CS 189/289A Introduction to Machine Learning
Spring 2021 Jonathan Shewchuk Final

- The exam is open book, open notes for material on **paper**. On your computer screen, you may have only this exam, Zoom, a limited set of PDF documents (see Piazza for details), and four browser windows/tabs: Gradescope, the exam instructions, clarifications on Piazza, and the form for submitting clarification requests.
- You will submit your answers to the multiple-choice questions directly into Gradescope via the assignment "**Final – Multiple Choice**"; please **do not** submit your multiple-choice answers on paper. If you are in the DSP program and have been granted extra time, select the "DSP, 150%" or "DSP, 200%" option. By contrast, you will submit your answers to the written questions by writing them on paper by hand, scanning them, and submitting them through Gradescope via the assignment "**Final – Free Response**."
- Please write your name at the top of each page of your written answers. (You may do this before the exam.) **Please start each top-level question (Q2, Q3, etc.) on a new sheet of paper. Clearly label all written questions and all subparts of each written question**.
- You have **180 minutes to complete the minal exam (3:10–6:10 PM)**. (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.)
- When the exam ends (6:10 PM), **stop writing**. You must submit your multiple-choice answers before 6:10 PM sharp. **Late multiple-choice submissions will be penalized at a rate of 5 points per minute after 6:10 PM.** (The multiple-choice questions are worth 60 points total.)
- From 6:10 PM, you have 15 minutes to scan the written portion of your exam and turn it into Gradescope via the assignment "Final – Free Response." Most of you will use your cellphone/pad and a third-party scanning app. If you have a physical scanner, you may use that. **Late written submissions will be penalized at a rate of 10 points per minute after 6:25 PM.** (The written portion is worth 90 points total.)
- Following the exam, you must use Gradescope's **page selection mechanism** to mark which questions are on which pages of your exam (as you do for the homeworks). Please get this done before midnight. This can be done on a computer different than the device you submitted with.
- The total number of points is 150. There are 15 multiple choice questions worth 4 points each, and six written questions worth a total of 90 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.

Q1. [60 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 conditions could serve as a sensible stopping condition while building a decision tree?

○ A: Stop if you find the validation error is decreasing as the tree grows

○ B: Don't split a treenode that has an equal number of sample points from each class

○ C: Don't split a treenode whose depth exceeds a specified threshold

○ D: Don't split a treenode if the split would cause a large reduction in the weighted average entropy

(b) [4 pts] Let classifier A be a random forest. Let classifier B be an ensemble of decision trees with bagging—identical to classifier A except that we do not limit the splits in each treenode to a subset of the features; at every treenode, the very best split among all d features is chosen. Which statements are true?

○ A: After training, all the trees in classifier B must be identical

○ B: Classifier B will tend to have higher bias than Classifier A

○ C: Classifier B will tend to have higher variance than Classifier A

○ D: Classifier B will tend to have higher training accuracy than Classifier A

(c) [4 pts] We are using an ensemble of decision trees for a classification problem, with bagging. We notice that the decision trees look too similar. We would like to build a more diverse set of learners. What are possible ways to accomplish that?

○ A: Increase the size of each random subsample

○ B: Decrease the size of each random subsample

○ C: Apply normalization to the design matrix first

○ D: Sample without replacement

(d) [4 pts] Suppose we have a feature map Φ and a kernel function $k(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j)$. Select the true statements about kernels.

○ A: If there are n sample points of dimension d, it takes $O(nd)$ time to compute the kernel matrix

○ B: The kernel trick implies we do not compute $\Phi(X_i)$ explicitly for any sample point X_i

○ C: For every possible feature map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ you could imagine, there is a way to compute $k(X_i, X_j)$ in $O(d)$ time

○ D: Running times of kernel algorithms do not depend on the dimension D of the feature space $\Phi(\cdot)$

(e) [4 pts] Which of the following are benefits of using the backpropogation algorithm to compute gradients?

○ A: Its running time is linear in the total number of units (neurons) in the network

○ B: It can be applied to any arithmetic function (assuming the directed computation graph has no cycles and we evaluate the gradient at a point where the gradient exists)

○ C: Compared to naive gradient computation, it improves the speed of each iteration of gradient descent by eliminating repeated computations of the same subproblem

○ D: Compared to naive gradient computation, it reduces the number of iterations required to get close to a local minimum, by protecting against sigmoid unit saturation (vanishing gradients)

(f) [4 pts] Which of the following are benefits of using convolutional neural networks—as opposed to fully connected ones—for image recognition tasks?

