CS 189/289A Introduction to Machine Learning
Spring 2024 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 12 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 5.
- 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. Partial credit rules: For questions with one correct bullet, 2/4 partial credit if you check the correct bullet plus exactly one other (wrong) bullet. For questions with two correct bullets: 2/4 partial credit if you have any 3 out of 4 bullets correct. For questions with three correct bullets: 2/4 partial credit if you check 2 out of 3 correct bullets but do not check the incorrect one. For questions with four correct bullets: 2/4 partial credit if you check 3 of the 4 bullets.

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

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. Partial credit rules: For questions with one correct bullet, 2/4 partial credit if you check the correct bullet plus exactly one other (wrong) bullet. For questions with two correct bullets: 2/4 partial credit if you have any 3 out of 4 bullets correct. For questions with three correct bullets: 2/4 partial credit if you check 2 out of 3 correct bullets but do not check the incorrect one. For questions with four correct bullets: 2/4 partial credit if you check 3 of the 4 bullets.

(a) [4 pts] **Why** do we usually choose splits that maximize the information gain—equivalently, minimize the weighted average of entropies $H_{\text{after}} = \frac{|S_l|H(S_l) + |S_r|H(S_r)}{|S_l| + |S_r|}$ after the split—**instead of** computing the number of misclassified points in each child and using the weighted average of those as a cost function?

○ A: The misclassified points cost function penalizes balanced splits (where the two children have roughly equal numbers of training points) more than the entropy cost function does.

● B: The misclassified points cost function often assigns the same weighted average cost to several different splits.

● C: The entropy cost is strictly concave, so if the children have different distributions of classes (and are nonempty), the information gain is positive.

● D: The misclassified points cost is not strictly concave, so we encounter more circumstances where the weighted average of the children's costs is equal to the parent's cost.

- A: False. Information gain does not heavily penalize large node size imbalances.
- B: True. Relying on misclassification does not give insight into the node composition after the split and doesn't distinguish as well between different options.
- C: True. We can only get zero information gain if teh child sets probabilities are the exact same.
- D: True. Misclassification is not strictly concave; it looks like an updside down absolute value graph, so there are quite a few splits along the same line.

(b) [4 pts] Select the true statements about **decision trees**.

○ A: Decision trees are grown from the bottom up, meaning that we start at the leaves and repeatedly fuse them until we reach the root node.

○ B: Pruning improves both training and validation accuracy.

● C: They can guarantee 100% training accuracy on any training set where no two identical points have different labels.

○ D: Decision trees are used for classification only, not regression.

- A: False. We build them top-down, starting with the entire training set and dividing it at each step.
- B: False. Pruning improves validation accuracy but decreases training accuracy.
- C: True, if you don't impose any max depth condition or use pruning.
- D: Just false.

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

○ A: Restricting the trees to be shallow decreases the bias.

● B: A random forest decreases the variance (compared to a single decision tree).

● C: We restrict which features can be used for splitting in each treenode to decrease the correlation be-

tween the trees' predictions.

● D: In a typical tree in the forest, some training points are not used.

Choice A is the opposite of what is true. Choice B is the primary purpose of random forests. Choice C is true since taking random features helps trees be more independent of one another. Choice D is true since we are using bootstrapping where we sample with replacement.

(d) [4 pts] Select the valid reasons why you might choose **not** to use **bagging** on a particular training set.

○ A: Your model has low bias and high variance.

● B: Your model has high bias and low variance.

● C: You want your model to be interpretable.

○ D: You are solving a regression problem.

One of the main advantages of bagging is to reduce variance, but it does not improve bias, so A is incorrect and B is correct. Bagging also reduces the interpretability of your model, so choice C is correct. Finally, you can use bagging even if you are doing regression, so D is incorrect.

(e) [4 pts] Select the true statements about ***k*-nearest neighbor classification**.

● A: As k increases, the decision boundary tends to look smoother.

○ B: In 3-class classification, choosing k to be prime ensures there will never be a tie between classes.

● C: Even the slow, $\Theta(nd)$-time exhaustive nearest neighbor algorithm is often used in practice.

● D: As the number of neighbors grows as $k \to \infty$ and the number of training points grows even faster, so $n/k \to \infty$, the k-nearest neighbor error rate converges to the Bayes risk.

- A: True. When we consider more neighbors, the labels fluctuate less for small changes in features.
- B: False. For example, you could have $\lfloor \frac{k}{2} \rfloor$ of class A, $\lfloor \frac{k}{2} \rfloor$ of class B, and one of class C.
- C: True. In many cases, it's the first algorithm you should try. Then optimize if it's not fast enough or good enough.
- D: True. This is Fix & Hodges' Theorem from 1951.

