly depicts layer normalization.

Batch normalization is A. Layer normalization is B.

(p) [4 pts] We try to determine the ground truth relationship between two parameters by taking some measurements. The measurements are noisy, so we try to determine the relationship by using either **ordinary least-squares polynomial regression** or **ridge regression (with the same polynomial features)**. We run multiple experiments, each time sampling new data and fitting another polynomial model. Plot C and Plot D display all of the experimental polynomial models for either ordinary or ridge regression. Plot A and Plot B each display two curves: the ground truth relationship and the average of our experimental polynomial models. (The ground truth relationship is the same in both figures.)



Select the correct statements.

- ● A: Plot A shows the average fit of ordinary least-squares, and Plot B shows the average fit of ridge regression.
- ○ B: Plot A shows the average fit of ridge regression, and Plot B shows the average fit of ordinary least-squares.
- ○ A: Plot C shows the models from ordinary least-squares, and Plot D shows the models from ridge regression.
- ● B: Plot C shows the models from ridge regression, and Plot D shows the models from ordinary least-squares.

Ridge regression has a higher bias because it has a regularization term, which leads to a smaller slope but (in this case) a worse fit of the data.

Ridge regression has a lower variance because it has a regularization term, which makes the model less sensitive to fluctuations in the data.

(q) [4 pts] Consider this design matrix, representing three training points in $\mathbb{R}^2$.

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

What percentage of the variability in the data is captured by the **first principal component**?

- ○ A: 33%
- ○ B: 50%
- ○ C: 60%
- ● D: 75%

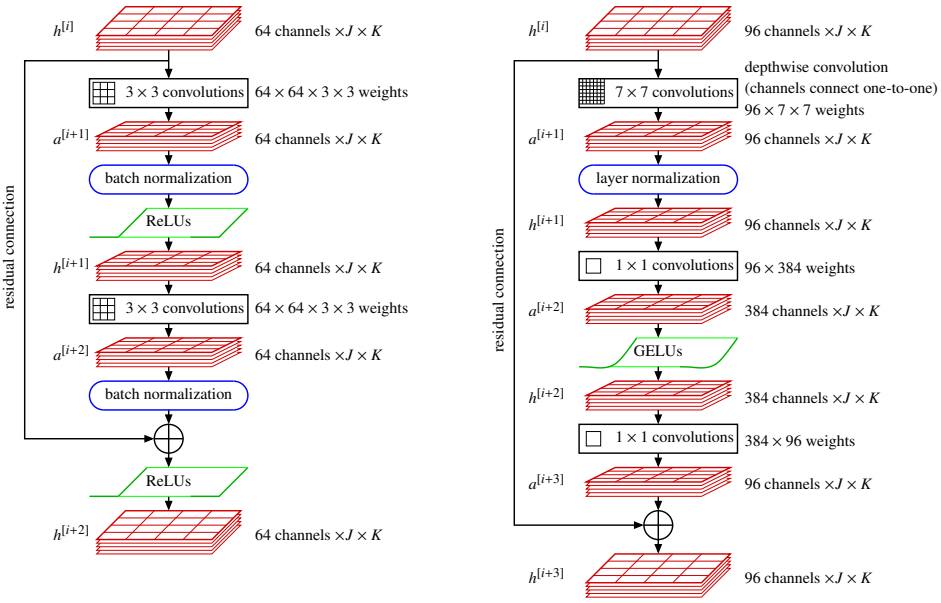
Our data matrix is already centered (i.e., its means is $[0, 0]$). Therefore, we can answer this question by looking at the eigenvalues of $X^\top X$.

$$X^\top X = \begin{bmatrix} 6 & 0 \\ 0 & 2 \end{bmatrix}.$$

This is a diagonal matrix, so we can read off its eigenvalues: $\lambda_2 = 6$ and $\lambda_1 = 2$. The principal component is the eigenvector of $X^\top X$ corresponding to the largest eigenvalue, $\lambda_2 = 6$, and the percentage of variability captured by this direction is

$$\frac{\lambda_1}{\lambda_1 + \lambda_2} * 100\% = \frac{6}{8} * 100\% = 75\%.$$

(r) [4 pts] Select the true statements about these two **residual neural network** motifs (building blocks).