○ A: The ability to express a wider variety of more complicated functions of the input features

○ B: Fewer model architecture hyperparameters for the designer to select

○ C: Enables the network to more easily learn and recognize features regardless of their position in the image

○ D: Typically requires less data to train well

(g) [4 pts] Select the correct statements about principal component analysis (PCA).

○ A: PCA is a method of dimensionality reduction

○ B: If we select only one direction (a one-dimensional subspace) to represent the data, the sample variance of the projected points is zero if and only if the original sample points are all identical

○ C: The orthogonal projection of a point x onto a unit direction vector w is $(x^\top w)w$

○ D: If we select only one direction (a one-dimensional subspace) to represent the data, PCA chooses the eigenvector of the sample covariance matrix that corresponds to the least eigenvalue

(h) [4 pts] Consider a **centered** design matrix $X \in \mathbb{R}^{n\times d}$, where $X_i \in \mathbb{R}^d$ is the i-th sample point (a column vector) and $X_i^\top$ is the i-th row of X. Which of these optimization problems is a correct formulation of finding the first principal component v?

○ A: Subject to $\|v\|_2 = 1$, find $v \in \mathbb{R}^d$ that minimizes $\sum_{i=1}^n \|X_i - vv^\top X_i\|_2^2$

○ B: Find $v \in \mathbb{R}^d$ that minimizes $\dfrac{v^\top X^\top X v}{\|v\|_2^2}$

○ C: Subject to $\|v\|_2 = 1$, find $v \in \mathbb{R}^d$ that maximizes $\sum_{i=1}^n \|X_i - vv^\top X_i\|_2^2$

○ D: Find $v \in \mathbb{R}^d$ that maximizes $\dfrac{v^\top X^\top X v}{\|v\|_2^2}$

(i) [4 pts] Consider a **centered** design matrix $X \in \mathbb{R}^{n\times d}$ and its singular value decomposition $X = UDV^\top$. X has d principal components, which are found by principal components analysis (PCA). Which statements are correct?

○ A: The row space of X (if not trivial) is spanned by some of the principal components

○ B: The null space of X (if not trivial) is spanned by some of the principal components

○ C: The principal components are all right singular vectors of X

○ D: The matrix UD lists the principal coordinates of every sample point in X

(j) [4 pts] Select the correct statements about the Fiedler vector.

○ A: The Fiedler vector is the eigenvector of the Laplacian matrix that is associated with the smallest eigenvalue

○ B: The Fiedler vector always satisfies the balance constraint (as written as an equation)

○ C: The Fiedler vector is a solution to the unrelaxed, NP-hard optimization problem

○ D: The *sweep cut* is a spectral graph partitioning technique that tries $n-1$ different cuts (in an n-vertex graph) and picks one of them.

(k) [4 pts] Consider the optimization problem of finding $q \in \mathbb{R}^r$ that minimizes $\|y - XPq\|_2^2 + \lambda\|q\|_2^2$, where $\lambda > 0$, $X \in \mathbb{R}^{n\times d}$ is a design matrix, $y \in \mathbb{R}^n$ is a vector of labels, and $P \in \mathbb{R}^{d\times r}$ is an arbitrary matrix with $r < d$. This problem has one unique solution. Which of the following is that one unique solution?

○ A: $q = (P^\top X^\top XP + \lambda I)^{-1} P^\top X^\top y$

○ B: $q = (X^\top P^\top PX + \lambda I)^{-1} P^\top X^\top y$

○ C: $q = (P^\top X^\top XP + \lambda I)^{-1} X^\top y$

○ D: $q = (X^\top P^\top PX + \lambda I)^{-1} X^\top y$

(l) [4 pts] Select the correct statements about AdaBoost.

○ A: "Ada" stands for "adaptive," as the meta-learner adapts to the performance of its learners

○ B: AdaBoost works best with support vector machines

○ C: At test/classification time, AdaBoost computes a weighted sum of predictions

○ D: AdaBoost can transform any set of classifiers to a classifier with better training accuracy

(m) [4 pts] Select the correct statements about AdaBoost.