(f) [4 pts] Select the valid reasons that **ReLUs may be preferred over sigmoids (logistic functions)** as activation functions for the **hidden** layers of a neural network.

● A: The forward and backward passes are computationally cheaper with ReLUs than with sigmoids.

○ B: The cost function of a ReLU-based neural network, trained with the squared-error loss, will be convex because the ramp (ReLU) function is convex.

● C: ReLUs are less vulnerable to the vanishing gradient problem than sigmoids.

○ D: The cost function of a ReLU-based neural network is smooth with a gradient defined everywhere in weight space, whereas the cost function of a sigmoid-based neural network is not.

- A: True. The forward and backward passes of ReLU only require a single thresholding operation, but you need to compute an exponential for a sigmoid.
- B: False. The composition of two or more convex functions need not be convex. ReLU network rarely have convex cost functions.
- C: True. Empirically, ReLUs don't get stuck as often as sigmoids, and this is one of the main reasons why ReLUs have largely replaced sigmoids as hidden unit activation functions. (See lecture 18).
- D: False. It's exactly the opposite of that.

(g) [4 pts] Which steps are customarily (usually) part of training a neural network's weights with **backpropagation**?

○ A: Computing the partial derivatives of each weight with respect to each weight in the previous layer.

● B: Computing the partial derivatives of a cost function or loss function with respect to each weight.

● C: Computing the partial derivatives of a cost function or loss function with respect to each hidden unit value.

○ D: Computing the partial derivatives of a cost function or loss function with respect to each input feature.

- A: False. Weights do not depend on each other.
- B: True. That's what we need for gradient descent.
- C: True. These values are needed as intermediate results to obtain the gradients with respect to the weights.
- D: False. We can't change the training points, so these derivatives are not useful.

(h) [4 pts] Select the true statements about **principal components analysis** (PCA).

● A: The first principal component is always orthogonal to the second principal component.

○ B: We use the subset of principal components associated with the smallest eigenvalues of the sample covariance matrix.

○ C: PCA requires the training points to have labels.

● D: If the sample covariance matrix has an eigenvalue of zero, the corresponding eigenvector (principal component) is orthogonal to a hyperplane that passes through all the training points.

- A: True. All the principal components are pairwise orthogonal.
- B: False. The largest eigenvalues.
- C: False. PCA is unsupervised learning.
- D: True. It indicates a principal component direction with zero variance.

(i) [4 pts] Select the true statements about **the objective of principal components analysis** (PCA).

● A: We want to find a subspace that minimizes the mean of the squared projection distances.

○ B: We want to find a subspace that maximizes the mean of the squared projection distances.

○ C: We want to find a subspace that minimizes the sample variance of the projected sample points.

● D: We want to find a subspace that maximizes the sample variance of the projected sample points.

We want to minimize mean squared projection distance and maximize sample variance of data, directly from lecture notes.

(j) [4 pts] Suppose that $\dot{X} \in \mathbb{R}^{n \times d}$ is a **centered** design matrix. Recall that its singular value decomposition (SVD) is written $\dot{X} = UDV^\top$. Select the true statements about **principal components analysis** (PCA) **and the SVD**.

● A: The principal components are columns of V.

● B: When k is much less than d, we can find k principal components faster by computing a partial SVD than we can by computing the eigenvectors of the sample covariance matrix.

○ C: The principal coordinates for sample point $\dot{X}_i$ appear in row i of V.

○ D: The diagonal entries of D are the eigenvalues that correspond to the principal components.

The principal components are the columns of V, since v_i is an eigenvector of $\dot{X}^\top \dot{X} = VD^2V^\top$, and we compute the eigenvectors of $\dot{X}^\top \dot{X}$ in the original PCA algorithm.
SVD is faster, it takes $O(ndk)$ time vs. $O(nd^2)$ to compute $\dot{X}^\top \dot{X}$ alone in the original PCA algorithm.
The principal coordinates of the data are given by the rows of UD, since $\dot{X}V$ represents the principal coordinates, and $\dot{X}V = UDV^\top V = UD$.
The diagonal entries of D are the singular values; they must be squared to get the eigenvalues.

(k) [4 pts] While training your spam classifier in Homework 1, you observe that your soft-margin support vector machine's **training accuracy is only 58% and the validation accuracy is 54%**. Which interventions have a significant chance of making it possible to obtain **test accuracy** on Kaggle of 80% or higher?

○ A: Try smaller values of the hyperparameter C.

○ B: Delete features with poor predictive power.

○ C: Obtain more training data.

● D: None of the above.