○ A: If we want to make it as easy as possible for the network to be capable of supporting an identity map from an early layer to a layer later, the architecture on the left has the advantage.

○ B: In the architecture on the right, the 1×1 convolutional layers learn to detect edges.

● C: The right architecture separates channel mixing from spatial mixing. Both kinds of mixing take place, but they take place in different convolutional layers.

○ D: The left architecture's reliance on batch normalization is primarily motivated by BatchNorm's superior ability to mix information across channels.

A: False. Placing the ReLU after the addition is harmful to this goal. The right architecture has the advantage.

B: False. A 1×1 convolutional filter cannot detect edges.

C: True. Channel mixing happens only in the 1×1 convolutions, whereas spatial mixing happens only in the 7×7 depthwise convolutions. See Lecture 23.

D: False. The primary motivation for BatchNorm is to stabilize training in deep networks. It does not mix information across channels.

Extra space: if you need extra space for your answer to a written problem, you may write here. **Be sure to write "see page 6" under the unfinished answer!**

Q2. [14 pts] The Compact SVD, the Pseudoinverse, and PCA

(a) [4 pts] Write down the form of the **compact singular value decomposition** (SVD) of an $n \times d$ matrix X. What is the **size** of each matrix in the decomposition? What **constraints** does each matrix in the decomposition have to satisfy for it to be a compact SVD?

Let r be the rank of X. The compact SVD is written $X = UDV^\top$, where
U is an $n \times r$ matrix satisfying $U^\top U = I$,
D is a diagonal, invertible $r \times r$ matrix, and
V is a $d \times r$ matrix satisfying $V^\top V = I$.

(b) [2 pts] How do we use the compact SVD to **define the pseudoinverse** X^+ of X? **Explain why** this definition would fail if you used the full SVD (the first SVD taught in lecture) instead of the compact SVD.

$X^+ = VD^{-1}U^\top$.
This definition does not work with the full SVD because in the full SVD, D might not be invertible. Alternatively, you could point out that D might not even be square.

(c) [4 pts] Use the compact SVD to **prove** that for any matrix X, $XX^+X = X$. In every case where two matrices "cancel" each other out, **explicitly indicate** which property you wrote down in (a) justifies that cancellation.

$$\begin{aligned} XX^+X &= UDV^\top VD^{-1}U^\top UDV^\top \\ &= UDD^{-1}DV^\top \qquad \text{because } U^\top U = I \text{ and } V^\top V = I \\ &= UDV^\top \qquad \text{because } D \text{ is invertible} \\ &= X. \end{aligned}$$

(d) [4 pts] Suppose that X is a **centered** design matrix. We want to perform **principal components analysis** (PCA) on X, yielding a number of principal components equal to the rank of X. (Note: you should know how the principal components of X relate to its SVD, but you don't have to prove it.)

We are also given a $m \times d$ matrix Z of m "test points"—that is, each row of Z is a test point in d-dimensional space—and we want to find the **principal coordinates of each test point**, with respect to the principal components of X.

What is the **simplest formula** we can write to compute a matrix that lists every principal coordinate of every test point? **Explain why your formula is correct**, using the definition of the projection of a point onto a principal component.

The simplest formula is ZV, but we will also accept $V^\top Z^\top$. Each column v_j of V is a (unit) principal component of X. The multiplication ZV has the effect of taking the inner product of every row of Z with every column of V. The inner product $Z_i \cdot v_j$ yields the jth principal coordinate of the test point Z_i, because it is the projection of Z_i onto v_j.