○ A: When we go from iteration t to iteration $t + 1$, the weight of sample point X_i is increased if the majority of the t learners misclassify X_i.

○ B: Unlike with decision trees, two data points that are identical ($X_i = X_j$) but have different labels ($y_i \neq y_j$) can be classified correctly by Adaboost.

○ C: AdaBoost can benefit from a learner that has only 5% training accuracy.

○ D: If you train enough learners and every learner achieves at least 51% validation accuracy, Adaboost can always achieve 100% validation accuracy.

(n) [4 pts] In which of the following cases should you prefer ***k*-nearest neighbors over *k*-means clustering**? For all the four options, you have access to images $X_1, X_2, \ldots, X_n \in \mathbb{R}^d$.

○ A: You do **not** have access to labels. You want to find out if any of the images are very different from the rest, i.e., are outliers.

○ B: You have access to labels $y_1, y_2, \ldots, y_n$ telling us whether image i is a cat or a dog. You want to find out whether the distribution of cats is unimodal or bimodal. You already know that the distribution of cats either has either one or two modes, but that's all you know about the distribution.

○ C: You have access to labels $y_1, y_2, \ldots, y_n$ telling us whether image i is a cat or a dog. You want to find out whether a new image z is a cat or a dog.

○ D: You have access to labels $y_1, y_2, \ldots, y_n$ telling us whether image i is a cat or a dog. Given a new image z, you want to approximate the posterior probability of z being a cat and the posterior probability of z being a dog.

(o) [4 pts] Select the correct statements about the k-nearest neighbor classifier.

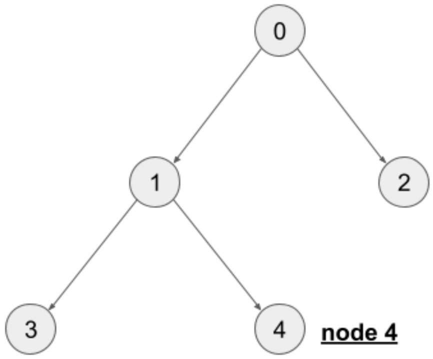
○ A: For exact nearest neighbor search in a very high-dimensional feature space, generally it is faster to use exhaustive search than to use a k-d tree.

○ B: When using a k-d tree, approximate nearest neighbor search is sometimes substantially faster than exact nearest neighbor search.

○ C: When using exhaustive search, approximate nearest neighbor search is sometimes substantially faster than exact nearest neighbor search.

○ D: Order-k Voronoi diagrams are widely used in practice for k-nearest neighbor search (with $k > 1$) in a two-dimensional feature space.

Q2. [20 pts] A Decision Tree



node 2

Label	A	B
# Samples	10	0

node 3

Label	A	B
# Samples	3	2

node 4

Label	A	B
# Samples	2	8

The decision tree drawn above stores sample points from two classes, A and B. The tables indicate the number of sample points of each class stored in each leaf node.

(a) [4 pts] What is entropy at the root (Treenode 0)? Simplify your answer if possible, but you don't need to compute logarithms.

(b) [6 pts] What is the information gain of the split at the root node (Treenode 0)? Show your work so we understand how you got that answer. Simplify your answer as much as possible, but again, you don't need to compute logarithms.

(c) [2 pts] What is the training accuracy of the tree?

(d) [4 pts] How can we increase the decision tree's training accuracy? Give a reasonable explanation for why we might have considered doing that but decided not to do that. ("Because we set a limit" is not a valid answer; we're looking for the underlying reason why we would set a limit.)

(e) [4 pts] Suppose I told you that we *did* increase the decision tree's training accuracy in the way you suggest in part (d), but when training completed, the decision tree training algorithm decided to revert to this tree in the end anyway. Explain how that might happen and why it might be the right thing to do.

Q3. [20 pts] The Singular Value Decomposition

We want you to compute (by hand) part of the singular value decomposition (SVD) $X = UDV^\top$ of the matrix

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

(a) [6 pts] One way to compute the SVD by hand is to exploit the relationship between the singular values of X and the eigenvalues of a related symmetric matrix. What symmetric matrix has eigenvalues related to the singular values of X? Compute that matrix, write down its characteristic polynomial and simplify it, and determine its eigenvalues. Show your work!