Poor training accuracy is a problem with underfitting/bias, which cannot be fixed by reducing C, deleting features, or adding training data. Even if these measures decrease the variance, they won't improve the training accuracy, and the validation/test accuracy is (almost) never better than the training accuracy.

(l) [4 pts] Suppose we have two classes, A and B, and we wish to train a classifier to recognize them.

○ A: If we want to use decision theory (risk minimization) to create a generative model, the two classes must each have a normal distribution.

○ B: If the classes are not linearly separable and we want to use a soft-margin support vector machine, we must create added features or the training algorithm will fail to output a classifier.

● C: If we have only one training point for class A and only one training point (at a different location) for class B, the centroid method classifier obtains 100% training accuracy.

● D: If all the training points are at distinct locations, the 1-nearest neighbor algorithm obtains 100% training accuracy.

A: No, decision theory works with many different distributions. For example, see Question 4 on this exam. B: No, a soft-margin SVM can be directly trained on points that aren't linearly separable. (By contrast, a hard-margin SVM will fail.) C: Yes; the centroid classifier will always separate the two points. D: Yes.

(m) [4 pts] Consider a least squares problem where we have a design matrix $X \in \mathbb{R}^{n \times d}$ (we don't use a bias term) and a vector of labels $y \in \mathbb{R}^n$. We wish to learn a weight vector $w \in \mathbb{R}^d$ that minimizes $\text{RSS}(w) = \|Xw - y\|^2$. If X is invertible, the least-squares solution is $\hat{w} = (X^\top X)^{-1} X^\top y$. Select the true statements.

● A: $n \geq d$ is a necessary condition for $X^\top X$ to be invertible.

○ B: $n \geq d$ is a sufficient condition for $X^\top X$ to be invertible.

● C: If X is invertible, then there is a solution w^* such that $\text{RSS}(w^*) = 0$.

● D: If we instead use ridge regression with a positive λ, then ridge regression always has a unique solution regardless of the invertibility of $X^\top X$.

- Yes, if $n < d$, then $r(X^\top X) = r(X) <= n < d$, thus $X^\top X$ cannot be full rank.
- No, X could still be singular. Consider a very long 0 matrix.
- Yes, the solution is $w^* = X^{-1} y$, thus $Xw^* = Iy = y$.
- Yes, by the eigenshift property applied on $(X^\top X + \lambda I)$ and the fact that $X^\top X$ is PSD.

(n) [4 pts] Select the true statements about **AdaBoost**.

○ A: AdaBoost is an ensemble method designed expressly to reduce the variance of decision trees.

○ B: After enough iterations, AdaBoost can always obtain 100% training accuracy, regardless of what classifier it uses.

● C: AdaBoost with decision trees typically uses different criteria/hyperparameters to build the trees than random forests do.

○ D: AdaBoost adjusts the weights of misclassified training points to increase their information gain.

A. False; that's random forests. AdaBoost is designed to reduce the bias, but it does not reliably reduce the variance. B. False. In Discussion 9 we saw that an ensemble of stumps can't compute XOR. More trivially, consider a classifier that always returns 1, regardless of the input. C. True. AdaBoost is usually used with short trees, whereas random forests usually use deep and often pure trees, because AdaBoost is a method for reducing the bias, whereas random forests are a technique for reducing the variance. D. False. AdaBoost has nothing to do with information gain.

(o) [4 pts] Select the true statements about **Lasso**.

○ A: The Lasso regression coefficients have a closed-form solution that can be computed by solving a linear system of equations.

● B: Lasso becomes least-squares linear regression when $\lambda = 0$.

○ C: The isocontours of the ℓ_1-regularization term are hypercubes (high-dimensional cubes).

● D: The main motivation of Lasso is that it tends to select weights of zero for weakly predictive features, which may reduce overfitting and improve interpretability.

A. False, Lasso has no closed-form solution.
B. True. C. False, they're cross-polytopes, which are not the same. (But the isocontours of ℓ_∞-regularization are hypercubes.)
D. Yes, that's the motivation!

(p) [4 pts] In a **convolutional neural network**, which of the following changes to network parameters will **decrease** the size (number of units) of a **hidden layer** of the network?

● A: Increasing the size of the filters (masks) in a convolutional layer. (Assume we do not use any padding nor "periodic boundaries.")

○ B: Increasing the number of filters (masks) in a convolutional layer.

● C: Changing a pooling layer to use a 4×4 sliding window with stride 4 instead of a 2×2 sliding window with stride 2.

○ D: Decreasing the size of the mini-batch used for stochastic gradient descent.

