CS 189/289A Introduction to Machine Learning
Spring 2023 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 DSP 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 the back of the page.
- 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

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**: the set of all correct answers must be checked.

(a) [4 pts] Which of the following are *typical* benefits that might motivate you to preprocess data with principal components analysis (PCA) before training a classifier?

○ A: PCA tends to reduce the bias of your classification algorithm.

● B: PCA can be used to avoid overfitting.

● C: PCA tends to reduce the variance of your classification algorithm.

○ D: PCA can be used to avoid underfitting.

PCA removes irrelevant dimensions, which helps reduce variance by limiting the effects of noisy or unreliable data. This then helps to avoid overfitting. In general, PCA does not reduce the bias, and often increases it. It is unlikely to improve underfitting.

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

○ A: PCA is a clustering algorithm.

● B: PCA produces features (principal coordinates) that are linear combinations of the input features.

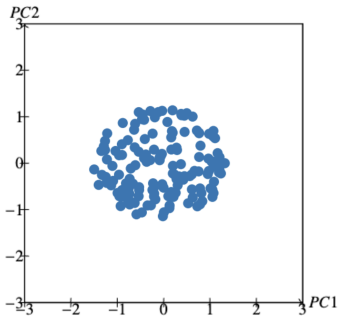
● C: The principal components are chosen to maximize the variance in the projected data.

○ D: The principal coordinates are the eigenvalues of the sample covariance matrix.

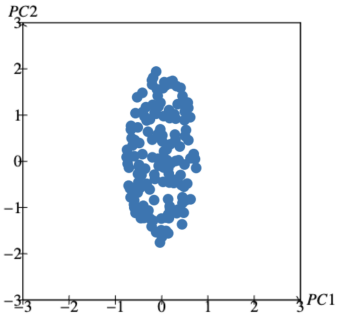
A: PCA is not a clustering algorithm, though PCA can help with clustering algorithms. B: True. C: True. The first principal component explains the most variance in the data, the second explains the second most, and so on. D: False; the principal coordinates are the inner product of a sample point with each eigenvector.

(c) [4 pts] In each plot below, the data is projected onto two (unit-length) principal component vectors. We say that a plot is "valid" if the x-coordinate (written as "PC1") would be the first chosen principal coordinate and the y-coordinate (written as "PC2") would be the second chosen principal coordinate. Which plots are valid?

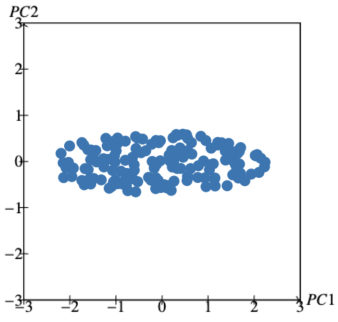
● A:



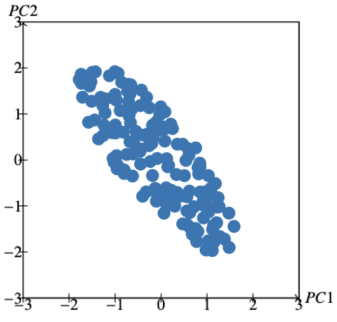
○ B:



● C:



○ D:



(d) [4 pts] Which of the following are *typical* benefits of ensemble learning in its basic form (that is, not AdaBoost and not with randomized decision boundaries), with all weak learners having the same learning algorithm and an equal vote?

○ A: Ensemble learning tends to reduce the bias of your classification algorithm.

● B: Ensemble learning tends to reduce the variance of your classification algorithm.

● C: Ensemble learning can be used to avoid overfitting.

○ D: Ensemble learning can be used to avoid underfitting.

In ensemble learning, increasing the number of classifiers reduces the variance of our model but generally has little effect on the bias. Therefore, basic ensembling can be used to avoid overfitting but would generally not be used to avoid underfitting. (By contrast, AdaBoost reduces bias.)

(e) [4 pts] Recall that in certain cases, Newton's method can converge to the global optimum of an objective function in just one step. For which of the following objective functions and methods will Newton's method always converge in just one step? (Assume $\lambda > 0$ where regularization is used.)