Q3. [14 pts] Weighted k-means Clustering

Consider a modified version of k-means clustering in which we are given a set of n sample points $X_1, X_2, \ldots, X_n$ and n positive weights $w_1, w_2, \ldots, w_n \in \mathbb{R}$. Our goal is to find a vector y of labels such that each label satisfies $y_j \in [1, k]$ and y minimizes

$$\sum_{i=1}^{k} \sum_{y_j=i} w_j \left\| X_j - \mu_i \right\|^2 .$$

This is similar to classic k-means clustering, but the weight w_j describes how "important" it is for the center of X_j's cluster to be close to X_j. As usual, the input parameter $k \leq n$ is the desired number of clusters.

Please describe Lloyd's algorithm—as if you were explaining it to someone who never heard of Lloyd's algorithm—modified for this new objective function. We initialize Lloyd's algorithm by the Forgy method: choose k random sample points to be the initial μ_i's.

(a) [5 pts] Lloyd's algorithm has a step where the cluster centers μ_i are fixed and we update the labels y_j to minimize the cost function. **State whether any changes are needed** to the standard k-means label update step to accommodate our addition of weights. Then **describe this step of the algorithm** well enough that someone could implement it. Be sure to account for **cases where there might be more than one solution.**

No changes are needed to accommodate weights; this is the same step used by the standard k-means algorithm.

For each $j \in [1, n]$, we choose each label y_j to be the index i such that μ_i is the cluster center closest to X_j.

If there is a tie, and one of the choices is to not change y_j, then we don't change y_j.

(b) [6 pts] Lloyd's algorithm has a step where the labels y_j are fixed and we update the cluster centers μ_i to minimize the cost function. **State whether any changes are needed** to the standard k-means center update step to accommodate our addition of weights. Then **describe this step of the algorithm** well enough that someone could implement it. Use **calculus to derive a formula for updating the values of** μ_i.

The formula for computing μ_i is different from standard k-means.

To optimize each μ_i, we take the derivative of the cost function with respect to μ_i and set it to zero.

$$\begin{aligned}
\frac{d}{d\mu_i} \sum_{z=1}^{k} \sum_{y_j=z} w_j \left\| X_j - \mu_z \right\|^2 &= 0, \\
\sum_{y_j=i} 2w_j (X_j - \mu_i) &= 0, \\
\sum_{y_j=i} 2w_j X_j &= \sum_{y_j=i} 2w_j \mu_i, \\
\mu_i &= \left(\sum_{y_j=i} w_j X_j \right) / \left(\sum_{y_j=i} w_j \right).
\end{aligned}$$

For each $i \in [1, k]$, we use this formula to update μ_i.

This formula is clearly different than the one in standard k-means. (To be precise, it's the weighted average of the sample points in cluster i, rather than just the average.)

(c) [3 pts] How do we combine the two steps above into an algorithm? When does the algorithm stop?

Alternate between running step (a) and step (b). The algorithm stops when step (a) does not change any labels assigned during the previous iteration of step (a).

Q4. [20 pts] Decision Trees

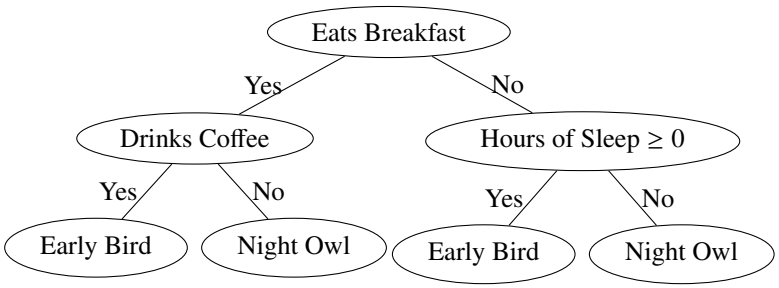
You are investigating the sleep habits of students at UC Berkeley, in hopes of learning to predict whether a student is an Early Bird or a Night Owl. You gather 12 training points and their features, listed in this table, and you decide to build a **decision tree** from this data.