A is correct because larger filters cut more off the edges of an image, yielding a smaller hidden layer. B is incorrect because more filters require more hidden units. C is correct because the larger sliding window and stride means fewer hidden units after pooling. D is incorrect because it's comic relief.

(q) [4 pts] Consider a matrix $X \in \mathbb{R}^{n \times d}$ with a singular value decomposition (SVD) $X = UDV^\top$. Which of the following formulae are **equal to the Moore–Penrose pseudoinverse** X^+? (Check a box only if it works for every X.)

○ A: $VD^{-1}U^\top$

● B: $VD^+U^\top$

● C: $VD^+DD^+U^\top$

● D: $VV^\top VD^+U^\top UU^\top$

B is a good working definition of Moore–Penrose pseudoinverse. A is flawed because D might not be invertible. D is equivalent to B because $V^\top V = I$ and $U^\top U = I$. C is equivalent to B because $D^+DD^+ = D^+$.

(r) [4 pts] Below are four examples where we have written a line of pseudocode from a primal, unfeaturized learning algorithm followed by the equivalent line in a dualized, featurized, kernelized version of the same learning algorithm. The primal algorithm computes the weight vector $w \in \mathbb{R}^d$, and the kernelized algorithm computes the dual weight vector $a \in \mathbb{R}^n$. The inputs are the design matrix $X \in \mathbb{R}^{n \times d}$ and a vector of labels $y \in \mathbb{R}^n$. The kernel matrix is $K \in \mathbb{R}^{n \times n}$ and the kernel function is $k(\cdot, \cdot)$. Select the cases where **the kernelized version correctly duplicates the effect of the primal version**.

● A: primal: $w \leftarrow w + X^\top v$
kernelized: $a \leftarrow a + v$

● B: primal: $h(z) \leftarrow s(w^\top z)$
kernelized: $h(z) \leftarrow s\left(\sum_{i=1}^{n} a_i \, k(X_i, z)\right)$

● C: primal: $w \leftarrow w + \epsilon X^\top (y - s(Xw))$
kernelized: $a \leftarrow a + \epsilon (y - s(Ka))$

○ D: primal: $\tau \leftarrow \|w\|^2$
kernelized: $\tau \leftarrow a^\top K^2 a$

B and C are taken directly from the primal/featurized and dual/kernelized logistic regression algorithms presented in lecture. A is a simpler transformation similar to C. But D is wrong because Ka produces Xw, not w.

Extra space: if you need extra space for your answer to a written problem on pages 6–12, you may write here. **Be sure to write "see page 5" under the unfinished answer!**

Q2. [18 pts] Decision Tree Construction

You're investigating the habits of UC Berkeley students. You collect data about whether students climb, drink coffee, and are night owls. You want to use those habits and a decision tree to predict if students are taking CS 189. Each training student is labeled with one of two classes: 189 or MissingOut (not taking 189). Here is the training data you've gathered. (Note that the student numbers are not features and you can't split on them.)

	Climbs	Drinks Coffee	Night Owl	Taking CS 189 (label)
Student 1	Yes	Yes	No	189
Student 2	No	No	Yes	189
Student 3	Yes	No	Yes	MissingOut
Student 4	No	No	No	MissingOut
Student 5	No	No	Yes	189
Student 6	No	Yes	Yes	189
Student 7	Yes	No	Yes	MissingOut
Student 8	Yes	Yes	No	189

(a) [4 pts] **What is the entropy H at the root** of the tree? Your expression can contain logarithms and fractions.

There are 8 people, 5 of whom are taking 189 and 3 of whom are losers.

$$H = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8} \approx 0.9544.$$

(The decimal approximation is not required.)

(b) [6 pts] **Which feature should you split on at the root to maximize the information gain? Write an expression for the information gain of the best split.** Your expression can contain logarithms and fractions, but **simplify** the fractions as much as possible. Show your work.

The best split is on whether or not a person drinks coffee. This split gives us a pure leaf node with 3 people in it.

Before calculating any entropies, we consider what happens if we choose to split on each feature. If we split on coffee, the Yes subtree is pure with 3 189 students, and the No subtree has a 2-3 split. If we split on being a night owl, the No subtree has a 2-1 split and the Yes subtree has a 3-2 split. If we split on climbing, the Yes subtree has a 2-1 split and the No subtree has a 3-2 split. All three cases have one child with a 3-2 split, but only coffee can deliver a pure leaf node (zero entropy) right away.