○ A: Dualized, kernelized logistic regression where the cost function is the mean of the logistic losses

● B: Ridge regression with an ℓ_2-regularized mean squared error

○ C: A neural network with ReLU activation functions on the hidden units and an ℓ_2-regularized mean squared error

● D: Dualized, kernelized ridge regression with an ℓ_2-regularized mean squared error

Newton's method converges in one step for quadratic, strictly convex cost functions. Ridge regression has a quadratic cost function, and ℓ_2-regularization with $\lambda > 0$ makes it strictly convex. This remains true after you dualize and kernelize the algorithm. But the neural network cost function is not even close to quadratic.

(f) [4 pts] We are fitting a linear function to data in which some features are known to be much noisier than others. We account for this by applying an asymmetric penalty in ridge regression: in the normal equations, we replace the identity matrix with a different diagonal matrix. Each entry on the diagonal is a hyperparameter. Although there are many hyperparameters, suppose that we magically find the best hyperparameter values for our validation set. How is the result likely to differ from the result of standard ridge regression?

● A: Lower validation error.

● B: It is equivalent to placing an anisotropic Gaussian prior probability on the regression weights, then finding the weights that maximize the likelihood.

○ C: The number of weights equal to zero tends to be greater.

○ D: The difference is usually small, as the hyperparameters will tend to have almost the same value.

Adding a more flexible space of hyperparameters means that the optimal choice of hyperparameters will give the same or lower validation error than the standard method. The fact that some features are noisier than others makes it likely to be lower. The modified method can be derived from MLE by placing an anisotropic Gaussian prior on the regression weights, whose covariance matrix is the inverse of the diagonal matrix used in our ridge regression normal equations (times a constant).

Sparsity is not likely to be affected by this change, as ridge regression rarely sets a weight to exactly zero. The hyperparameters are not likely to have similar values, because the purpose of regularization is to prevent weights from getting crazy big to overfit noisy features in training data, if those big weights don't do well on the validation data. The modified method has extra latitude to selectively restrain weights that want to be big but have little predictive power, without sacrificing the predictive power of less noisy weights.

(g) [4 pts] We are given n training points having d features each. Select the true statements about applying a k-nearest neighbor algorithm to a test point.

● A: It is possible to implement the algorithm to use the ℓ_1 metric as your distance function.

○ B: The k-d tree k-nearest neighbor algorithm is fast because it computes the distance between the test point and a training point for only k training points.

● C: There is a k-nearest neighbor algorithm that classifies a test point in at most $O(nd + n \log k)$ time, even if k is much larger than d.

○ D: The k-nearest neighbor algorithm can be used for classification, but not regression.

(h) [4 pts] Select the true statements about the singular value decomposition (SVD) and the eigendecomposition.

○ A: The SVD applies only to square matrices.

● B: The eigendecomposition applies only to square matrices.

● C: The right singular vectors of a matrix $X \in \mathbb{R}^{n \times d}$ are eigenvectors of $X^\top X$.

● D: Consider a non-square matrix $X \in \mathbb{R}^{n \times d}$ and the vector $w \in \mathbb{R}^d \setminus \{\mathbf{0}\}$ that maximizes the Rayleigh quotient $(w^\top X^\top X w)/(w^\top w)$. The singular values of X are no greater than the (positive) square root of the maximum Rayleigh quotient.

The SVD applies to any matrix, but the eigendecomposition exists only for square matrices. C is true because the SVD of a symmetric matrix is almost the same as an eigendecomposition, except that it might be necessary to flip the signs of some singular values and their corresponding right (or left, you choose) singular vectors. (If an eigenvalue is negative, the corresponding singular value is positive.) But when you flip the sign of an eigenvector, it's still an eigenvector. D is true because the maximum possible value of the Rayleigh quotient is the greatest eigenvalue of $X^\top X$, obtained when w is an eigenvector with that eigenvalue. This eigenvalue is the square of the greatest singular value of X (whose right singular vector is the maximizing unit w).

(i) [4 pts] Select the true statements about AdaBoost for two-class classification.

○ A: We can train all T weak learners simultaneously in parallel.

● B: After a weak learner is trained, the weights associated with the training points it misclassifies are increased.

○ C: The coefficient β_T assigned to weak learner G_T is 0 if the weighted error rate err_T of G_T is 1.

○ D: AdaBoost makes no progress if it trains a weak learner only to discover that its weighted error rate is substantially greater than 0.5.