(b) [4 pts] What is the relationship between the singular values of X and the eigenvalues of your symmetric matrix? Use that relationship to write down the two specific singular values of X.

(c) [5 pts] Generalize your answer above. For **any** matrix Z (not only X), what symmetric matrix has eigenvalues related to the singular values of Z? **Prove** that every singular value of Z can be identified by looking at the eigenvalues of the matrix your method constructs.

(d) [5 pts] Explicitly write down enough eigenvectors of your symmetric matrix to form a complete basis for the space. (Hint: the symmetry of your matrix should make it easy to guess eigenvectors; then you can confirm that they are eigenvectors.) Based on these, write down the value of U or V (whichever is appropriate).

Q4. [10 pts] Maximum Likelihood Estimation of a Mean

In class, we derived the maximum likelihood estimates for the mean and variance of a univariate (or isotropic multivariate) normal distribution, based on n points sampled independently from that distribution. Then we considered the more general (anisotropic) multivariate distribution $\mathcal{N}(\mu, \Sigma)$, and I told you the MLE estimates without deriving them. Let's derive the maximum likelihood estimate for the mean.

We are given n independent sample points $X_1, X_2, \ldots, X_n \sim \mathcal{N}(\mu, \Sigma)$ with d features each, where $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d \times d}$ is symmetric and positive definite.

(a) [4 pts] Write out an expression for the likelihood $\mathcal{L}(\mu, \Sigma; X_1, \ldots, X_n)$. (You should substitute in the normal PDF; a placeholder will only get part marks.) Simplify the expression as much as possible.

(b) [6 pts] Derive the maximum likelihood estimate $\hat{\mu}$ of the distribution mean μ. (You probably already know what its value should be. If you get it right, it will not depend on Σ.)

Q5. [20 pts] Kernel Principal Components Analysis

In this problem, we will derive the kernelized version of principal components analysis (PCA). You are given a **centered** $n \times d$ design matrix X whose rows express the sample points $X_1^\top, X_2^\top, ..., X_n^\top$, and a feature map Φ that maps each sample point $X_i \in \mathbb{R}^d$ to a high-dimensional "lifted" sample point $\Phi(X_i) \in \mathbb{R}^D$, where $D \gg d$. Define the kernel function $k(X_i, X_j) = \Phi(X_i)^\top \Phi(X_j)$, and suppose it takes $O(d)$ time to compute (which is much, much less than D).

The sample covariance matrix of the lifted points in $\mathbb{R}^D$ is $C = \frac{1}{n}\Phi(X)^\top \Phi(X) = \frac{1}{n}\sum_{i=1}^n \Phi(X_i)\,\Phi(X_i)^\top \in \mathbb{R}^{D\times D}$. (Recall that $\Phi(X) \in \mathbb{R}^{n\times D}$. Recall also that our convention is to express each X_i and each $\Phi(X_i)$ as a column vector, but these points are stored as rows of X and $\Phi(X)$, respectively, which is why the transpose moved.)

(a) [6 pts] To perform PCA, we want to find the eigenvectors of C with nonzero eigenvalues. However, we don't ever want to construct C—it's way too big! We don't even want to construct one eigenvector of C! Show that if a vector $v \in \mathbb{R}^D$ is an eigenvector of C with a nonzero eigenvalue, then v can be expressed as a linear combination of lifted sample points: that is, $v = \sum_{j=1}^{n} a_j \Phi(X_j) = \Phi(X)^\top a$ for some vector $a \in \mathbb{R}^n$ of "dual" coefficients. Hint: think of the standard definition of an eigenvector of C, then substitute in the value of C.

(b) [5 pts] Recall that the kernel matrix $K = \Phi(X)\,\Phi(X)^\top$ is the symmetric $n \times n$ matrix such that $K_{ij} = k(X_i, X_j)$. In the circumstance described in part (a), prove that $K^2 a = \lambda n K a$. Hint: again start with the standard definition of an eigenvector of C, then substitute in the value of v from above.

(c) [5 pts] Conversely, suppose that a is an eigenvector of K with eigenvalue λ_a. Show that $v = \Phi(X)^\top a$ is an eigenvector of C. What is v's corresponding eigenvalue?