Recall that the entropy after a split is $H_{\text{after}} = \dfrac{|S_l|H(S_l) + |S_r|H(S_r)}{|S_l| + |S_r|}$. We split on coffee, so the information gain is

$$\begin{aligned} H_{\text{before}} - H_{\text{after}} &= \left(-\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8}\right) - \left(\frac{3}{8}(-1\log_2 1) + \frac{5}{8}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right)\right) \\ &= -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8} + \frac{1}{4}\log_2\frac{2}{5} + \frac{3}{8}\log_2\frac{3}{5} \approx 0.348. \end{aligned}$$

Students are not expected to show the information gain for other splits, but we include it below (it's the same for climbing and night owls) as proof that coffee is the best split.

$$H_{\text{before}} - H_{\text{after}} = \left(-\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8}\right) - \left(\frac{3}{8}\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) + \frac{5}{8}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right)\right) \approx 0.003.$$

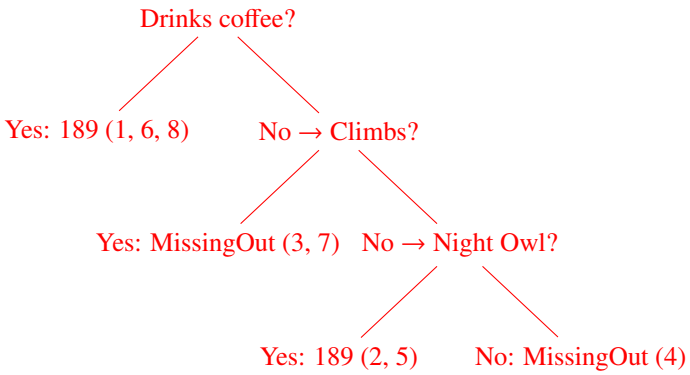
(c) [4 pts] **Draw the complete decision tree** that maximizes the information gain at each split. In each leaf, write

- what class is assigned (189 or MissingOut) and
- which training points reach that leaf (numbered 1–8).

For each internal treenode, write

- the splitting feature and
- which child is "Yes" and which child is "No."

Split until all the leaves are pure. Do not write any entropies or information gains.



(d) [4 pts] Suppose that we are building a random forest with this training set, but we do **not** use bagging and every tree receives the full training set; the only randomization we use is randomized selection of features at each treenode (in the usual way taught in class). After training the forest, you discover that the first tree in your ensemble classifies a test point in which all three features are "Yes" as MissingOut. By contrast, your tree in part (c) (if it's correct) predicts 189.

Explain what must have happened during training that led to the tree in the random forest making a different prediction than your tree from part (c).

When the root was split, the feature "Drinks Coffee" was not available because of the randomized choice of features to split on.

Q3. [20 pts] Backpropagation

Consider a fully-connected neural network with d units in the input layer, one hidden layer of k units with ReLU activations, and a single output unit with a sigmoid activation. (Hence there are two layers of weight connections.) We are using the network for two-class classification; we predict in-class for $z > 0.5$ or out-of-class for $z \leq 0.5$. Each training point $X_i \in \mathbb{R}^d$ is accompanied by a ground truth label y_i equal to either 1 (in class) or 0 (not in class). We train our model with stochastic gradient descent (one training point at a time; no batches or mini-batches) and the logistic loss $L(z, y)$ (for prediction z and label y).

Let $V \in \mathbb{R}^{k \times d}$ be the weight matrix associated with the first layer of connections. As there is only one output unit, we represent the weights in the second layer of connections as a vector $w \in \mathbb{R}^k$. For simplicity, we do not use bias terms. Here is pseudocode for the forward pass with training point $x \in \mathbb{R}^d$, label $y \in \{0, 1\}$, hidden layer $h \in \mathbb{R}^k$, and output $z \in \mathbb{R}$.

h	$\leftarrow r(Vx)$	hidden units; r is the ramp function (a ReLU) applied element-wise to the vector Vx
z	$\leftarrow s(w^\top h)$	z is the output unit; $s(\gamma) = 1/(1 + e^{-\gamma})$ is the sigmoid/logistic function
L	$\leftarrow -y \ln z - (1 - y) \ln(1 - z)$	compute the logistic loss

Work out the equations for backpropagation in this neural network. The derivatives and gradients we ask you to derive may include x, h, z, w, V, y, and/or components of those vectors and matrix. Your final formulae may **not** include s nor s'; express derivatives related to the sigmoid function in terms of x, h, z, w, V, and/or y. All vectors and gradients are column vectors.

(a) [2 pts] **Write the derivative r' of the ramp function** $r(\gamma)$ (where γ is a scalar real value). To make backpropagation work, choose a reasonable value for r' at $\gamma = 0$ (rather than the actual derivative).

$$r'(\gamma) = \begin{cases} 1, & \gamma \geq 0, \\ 0, & \gamma < 0. \end{cases}$$

At $\gamma = 0$, any value in $[0, 1]$ will do.