A is false because the weak learners must be trained sequentially, as the weight of each training point depends on how the previous weak learners performed on the training data. B is true because that's just what the algorithm does. C is false. When $\text{err}_T = 1$, $\beta_T = -\infty$. When β_T is zero, it's because $\text{err}_T = 1/2$. D is false for a similar reason. When the weighted error rate exceeds 0.5, AdaBoost chooses a negative value of β_T, effectively flipping the predictions of the weak learner. The flipped weak learner has an error rate substantially less than 0.5.

(j) [4 pts] Which of the following are valid kernel functions? A kernel function $k(x, z)$ is valid when there exists some function $\Phi : \mathbb{R}^d \to S$ where S is a space (possibly finite, possibly infinite) that has inner products such that we can write $k(x, z) = \Phi(x)^\top \Phi(z)$.

- [x] A: $k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$
- [x] B: $k(x, z) = \|x\| \, \|z\|$
- [x] C: $k(x, z) = x^\top \begin{bmatrix} 727 & 1 \\ 1 & 42 \end{bmatrix} z$
- [ ] D: $k(x, z) = x^\top \begin{bmatrix} -727 & 1 \\ 1 & -42 \end{bmatrix} z$

A: Yes; this is the Gaussian kernel discussed in class.

B: Yes; we can take $\Phi(x) = \|x\|$.

C: Yes; the matrix (call it A) is positive definite, so we can take $\Phi(x) = A^{1/2}x$.

D: No; the matrix has a negative eigenvalue. Hence there exists some vector x such that $k(x, x) < 0$ (for example, $x = [1 \quad 0]^\top$), which is not possible for an inner product.

(k) [4 pts] You are training a neural network with sigmoid activation functions. You discover that you are suffering from the vanishing gradient problem: with most of the training points, most of the components of the gradients are close to zero. It is causing training to be very slow. How could you combat this problem?

- [ ] A: Make the network deeper (more layers).
- [x] B: Make the network shallower (fewer layers).
- [ ] C: Initialize the weights with larger values.
- [x] D: Use ReLU activations instead of sigmoids.

A: False. Making sigmoid-based neural networks deeper without other mitigations can exacerbate the vanishing gradient problem as a natural byproduct of the chain rule. (The derivative of a sigmoid is between 0 and 0.25).

B: True. Reducing the number of layers is likely to increase the norm of the gradient for similar reasoning as above.

C: False. Using larger initial weights will not necessarily increase the norm of gradients. The gradient of a sigmoid vanishes as the input deviates from zero, so larger initial weights could lead to smaller gradients.

D: True. Although ReLU gradients can be zero, it is rarely true in practice that most of the components are zero for most of the training points. A substantial number of components of the gradients will be 1, thereby helping gradient descent to make progress.

(l) [4 pts] Select the true statements about convolutional neural networks.

- [x] A: Pooling layers (of edges) reduce the number of hidden units in the subsequent layer (of units).
- [ ] B: For a convolutional layer, increasing the number of filters decreases the the number of hidden units in the subsequent layer.
- [ ] C: Each unit in a convolutional layer is connected to all units in the previous layer.
- [x] D: For a convolutional layer, increasing the filter height and width decreases the the number of hidden units in the subsequent layer. (Assume no padding.)

A: True. Reducing the number of hidden units is the purpose of pooling.

B: False. Every added filter requires more hidden units.

C: False. Convolutional layers take advantage of local connections; that is, a unit should be connected to only a small patch of units in the previous layer.

D: True. Assuming stride 1 and kernel height K, $H_{out} = H_{in} - K + 1$.

(m) [4 pts] Recall that k-means clustering (Lloyd's algorithm) takes n sample points $X_1, X_2, \ldots, X_n$ and seeks to find a vector y of cluster assignments that minimizes $\sum_{i=1}^{k} \sum_{y_j=i} \|X_j - \mu_i\|^2$, where $y_j \in \{1, 2, \ldots, k\}$ and the cluster center $\mu_i = \frac{1}{n_i} \sum_{y_j=i} X_j$ is the average of the sample points assigned to cluster i. Select the true statements.

○ A: k-means is guaranteed to find clusters that minimize its cost function, as the steps updating the cluster assignments y_j can be solved optimally, and so can the steps updating the cluster means μ_is.

● B: In the algorithm's output, any two clusters are separated by a linear decision boundary.