i	Eats Breakfast	Drinks Coffee	Hours of Sleep	label (y_i)
1	Yes	Yes	5	Early Bird
2	Yes	No	5	Night Owl
3	Yes	Yes	9	Early Bird
4	Yes	No	4	Night Owl
5	Yes	No	7	Early Bird
6	Yes	No	9	Early Bird
7	No	Yes	4	Night Owl
8	No	Yes	5	Night Owl
9	No	Yes	6	Night Owl
10	No	Yes	7	Early Bird
11	No	No	6	Night Owl
12	No	No	8	Early Bird

(a) [4 pts] What is the **entropy** at the root node of the decision tree? **Show your work and simplify your answer as much as possible** (but do not use decimals).

$$H(S) = -p_{\text{Early Bird}} \log p_{\text{Early Bird}} - p_{\text{Night Owl}} \log p_{\text{Night Owl}} = -\frac{6}{12}\log_2\frac{6}{12} - \frac{6}{12}\log_2\frac{6}{12} = -\log_2\frac{1}{2} = 1.$$

(b) [4 pts] What is the **information gain** for the first split (the root node and its children) in this tree? (Note that this tree is **not** optimal.) **Show your work and simplify your answer as much as possible** (but do not use decimals).

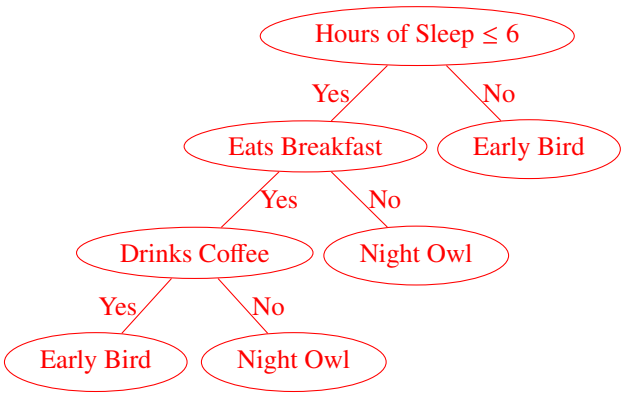


The left child and the right child have six training points each; both children have four of one class and two of the other class. Therefore, the entropy for either child is

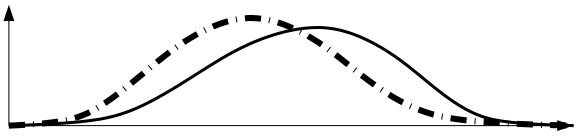
$$H(S_l) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = -\frac{4}{6}\log_2 2 + \frac{4}{6}\log_2 3 + \frac{2}{6}\log_2 3 = \log_2 3 - \frac{2}{3}.$$

and the weighted average for the two children is the same. So the entropy gain is $H(S) - H(S_l) = \frac{5}{3} - \log_2 3$.

(c) [6 pts] Construct a decision tree with pure leaves for the training points. At each treenode, choose the split with the greatest information gain. Draw your tree in the format shown in part (b), with a predicted label inside each leaf node and a splitting condition inside each internal treenode.



(d) [4 pts] Suppose we repeat the experiment, but this time we gather 10,000,000 training points (students) and we record their average sleeping times with one-second precision. We realize that the distributions of the classes Early Bird and Night Owl are strongly overlapping, akin to the following diagram (but in a three-dimensional feature space). **Explain why** a decision tree with pure leaves is not suitable, and **explain what should we do instead.** (. . . With a modified decision tree! Do not replace the decision tree with an alternative learning method.)



A decision tree with pure leaves will fail to approximate the Bayes classifier because every point gets its own leaf node, even if it is very unrepresentative of the points around it; therefore, the tree overfits quite badly. A better strategy is to stop splitting a node if it has 100 or fewer students in it, and use a vote to determine the leaf's class.