(b) [3 pts] Derive $\dfrac{\partial L}{\partial z}$.

$$\frac{\partial L}{\partial z} = -\frac{y}{z} + \frac{1 - y}{1 - z}.$$

(c) [3 pts] **Derive $\dfrac{\partial L}{\partial h_i}$ as a function of $\dfrac{\partial L}{\partial z}$**, as well as x, h, z, w, and/or V. (Do not substitute your expression from part (b) yet; you'll do that in part (d).) **Show your work.** We want to see s' in an intermediate step, but **not** in your final answer.

$$\frac{\partial L}{\partial h_i} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial h_i} = \frac{\partial L}{\partial z} s'(w^\top h) w_i = \frac{\partial L}{\partial z} s(w^\top h)(1 - s(w^\top h)) w_i = \frac{\partial L}{\partial z} z(1 - z) w_i.$$

(d) [3 pts] Substitute your expression for $\dfrac{\partial L}{\partial z}$ from part (b) into your expression for $\dfrac{\partial L}{\partial h_i}$ from part (c) and **simplify as much as possible.** Show your work. Hint: there should be no need for a fraction in your final answer.

$$\frac{\partial L}{\partial h_i} = \left(-\frac{y}{z} + \frac{1 - y}{1 - z}\right) z(1 - z) w_i = (-y(1 - z) + (1 - y)z) w_i = (z - y) w_i.$$

(e) [2 pts] **Derive $\nabla_w L$ as a function of $\frac{\partial L}{\partial z}$.** Then **substitute in your expression for $\frac{\partial L}{\partial z}$ from part (b) and simplify** as much as possible. Again, do not have s' in your final answer. (There's no need to show your work this time.)

$$\nabla_w L = \frac{\partial L}{\partial z} \nabla_w z = \frac{\partial L}{\partial z} z(1-z)h = (z-y)h.$$

(f) [2 pts] Let $V_i^\top$ denote the ith row of V (so V_i is a column vector). **Derive $\nabla_{V_i} L$ as a function of $\frac{\partial L}{\partial h_i}$**, as well as x, h, z, w, and/or V. Your final answer may have r' in it.

$$\nabla_{V_i} L = \frac{\partial L}{\partial h_i} \nabla_{V_i} h_i = \frac{\partial L}{\partial h_i} r'(V_i^\top x) x.$$

(g) [3 pts] **Write pseudocode to perform a pass of backpropagation** (following the forward pass we wrote above) and **a step of stochastic gradient descent** with learning rate (step size) ϵ, given a single training point x and a label y. Do **not** directly compute $\frac{\partial L}{\partial z}$, as we want to take advantage of the simplifications in parts (d) and (e).

$$\begin{aligned}
\nabla_w L &\leftarrow (z-y)h \\
\frac{\partial L}{\partial h_i} &\leftarrow (z-y)w_i \text{ for all } i \in [1,k] \\
\nabla_{V_i} L &\leftarrow \frac{\partial L}{\partial h_i} r'(V_i^\top x) x \text{ for all } i \in [1,k] \\
w &\leftarrow w - \epsilon \nabla_w L \\
V_i &\leftarrow V_i - \epsilon \nabla_{V_i} L \text{ for all } i \in [1,k]
\end{aligned}$$

Grading note: make sure w is not updated before $\partial L / \partial h_i$ is computed!

It is also acceptable to write $w \leftarrow w - \epsilon(z-y)h$ and $V_i \leftarrow V_i - \epsilon \frac{\partial L}{\partial h_i} r'(V_i^\top x) x$.

(h) [2 pts] Suppose we run **batch gradient descent** to convergence with this neural network, so V and w are at **a local minimum of the cost function**, which is the mean of the training points' loss functions. After the network is trained, let $H_i \in \mathbb{R}^k$ be the value of the vector of hidden units when the training point X_i is presented at the inputs of the neural network. Suppose we run logistic regression with no bias term (no α) on the points $H_1, H_2, \ldots, H_n$ with the labels $y_1, y_2, \ldots, y_n$, obtaining a weight vector w_{logreg}. **What is the relationship between the final neural network weights V and w and the final logistic regression weights w_{logreg}?**

$w = w_{\text{logreg}}$. (The last stage of the neural network is doing logistic regression on the hidden unit values.)