○ C: It is not possible to kernelize the k-means algorithm, because the means μ_i are not accounted for in the kernel matrix.

○ D: Statisticians justify k-means optimization by assuming a Gaussian prior on the means and applying maximum likelihood estimation.

A: False. k-means is NP-hard, but we do not know that P = NP.

B: True. Each decision boundary is a hyperplane equidistant from two class centroids.

C: False. As we saw in Discussion Section, each μ_i can be written as a linear combination of the x_js, which allows for kernelization.

D: False, haha. This description faintly resembles how statisticians justify ridge regression, but it has nothing to do with k-means clustering.

(n) [4 pts] Select the true statements about the running time of k-means clustering of n sample points with d features each.

● A: The step that updates the cluster means μ_i, given fixed cluster assignments y_j, can be implemented to run in at most $O(nd)$ time.

○ B: Increasing k always increases the running time.

● C: The step that updates the cluster assignments y_j, given fixed cluster means μ_i, can be implemented to run in at most $O(nkd)$ time.

○ D: The k-means algorithm runs in at most $O(nkd)$ time.

A: True. Each of the n sample points contributes to one cluster, and it takes $O(d)$ time to add it to its cluster's sum.

B: False. Consider $k = n$; the algorithm will terminate after the first iteration.

C: True. The main expense is calculating the distance from n sample points to k centroids in d-dimensional space.

D: False. The algorithm may not converge in a constant number of steps. Sometimes it needs exponentially many steps!

(o) [4 pts] Which of the following techniques tend to increase the likelihood that the decision trees in your random forest differ from one another?

- ○ A: Using shorter decision trees
- ● B: Using deeper decision trees
- ● C: Considering only a subset of the features for splitting at a treenode
- ● D: Bagging

Shorter decision trees have less variance whereas deeper decision trees have greater variance. Bagging and randomly restricting the choice of splits are both standard techniques for increasing variation in random forests.

(p) [4 pts] Select the true statements about the bias-variance tradeoff in random forests.

- ● A: Decreasing the number of randomly selected features we consider for splitting at each treenode tends to increase the bias.
- ○ B: Increasing the number of decision trees tends to increase the variance.
- ○ C: Decreasing the number of randomly selected features we consider for splitting at each treenode tends to decrease the bias.
- ● D: Increasing the number of decision trees tends to decrease the variance.

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

- ○ A: The information gain is always strictly positive at each split in the tree.
- ○ B: Pruning is a technique used to reduce tree depth by removing nodes that don't reduce entropy enough.
- ● C: Decision trees with all their leaves pure are prone to overfitting.
- ● D: Calculating the best split among quantitative features for a treenode can be implemented so it takes asymptotically the same amount of time as calculating the best split among binary features.

A: False. Information gain is nonnegative, but it can be zero. Imagine an XOR scenario with two features and four training points arranged in a square, with XOR classifications. Every possible split has information gain zero.

B: False. Pruning is used to remove treenodes if removing them improves validation performance. It isn't based on entropy at all.

C: True. Pure decision trees are sensitive to outliers.

D: True. Choosing a split for n points among d binary features takes $O(nd)$ time, which is the same as the time for d quantitative features (assuming you use radix sort and the algorithm we discussed in class).

(r) [4 pts] Select the true statements about awards given for research related to this course.

- ● A: A Nobel Prize in Physiology was awarded for characterizing action potentials in squid axons.
- ● B: A Nobel Prize in Physiology was awarded for discoveries about how neurons in the visual cortex process images.
- ● C: A Turing Award was awarded for work on deep neural networks.
- ● D: A Gödel Prize was awarded for the paper on AdaBoost.

Q2. [16 pts] Performing PCA by Hand

Let's do principal components analysis (PCA)! Consider this sample of six points $X_i \in \mathbb{R}^2$.

$$\left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}2\\2\end{bmatrix}\right\}.$$

(a) [3 pts] Compute the mean of the sample points and write the centered design matrix $\dot{X}$.

The sample mean is

$$\mu = \frac{1}{6}\sum_{i=1}^{6} X_i = \begin{bmatrix}1\\1\end{bmatrix}.$$

By subtracting the mean from each sample, we form the centered design matrix

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

(b) [6 pts] Find all the principal components of this sample. Write them as **unit** vectors.