(d) [4 pts] The parts above imply that by finding the eigendecomposition of the kernel matrix K, we can indirectly express the eigenvectors (that have nonzero eigenvalues) of the kernelized covariance matrix C without ever explicitly computing C nor any vector of length D. Suppose that we have a test point $z \in \mathbb{R}^d$ and we want to efficiently compute the principal coordinate of $\Phi(z)$ projected onto a principal component $v = \sum_{j=1}^{n} a_j \Phi(X_j) = \Phi(X)^\top a$. Assume that we have already computed a. Describe how to compute $\Phi(z)$'s principal coordinate quickly, without ever computing a vector of length $\Theta(D)$ or doing a computation taking $\Theta(D)$ time. (Recall our assumption that kernel function computations are much, much faster than that.)

Q6. [20 pts] Boosting and 0-1 Loss

(This question has independent parts. If you get stuck on one, try the others. You may use the statements made in the previous parts for each question. Part (d) is the easiest one. Please show your work!)

Recall that the AdaBoost algorithm computes coefficients β_t for classifiers G_t, $t \in [1, T]$, and uses the *exponential loss function* $L(\rho, \ell) = e^{-\rho\ell}$, where both the prediction ρ and the label ℓ are $+1$ or -1. After t rounds of training, the metalearner's output is $\sum_{k=1}^{t} \beta_k G_k$.

Let the sample points be $X_1, X_2, \ldots, X_n \in \mathbb{R}^d$ with labels $y_1, y_2, \ldots, y_n \in \{+1, -1\}$. Given $G_1, \ldots G_{t-1}$ with coefficients $\beta_1, \ldots \beta_{t-1}$, AdaBoost calculates β_t and classifier G_t that minimizes the *exponential risk* $R(\sum_{k=1}^{t} \beta_k G_k) = \frac{1}{n} \sum_{i=1}^{n} L(\sum_{k=1}^{t} \beta_k G_k(X_i), y_i)$ of the metalearner output $\sum_{k=1}^{t} \beta_k G_k$.

Recall from lecture that $w_i^{(t)}$ is the weight assigned to training point X_i at iteration t; $\mathrm{err}_t = \sum_{i:G_t(X_i) \neq y_i} w_i^{(t)}$ is the (weighted) error rate of classifier G_t; and $\beta_t = \frac{1}{2} \ln\left(\frac{1-\mathrm{err}_t}{\mathrm{err}_t}\right)$. The initial weights are $w_i^{(1)} = 1/n$. For the next iteration $t+1$, we will assign sample point X_i a new weight $w_i^{(t+1)} = \frac{w_i^{(t)}}{Z_t} e^{-\beta_t y_i G_t(X_i)}$, where $Z_t = \sum_{i=1}^{n} w_i^{(t)} e^{-\beta_t y_i G_t(X_i)}$ is a normalization divisor that ensures the weights in iteration $t+1$ sum to 1. (Recall that we introduced this normalization in Homework 7.)

(a) [5 pts] We can write $w_i^{(t+1)} = \omega\, L(\sum_{k=1}^{t} \beta_k G_k(X_i), y_i)$ where ω does not include any w_i nor β_j terms (but it can include Z_k terms). What is ω?

(b) [4 pts] Show that the *exponential risk* is $R(\sum_{k=1}^{t} \beta_k G_k) = \prod_{k=1}^{t} Z_k$.

(c) [4 pts] By substituting the values of β_k, show that $R(\sum_{k=1}^{t} \beta_k G_k) = \prod_{k=1}^{t} 2\sqrt{\mathrm{err}_k(1-\mathrm{err}_k)}$.

(d) [7 pts] Although AdaBoost defaults to the exponential loss function L, it is educational to see how boosting behaves under the *0-1 loss function* $L_{01}(\rho, \ell) = \mathbf{1}[\rho \neq \ell]$, which is 1 for incorrect predictions and 0 for correct ones. The main observation is that boosting reduces the average 0-1 loss just as dramatically as the average exponential loss.

Show that the *0-1 risk* $R_{01}(f) = \frac{1}{n} \sum_{i=1}^{n} L_{01}(f(X_i), y_i)$ is less than or equal to the *exponential risk* $R(f)$ for any classifier $f(\cdot)$.