Q4. [14 pts] The Bayes Classifier

We have a distribution $\mathcal{D}$ of real numbers with class labels. When we draw a random point-and-label $(X, Y) \sim \mathcal{D}$, There is a 40% chance that the label Y is class A, in which case X is a random real number drawn from a uniform distribution on the interval $[-1, 1]$. There is a 60% chance that the label Y is class B, in which case X is a random real number drawn from a distribution on $[-1, 1]$ with the probability density function (PDF) $f_{X|Y=\mathrm{B}}(x) = 1 - |x|$. We will use the 0-1 loss function.

(a) [6 pts] **What is the Bayes optimal decision boundary? Show your work.** (Hint: it might help you to draw a plot.)

For points in class A, the PDF is $f_{X|Y=\mathrm{A}}(x) = 0.5$ (on the domain $[-1, 1]$) and the posterior probability is

$$P(Y = \mathrm{A}|X = x) = \frac{f_{X|Y=\mathrm{A}}(x)P(Y = \mathrm{A})}{f(x)} = \frac{0.2}{f(x)}.$$

For points in class B, the posterior probability is

$$P(Y = \mathrm{B}|X = x) = \frac{f_{X|Y=\mathrm{B}}(x)P(Y = \mathrm{B})}{f(x)} = \frac{0.6(1 - |x|)}{f(x)}.$$

The Bayes optimal decision boundary consists of the two points where $P(Y = \mathrm{A}|X = x) = P(Y = \mathrm{B}|X = x)$, or equivalently $0.2 = 0.6(1 - |x|)$, so the two points are $x = \pm 2/3$.

(b) [4 pts] **What is the Bayes decision rule $r^*(x)$?** (Over the domain $[-1, 1]$; don't worry about points outside that range.)

The Bayes decision rule $r^*(x)$ returns class B for $x \in (-2/3, 2/3)$ and class A for $x < -2/3$ or $x > 2/3$. (If x is one of the two points on the decision boundary, r^* can return either A or B; we don't really care which.)

(c) [4 pts] **What is the Bayes risk $R(r^*)$? Show your work and express your answer as an exact fraction.** (If you drew a plot, it should help you here too.)

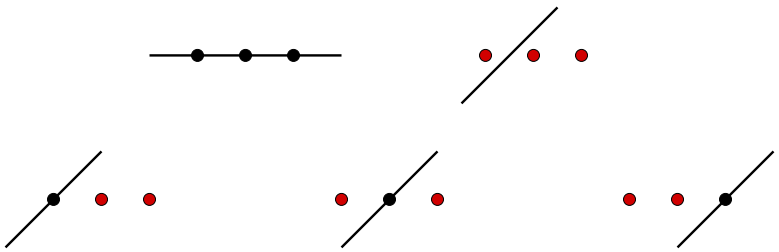
$$\begin{aligned}
R(r^*) &= \int_{-1}^{1} \min_y f_{X|Y=y}(x)\, P(Y = y)\, \mathrm{d}x \\
&= \int_{-1}^{-2/3} f_{X|Y=\mathrm{B}}(x)\, P(Y = \mathrm{B})\, \mathrm{d}x + \int_{-2/3}^{2/3} f_{X|Y=\mathrm{A}}(x)\, P(Y = \mathrm{A})\, \mathrm{d}x + \int_{2/3}^{1} f_{X|Y=\mathrm{B}}(x)\, P(Y = \mathrm{B})\, \mathrm{d}x \\
&= \int_{-1}^{-2/3} 0.6(1 + x)\, \mathrm{d}x + \int_{-2/3}^{2/3} 0.2\, \mathrm{d}x + \int_{2/3}^{1} 0.6(1 - x)\, \mathrm{d}x \\
&= 0.1/3 + 0.8/3 + 0.1/3 \\
&= 1/3.
\end{aligned}$$

Q5. [12 pts] CS 189/289A Art School

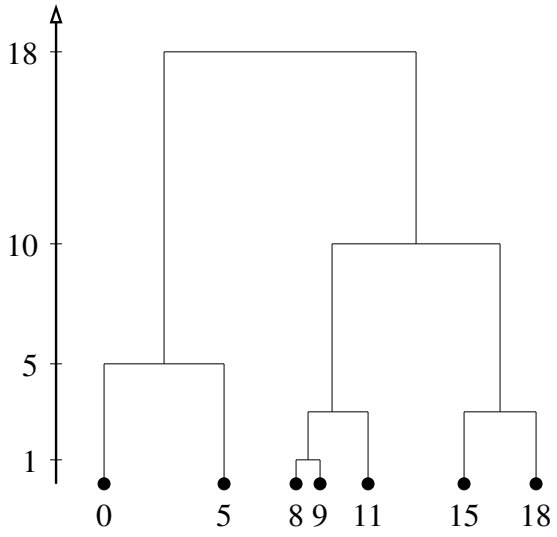
(a) [6 pts] Consider a range space $(\mathbb{R}^2, H)$ where H is the set of all lines in the plane. (Not halfplanes; lines! Not line segments; infinite lines! A point $x \in \mathbb{R}^2$ is in class C if x lies precisely on the line; otherwise it's not in class C. Yes, it's not a very useful classifier.)