(e) [2 pts] We decide to use an ensemble of decision trees to improve our answer to (d). Which would work better: a random forest or AdaBoost? **Explain why.**

A random forest. Random forests are good at reducing variance/overfitting, whereas AdaBoost is good at reducing bias/underfitting. When outliers are a big potential problem, it's important to reduce the variance/overfitting. By contrast, this data is likely to have a simple Bayes classifier and bias should not be a problem.

Q5. [15 pts] AdaBoost

You are using AdaBoost to train a sequence of learners $G_1, G_2, \ldots$ on four training points, tabulated below. We omit the training points X_i because they are irrelevant to this problem, but we include the binary training labels $y_i \in \{-1, 1\}$ and the classes predicted by the first two learners, G_1 and G_2, for the four training points.

i	X_i	y_i	$G_1(X_i)$	$G_2(X_i)$
1	$*$	1	1	1
2	$*$	1	1	-1
3	$*$	-1	1	-1
4	$*$	-1	-1	-1

(a) [5 pts] What is the weighted training error rate err_1 of learner G_1 (as a number)? **Show your work.**

The weighted training error rate is $\frac{\sum_{y_i \neq G_1(X_i)} w_i}{\sum_{i=1}^n w_i}$, where the weights w_i are all equal at the beginning. As G_1 misclassified one point out of four, the weighted training error rate is $\text{err}_1 = 1/4$.

(b) [4 pts] After the first learner G_1 was trained, AdaBoost updated the weights w_1, w_2, w_3, and w_4 associated with the four training points. Recall that each weight is either multiplied or divided by $\sqrt{\frac{1-\text{err}}{\text{err}}}$. What are the updated values of w_1, w_2, w_3, and w_4 (that are used to train G_2)? **Show your work.**

All four weights were initialized to a value of $1/4$, though we'll accept an answer that initializes them all to 1. As X_3 was misclassified, its weight get multiplied by $\sqrt{\frac{1-1/4}{1/4}} = \sqrt{3}$, so $w_3 = \frac{\sqrt{3}}{4}$. The other three training points are correctly classified, so their weights get multiplied by $\sqrt{\frac{1/4}{1-1/4}} = \frac{1}{\sqrt{3}}$, so $w_1 = w_2 = w_4 = \frac{1}{4\sqrt{3}}$.

But we'll also accept $w_3 = \sqrt{3}$ and $w_1 = w_2 = w_4 = 1/\sqrt{3}$.

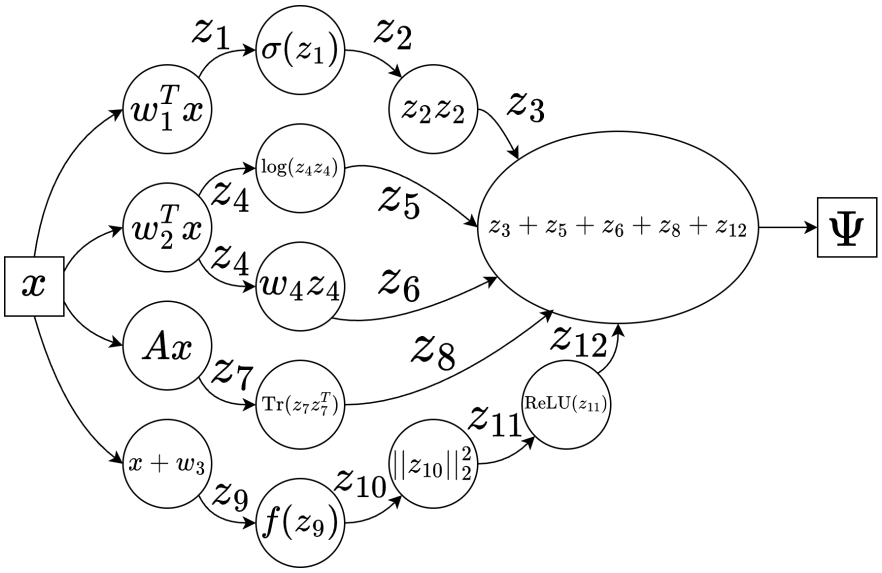
(c) [4 pts] What is the weighted training error rate err_2 of learner G_2 (as a number)? **Show your work.**