The principal components of our dataset are the eigenvectors of the matrix

$$\dot{X}^\top \dot{X} = \begin{bmatrix}4 & 2\\ 2 & 4\end{bmatrix}.$$

The characteristic polynomial of this symmetric matrix is

$$\det(sI - X^\top X) = \det\begin{bmatrix}s-4 & -2\\ -2 & s-4\end{bmatrix} = (s-4)(s-4) - (-2)(-2)$$
$$= s^2 - 8s + 12 = (s-6)(s-2).$$

Hence the eigenvalues of $\dot{X}^\top \dot{X}$ are $\lambda_1 = 2$ and $\lambda_2 = 6$. With these eigenvalues, we can compute the eigenvectors of this matrix as follows. (Or you could just guess them and verify them.)

$$\begin{bmatrix}\lambda_1 - 4 & -2\\ -2 & \lambda_1 - 4\end{bmatrix} v_1 = \begin{bmatrix}-2 & -2\\ -2 & -2\end{bmatrix} v_1 = \begin{bmatrix}0\\0\end{bmatrix} \Rightarrow v_1 = \begin{bmatrix}1/\sqrt{2}\\ -1/\sqrt{2}\end{bmatrix}$$

$$\begin{bmatrix}\lambda_2 - 4 & -2\\ -2 & \lambda_2 - 4\end{bmatrix} v_2 = \begin{bmatrix}2 & -2\\ -2 & 2\end{bmatrix} v_2 = \begin{bmatrix}0\\0\end{bmatrix} \Rightarrow v_2 = \begin{bmatrix}1/\sqrt{2}\\ 1/\sqrt{2}\end{bmatrix}$$

(c) [4 pts]

- Which of those two principal components would be preferred if you use only one? [1 pt]
- What information does the PCA algorithm use to decide that one principal components is better than another? [1 pt]
- From an optimization point of view, why do we prefer that one? [2 pts]

We choose $v_2 = [1/\sqrt{2} \quad 1/\sqrt{2}]^\top$ first.

PCA picks the principal component with the largest eigenvalue.

We prefer it because it maximizes the variance of the sample points after they are projected onto a line parallel to v_2. (Alternatively, because it minimizes the sum of squares of distances from each sample point to its projection on that line.)

(d) [3 pts] Compute the **vector** projection of each of the **original** sample points (**not** the centered sample points) onto your **preferred principal component**. By "vector projection" we mean that the projected points are still in $\mathbb{R}^2$. (Don't just give us the principal coordinate; give us the projected point.)

The vector projection of X_i onto a direction w is

$$\tilde{X}_i = \frac{X_i^\top w}{\|w\|_2^2} w.$$

If w is a unit vector, we can just write $\tilde{X}_i = (X_i^\top w)w$.

For our sample and our chosen principal component $w = v_2$, the projections are

$$\tilde{X}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tilde{X}_2 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \tilde{X}_3 = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}, \tilde{X}_4 = \begin{bmatrix} 3/2 \\ 3/2 \end{bmatrix}, \tilde{X}_5 = \begin{bmatrix} 3/2 \\ 3/2 \end{bmatrix}, \tilde{X}_6 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

Q3. [14 pts] Decision Tree Construction

You are Chief Executive Data Scientist for an e-commerce website. You have collected data on some customers who browsed your site, including whether or not they qualify for free shipping, the time they have spent on the website, whether or not they subscribed to the company's newsletter, and whether or not they made a purchase. The table below is your training set. You want to build a decision tree to predict whether future customers will make a purchase. The rightmost column of the table is the label you wish to predict. The other columns are the features, **except** the customer number, which you should use solely to show us which customers are in which leaves of your decision tree.