Consider a set X of **three collinear** (but distinct) points in the plane. (Collinear means they all lie on a common line.) How many dichotomies of X can you obtain with the hypothesis class H? **Draw all of the dichotomies:** for each dichotomy, draw the three points and a hypothesis that generates the dichotomy.

There are five.



(b) [6 pts] Let's do hierarchical clustering in one dimension. Specifically, we will do agglomerative clustering with **complete linkage** (which you might think of as the "max" linkage) on the numbers listed below. **Draw the dendrogram** for the points $\{0, 5, 8, 9, 11, 15, 18\}$. The distance function for two individual points p and q is $d(p, q) = |p - q|$. Make sure you draw the horizontal bars at the correct heights! We have included two copies of the figure in case your first drawing is wrong and you need to start over, but you only need to draw one.



Q6. [14 pts] Random Projection

Recall the method of random projection, where we project every sample point $x \in \mathbb{R}^d$ onto a randomly chosen k-dimensional subspace $S \subset \mathbb{R}^d$, where $k < d$. Let's consider a closely related method for dimensionality reduction that also approximately preserves distances between points (with high probability).

(a) [4 pts] Recall from our discussion of principal components analysis (PCA) that, if two conditions hold, we can orthogonally project a point $x \in \mathbb{R}^d$ onto the subspace spanned by the vectors $v_1, \ldots, v_k$ with the formula $\tilde{x} = \sum_{i=1}^{k} (x \cdot v_i)\, v_i$. **What two conditions** must $v_1, \ldots, v_k$ satisfy for this formula to be correct?

The vectors $v_1, \ldots, v_k$ must have unit length and be pairwise orthogonal to each other.

(b) [5 pts] Consider a random matrix $G \in \mathbb{R}^{k \times d}$, with $k < d$. Every component of G is a random real number $G_{ij} \sim \mathcal{N}(0, 1/d)$; that is, every component is drawn from a univariate normal distribution with mean zero and variance $1/d$. These components are all independent of each other. Consider a (**not** random) vector $x \in \mathbb{R}^d$. **Show that** the expected squared ℓ_2-norm of Gx is

$$E\left[\|Gx\|^2\right] = \frac{k}{d}\|x\|^2.$$

$$\|Gx\|^2 = \sum_{i=1}^{k} \sum_{j=1}^{d} \left(G_{ij} x_j\right)^2.$$

As x is not random,

$$\mathbb{E}[\|Gx\|^2] = \sum_{i=1}^{k} \sum_{j=1}^{d} \mathbb{E}[G_{ij}^2] x_j^2.$$

$\mathbb{E}[G_{ij}^2]$ is the variance of G_{ij}, which is $1/d$. So

$$\mathbb{E}[\|Gx\|^2] = \sum_{i=1}^{k} \sum_{j=1}^{d} \frac{1}{d} x_j^2 = \frac{k}{d}\|x\|^2.$$

(c) [5 pts] The result in part (b) suggests that if we choose a random G as described in part (b) and replace the sample points $X_1, X_2, \ldots, X_n \in \mathbb{R}^d$ with the k-dimensional points

$$\tilde{X}_1 = \sqrt{\frac{d}{k}} G X_1, \quad \tilde{X}_2 = \sqrt{\frac{d}{k}} G X_2, \quad \ldots, \quad \tilde{X}_n = \sqrt{\frac{d}{k}} G X_n,$$

the distance between two points might not "change much"; that is, $\|\tilde{X}_i - \tilde{X}_j\|$ is an approximation of $\|X_i - X_j\|$. This is similar to computing an orthogonal projection onto a randomly chosen k-dimensional subspace, but not quite the same.

In the limit as $d \to \infty$ (with k fixed), **why does this method become functionally the same as the method of random projection?** Hint: what two counterintuitive properties of vectors in high-dimensional spaces are relevant, and which vectors should we apply them to? (A written answer suffices; no math is necessary.)

In high-dimensional spaces, random vectors from a normal distribution are approximately the same length and approximately orthogonal to each other. We apply these properties to the rows of G.

In the limit as $d \to \infty$, the length of every row of G approaches 1 and the dot product of every pair of distinct rows of G approaches zero. Hence, in the limit Gx approaches an orthogonal projection of x onto the subspace spanned by G's rows.