G_2 misclassifies only point X_2. The weighted training error rate is $\text{err}_2 = \frac{\sum_{y_i \neq G_2(X_i)} w_i}{\sum_{i=1}^n w_i} = \frac{1/(4\sqrt{3})}{3/(4\sqrt{3}) + \sqrt{3}/4} = \frac{1}{6}$.

(d) [2 pts] Which learner will have a bigger vote β_i in the metalearner—G_1 or G_2? **Explain your answer.**

G_2 will have a bigger vote ($\beta_2 > \beta_1$) because it has a lower weighted error rate.

Q6. [15 pts] Backpropagation in a Neural Network



Above we diagram the computational graph of a **neural network** with **input** $x \in \mathbb{R}^d$, **hidden unit** values $z_1, z_2, z_3, z_4, z_5, z_6, z_8, z_{11}, z_{12} \in \mathbb{R}$ and $z_7, z_9, z_{10} \in \mathbb{R}^d$ (column vectors), and **cost** $\Psi = z_3 + z_5 + z_6 + z_8 + z_{12} \in \mathbb{R}$, which we wish to minimize. The learnable **weights** of the neural network are $w_1, w_2, w_3 \in \mathbb{R}^d$ (column vectors), $w_4 \in \mathbb{R}$, and $A \in \mathbb{R}^{d \times d}$.

Assume $z_4 \neq 0$ so that $\log\left(z_4^2\right)$ is well-defined. This "log" is the natural logarithm (base e). Let $\mathrm{Tr}(z_7 z_7^\top)$ denote the trace of the matrix $z_7 z_7^\top$. Define the nonlinear activation functions

$$\sigma(\gamma) = \frac{1}{1+e^{-\gamma}} \quad \text{(sigmoid)}, \qquad \mathrm{ReLU}(\gamma) = \max\{0, \gamma\}.$$

(a) [5 pts] Derive the gradient $\nabla_{w_1}\Psi$ (or its transpose, $\dfrac{\partial \Psi}{\partial w_1}$; your choice). You can write it as a column vector or a row vector; either is fine. **Show your work.** Your final answer should **not** have any derivatives left unexpanded. Your answer can be defined in terms of any of the variables in the computational graph.

$$\begin{aligned}
\frac{\partial \Psi}{\partial w_1} &= \frac{\partial \Psi}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial w_1} \\
&= \frac{\partial (z_3 + z_5 + z_6 + z_8 + z_{12})}{\partial z_3}\frac{\partial (z_2 z_2)}{\partial x_2}\frac{\partial \sigma(z_1)}{\partial x_2}\frac{\partial (w_1^\top x)}{\partial w_1} \\
&= 1 \cdot 2z_2 \cdot \sigma'(z_1) \cdot x^\top \\
&= 2z_2\sigma(z_1)(1-\sigma(z_1))x^\top.
\end{aligned}$$

It would also be fine to write $2z_2\sigma(w_1^\top x)(1-\sigma(w_1^\top x))x^\top$ or $2z_2^2(1-z_2)x^\top$, for instance.