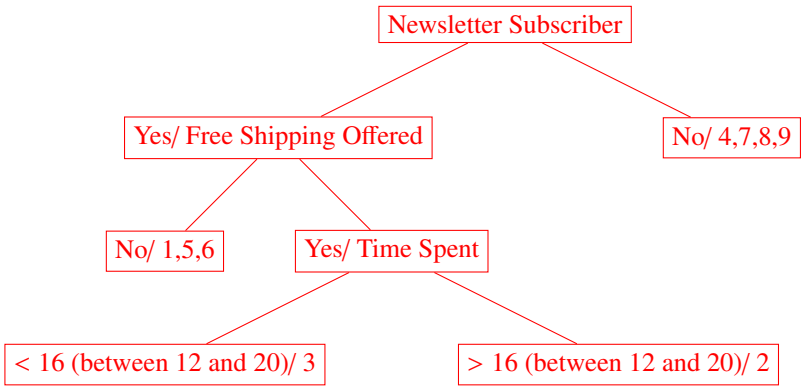
Customer	Free Shipping Offered	Time Spent (min)	Newsletter Subscriber	Purchase
1	No	12	Yes	Yes
2	Yes	20	Yes	Yes
3	Yes	12	Yes	No
4	No	20	No	No
5	No	5	Yes	Yes
6	No	20	Yes	Yes
7	Yes	12	No	No
8	Yes	5	No	No
9	No	12	No	No

(a) [4 pts] **Which feature** should you split on at the root of the decision tree to maximize the information gain? Write an expression for the information gain of the best split. (Your expression can include logarithms and fractions.)

Split on "Newsletter Subscriber". The information gain is

$$-\frac{4}{9}\log_2\frac{4}{9}-\frac{5}{9}\log_2\frac{5}{9}+\frac{4}{9}(0)+\frac{5}{9}\left(\frac{4}{5}\log_2\frac{4}{5}+\frac{1}{5}\log_2\frac{1}{5}\right)=-\frac{4}{9}\log_2\frac{4}{9}-\frac{5}{9}\log_2\frac{5}{9}+\frac{4}{9}(0)+\frac{4}{9}\log_2\frac{4}{5}+\frac{1}{9}\log_2\frac{1}{5}.$$

(b) [8 pts] **Draw the decision tree that maximizes information gain at each split.** The leaves of your tree should be drawn; write inside each leaf the training points it stores. In the internal treenodes, write the splitting features and splitting values. There is no need to write any entropies or information gains. Split until all leaves are pure.



(c) [2 pts] For the decision tree in part (b), suppose we limit the tree to a depth of two (where the root counts as depth zero). That is, the root node can have grandchildren but no great-grandchildren. What will be the training error rate (misclassified training points divided by all training points) of the impure tree?

Only customer 2 or 3 is misclassified. The training error is 1/9.

Q4. [15 pts] Backpropagation on an Arithmetic Expression

Consider an arithmetic network with the inputs a, b, and c, which computes the following sequence of operations, where $s(\gamma) = \dfrac{1}{1+e^{-\gamma}}$ is the logistic (sigmoid) function and $r(\gamma) = \max\{0, \gamma\}$ is the hinge function used by ReLUs.

$$d = ab \quad e = s(d) \quad f = r(a) \quad g = 3a \quad h = 2e + f + g \quad i = ch \quad j = f + i^2$$

We want to find the partial derivatives of j with respect to every other variable a through i, in backpropagation style. This means that for each variable z, we want you to write $\partial j/\partial z$ in two forms: (1) in terms of derivatives involving each variable that *directly* uses the value of z, and (2) in terms of the inputs and intermediate values $a \ldots i$, as simply as possible but with no derivative symbols. For example, we write

$$\frac{\partial j}{\partial i} = \frac{\mathrm{d}j}{\mathrm{d}i} = 2i \quad \text{(no chain rule needed for this one only)}$$
$$\frac{\partial j}{\partial h} = \frac{\partial j}{\partial i}\frac{\partial i}{\partial h} = 2ic \quad \text{(chain rule, then backprop the derivative expressions)}$$

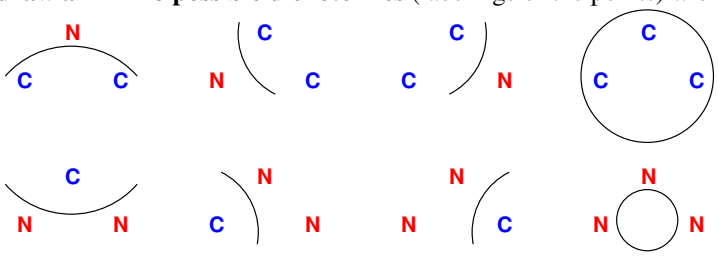
(a) [15 pts] Now, please write expressions for $\partial j/\partial g$, $\partial j/\partial f$, $\partial j/\partial e$, $\partial j/\partial d$, $\partial j/\partial c$, $\partial j/\partial b$, and $\partial j/\partial a$ as we have written $\partial j/\partial h$ above. If they are needed, express the derivative $s'(\gamma)$ in terms of $s(\gamma)$ and express the derivative $r'(\gamma)$ as the indicator function $\mathbf{1}(\gamma \geq 0)$. (*Hint: f is used in two places and a is used in three, so they will need a multivariate chain rule. It might help you to draw the network as a directed graph, but it's not required.*)

$$\frac{\partial j}{\partial g} = \frac{\partial j}{\partial h}\frac{\partial h}{\partial g} = 2ic$$
$$\frac{\partial j}{\partial f} = \frac{\partial j}{\partial h}\frac{\partial h}{\partial f} + \frac{\mathrm{d}j}{\mathrm{d}f} = 2ic + 1$$
$$\frac{\partial j}{\partial e} = \frac{\partial j}{\partial h}\frac{\partial h}{\partial e} = 4ic$$
$$\frac{\partial j}{\partial d} = \frac{\partial j}{\partial e}\frac{\partial e}{\partial d} = 4ic\, s(d)(1 - s(d))$$
$$\frac{\partial j}{\partial c} = \frac{\partial j}{\partial i}\frac{\partial i}{\partial c} = 2ih$$
$$\frac{\partial j}{\partial b} = \frac{\partial j}{\partial d}\frac{\partial d}{\partial b} = 4ica\, s(d)(1 - s(d))$$
$$\frac{\partial j}{\partial a} = \frac{\partial j}{\partial d}\frac{\partial d}{\partial a} + \frac{\partial j}{\partial f}\frac{\partial f}{\partial a} + \frac{\partial j}{\partial g}\frac{\partial g}{\partial a} = 4icb\, s(d)(1 - s(d)) + (2ic + 1) \cdot \mathbf{1}(a \geq 0) + 6ic$$

Q5. [13 pts] We Hope You Learned This

Consider a two-class classifier for points in a two-dimensional feature space ($d = 2$). The decision boundaries are circles. However, the algorithm is a bit dumb, because it can only find classifiers for which the points inside the circle are classified as class C and the points outside are classified at not-class-C. It cannot create classifiers for which the points inside are not-C and the points outside are C.

(a) [4 pts] Draw a picture to show that there exists a set of three points in the plane such that our circle classifier shatters the point set. Specifically, **draw all** $2^3 = 8$ **possible dichotomies** (labelings of the points) **with their decision boundaries**.



(b) [3 pts] There is no set of four points in the plane that our circle classifier can shatter. (Take our word for it.) Suppose that we build a synthetic (computer-generated, not real-world) training set by randomly sampling training points from some statistical distribution (over the features and labels together). We want to have a 99% chance that the best circle classifier (the one that achieves the lowest training error among all possible circle classifiers) has a training error (empirical risk) that predicts, within a margin of 5%, the test error on infinitely many test points (the true risk). What does the fact that our classifier cannot shatter four points tell us about **the asymptotic number of training points** we will need to achieve this (i.e., the **sample complexity** in big-Oh notation)?

It tells us that we need only a constant number of training points. (You could write $O(1)$ points.)

(c) [3 pts] With the same assumptions as part (b), **what statistical assumption** do we need to make about the **test points** to ensure that we have a 99% chance of achieving the goal stated in (b)? (In general, an arbitrary set of labeled points in the plane will have a test error unrelated to the training points; we need to consider how the test points are generated or collected.)

The test points need to be drawn from the same distribution as the training points.

(d) [3 pts] Let's revisit the classifer from part (a), but now we are classifying points with only one feature—that is, all our training/test points are numbers in $\mathbb{R}$. The circle classifying software requires two-dimensional points, so we write wrapper code that appends a zero to the end of each training/test point (as the second feature), then feeds it into the circle classifying software. Strangely, we can no longer shatter any set of three points (in $\mathbb{R}$). **Give an example of a dichotomy on three distinct points in $\mathbb{R}$ that the wrapper-code classifier cannot represent.**