(b) [5 pts] Derive the gradient $\nabla_{w_2}\Psi$ (or its transpose, $\dfrac{\partial\Psi}{\partial w_2}$; your choice). You can write it as a column vector or a row vector; either is fine. **Show your work.** Your final answer should **not** have any derivatives left unexpanded.

$$\begin{aligned}
\frac{\partial\Psi}{\partial w_2} &= \frac{\partial\Psi}{\partial z_5}\frac{\partial z_5}{\partial z_4}\frac{\partial z_4}{\partial w_2} + \frac{\partial\Psi}{\partial z_6}\frac{\partial z_6}{\partial z_4}\frac{\partial z_4}{\partial w_2} \\
&= \frac{\partial(z_3+z_5+z_6+z_8+z_{12})}{\partial z_5}\frac{\partial\log(z_4 z_4)}{\partial z_4}\frac{\partial(w_2^\top x)}{\partial w_2} + \frac{\partial(z_3+z_5+z_6+z_8+z_{12})}{\partial z_6}\frac{\partial(w_4 z_4)}{\partial z_4}\frac{\partial(w_2^\top x)}{\partial w_2} \\
&= 1\cdot\frac{2z_4}{z_4 z_4}\cdot x^\top + 1\cdot w_4\cdot x^\top \\
&= \left(\frac{2}{z_4}+w_4\right)x^\top.
\end{aligned}$$

(c) [5 pts] Derive the gradient $\nabla_A\Psi$ (or its transpose, $\dfrac{\partial\Psi}{\partial A}$; your choice). **Show your work.** Your final answer should **not** have any derivatives left unexpanded.

By the cyclic property of the trace, $\mathrm{Tr}(z_7 z_7^\top) = \mathrm{Tr}(z_7^\top z_7) = \|z_7\|^2$, so

$$\begin{aligned}
\frac{\partial\Psi}{\partial A} &= \frac{\partial\Psi}{\partial z_8}\frac{\partial z_8}{\partial z_7}\frac{\partial z_7}{\partial A} \\
&= \frac{\partial(z_3+z_5+z_6+z_8+z_{12})}{\partial z_8}\frac{\partial\mathrm{Tr}(z_7 z_7^\top)}{\partial z_7}\frac{\partial Ax}{\partial A} \\
&= 1\cdot 2z_7 x^\top \\
&= 2z_7 x^\top.
\end{aligned}$$

(Note that this is an outer product matrix. $z_7^\top x$ or $x^\top z_7$ would be wrong.)

(d) [0 pts] **[One bonus point, for perfect execution only.** Save this question for last.**]** The function $f(\cdot)$ that produces z_{10} is a centering map (i.e., a function that subtracts the mean components of a vector from every component)

$$f:\mathbb{R}^d\to\mathbb{R}^d,\quad f(y) = y - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top y,\quad \text{where } \mathbf{1}_d = (1,\ldots,1)^\top.$$

Derive the gradient $\nabla_{w_3}\Psi$ (or its transpose, $\dfrac{\partial\Psi}{\partial w_3}$; your choice). **Show your work.** Your final answer should **not** have any derivatives left unexpanded.

$$\begin{aligned}
\frac{\partial\Psi}{\partial w_3} &= \frac{\partial\Psi}{\partial z_{12}}\frac{\partial z_{12}}{\partial z_{11}}\frac{\partial z_{11}}{\partial z_{10}}\frac{\partial z_{10}}{\partial z_9}\frac{\partial z_9}{\partial w_3} \\
&= \frac{\partial(z_3+z_5+z_6+z_8+z_{12})}{\partial z_{12}}\frac{\partial\mathrm{ReLU}(z_{11})}{\partial z_{11}}\frac{\partial\|z_{10}\|^2}{\partial z_{10}}\frac{\partial f(z_9)}{\partial z_9}\frac{\partial(x+w_3)}{\partial w_3} \\
&= 1\cdot\mathrm{ReLU}'(z_{11})\cdot 2z_{10}^\top\cdot\left(\boldsymbol{I}_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top\right)\cdot\boldsymbol{I}_d \\
&= \begin{cases} 2z_{10}^\top\left(\boldsymbol{I}_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top\right), & z_{11}\geq 0, \\ 0, & z_{11} < 0. \end{cases}
\end{aligned}$$