C X C

Q6. [20 pts] Quadratic Regression

You are given a design matrix $X \in \mathbb{R}^{n \times d}$, storing n training points with d features, and a vector $y \in \mathbb{R}^n$ of continuously-valued labels. As usual, let X_i be training point i, expressed as a column vector such that row i of X is $X_i^\top$. We want to perform least-squares regression with the quadratic regression function $h(z) = z^\top W z$ (no linear nor constant terms), where $W \in \mathbb{R}^{d \times d}$ is symmetric. The objective is to find the symmetric matrix W that minimizes the least-squares cost function

$$J(W) = \sum_{i=1}^{n} \left(X_i^\top W X_i - y_i \right)^2 .$$

(*Note: If you get stuck on one part, try the other parts; each part is independent of solving the others.*)

(a) [6 pts] Find $\nabla_W J(W)$. (*Hint: It should be a $d \times d$ matrix like W. If you don't remember how to take a derivative with respect to a matrix, try writing $X_i^\top W X_i$ as a summation, but your final answer should have just one summation, over i.*)

$$\begin{aligned}
\nabla_W J(W) &= \sum_{i=1}^{n} \nabla_W (X_i^\top W X_i - y_i)^2 \\
&= \sum_{i=1}^{n} 2(X_i^\top W X_i - y_i) \cdot \nabla_W (X_i^\top W X_i - y_i) \\
&= 2 \sum_{i=1}^{n} (X_i^\top W X_i - y_i) X_i X_i^\top .
\end{aligned}$$

(b) [4 pts] Suppose that X has rank n. (That is, the training points are linearly independent and $n \leq d$.) Let I_i be the **row vector** that is row i of the $n \times n$ identity matrix; hence, its entries are zero except a 1 at position i. Observe that we can write training point i as the row vector $X_i^\top = I_i X$. Let X^+ be the Moore–Penrose pseudoinverse of X. As a prelude to part (c), show that $X_i^\top X^+ = I_i$. (*Hint: What do you know about pseudoinverses?*)

By postmultiplying both sides of $X_i^\top = I_i X$ by X^+, we have $X_i^\top X^+ = I_i X X^+ = I_i$. (As X is an $n \times d$ matrix with rank n, $XX^+ = I$.)

(c) [5 pts] Again suppose that X has rank n. Write the labels as a diagonal $n \times n$ matrix

$$Y = \operatorname{diag}(y_i) = \begin{bmatrix} y_1 & 0 & \dots & 0 \\ 0 & y_2 & & 0 \\ \vdots & & \ddots & \\ 0 & \dots & & y_n \end{bmatrix}.$$

Using (b), show that the cost function $J(W)$ is minimized by $W^* = X^+ Y X^{+\top}$, where $X^{+\top}$ denotes the transpose of X^+.

For each $i \in [1, n]$,

$$X_i^\top W^* X_i = X_i^\top X^+ Y X^{+\top} X_i = I_i Y I_i^\top = y_i.$$

Therefore, $J(W^*) = \sum_{i=1}^{n} (X_i^\top W^* X_i - y_i)^2 = 0$. As $J(W)$ is nonnegative for all W, no matrix has a lower cost than W^*, so W^* is optimal.

Note: we will also accept answers that show that $\nabla_W J(W)$ is zero at $W = W^*$ (instead of showing that $J(W^*) = 0$).

(d) [5 pts] Suppose the ground truth for the labels is $g(z) = z^\top A z$, where $A \in \mathbb{R}^{d \times d}$ is a fixed, symmetric matrix. Each training label is drawn from this ground truth with random Gaussian noise; that is, $y_i = g(X_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, 1)$. Assume that the regression function we learn is $h(z) = z^\top W^* z$ where W^* is given in part (c). Show that the **bias** of this regression method at a test point z is

$$\text{bias}(z) = |z^\top (X^+ \operatorname{diag}(X_i^\top A X_i) X^{+\top} - A) z|,$$

where $\text{diag}(X_i^\top A X_i)$ is a diagonal $n \times n$ matrix whose diagonal values are $X_i^\top A X_i$.

$$\text{bias}(z) = |E[h(z)] - g(z)| = |E[z^\top X^+ Y X^{+\top} z] - z^\top A z| = |z^\top (X^+ E[Y] X^{+\top} - A) z|.$$

$E[Y]$ is a diagonal matrix whose diagonal values are $E[y_i] = E[g(X_i) + \epsilon_i] = g(X_i) = X_i^\top A X_i$, so

$$\text{bias}(z) = |z^\top (X^+ \text{diag}(X_i^\top A X_i) X^{+\top} - A) z|.$$