

- Please do not turn over the page before you are instructed to do so.
- You have 2 hours and 50 minutes.
- Please write your initials on the top-right of each odd-numbered page (e.g., write “AE” if you are Alexei Efros). Complete this by the end of your 2 hours and 50 minutes.
- The exam is closed book, closed notes except your one-page cheat sheet.
- No calculators or other electronic devices allowed.
- Mark your answers ON THE EXAM ITSELF IN THE SPACE PROVIDED. If you are not sure of your answer you may wish to provide a *brief* explanation. Do NOT attach any extra sheets.
- The total number of points is 150. There are 15 true/false questions worth 2 points each, 10 multiple choice questions worth 3 points each, and 6 descriptive questions with unequal point assignments.
- For true/false questions, fill in the *True/False* bubble.
- For multiple-choice questions, fill in the bubbles for **ALL CORRECT CHOICES**: There may be more than one correct choice, but there will be at least one correct choice. NO PARTIAL CREDIT: the set of all correct answers must be checked.

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

Q1. [30 pts] True or False

- (1) [2 pts] Random forests usually perform better than AdaBoost when your dataset has mislabeled data points.
 True False
- (2) [2 pts] The discriminant function computed by kernel methods are a linear function of its parameters, not necessarily a linear function of the inputs.
 True False
- (3) [2 pts] The XOR operator can be modeled using a neural network with a single hidden layer (i.e. 3-layer network).
 True False
- (4) [2 pts] Convolutional neural networks are rotation invariant.
 True False
- (5) [2 pts] Making a decision tree deeper will assure better fit but reduce robustness.
 True False
- (6) [2 pts] Bagging makes use of the bootstrap method.
 True False
- (7) [2 pts] K-means automatically adjusts the number of clusters.
 True False
- (8) [2 pts] Dimensionality reduction can be used as pre-processing for machine learning algorithms like decision trees, kd-trees, neural networks etc.
 True False
- (9) [2 pts] K-d trees guarantee an exponential reduction in the time it takes to find the nearest neighbor of an example as compared to the naive method of comparing the distances to every other example.
 True False
- (10) [2 pts] Logistic regression is equivalent to a neural network without hidden units and using cross-entropy loss.
 True False
- (11) [2 pts] Convolutional neural networks generally have fewer free parameters as compared to fully connected neural networks.
 True False
- (12) [2 pts] K-medoids is a kind of agglomerative clustering.
 True False
- (13) [2 pts] Whitening the data doesn't change the first principal direction.
 True False
- (14) [2 pts] PCA can be kernelized.
 True False
- (15) [2 pts] Performing K-nearest neighbors with $K = N$ yields more complex decision boundaries than 1-nearest neighbor.
 True False

Q2. [30 pts] Multiple Choice

(1) [3 pts] Which of the following guidelines is applicable to initialization of the weight vector in a fully connected neural network.

- Should not set it to zero since otherwise it will cause overfitting
- Should set it to zero since otherwise it causes a bias
- Should not set it to zero since otherwise (stochastic) gradient descent will explore a very small space
- Should set it to zero in order to preserve symmetry across all neurons

(2) [3 pts] Duplicating a feature in linear regression

- Can reduce the L2-Penalized Residual Sum of Squares.
- Can reduce the L1-Penalized Residual Sum of Squares (RSS).
- Does not reduce the Residual Sum of Squares (RSS).
- None of the above

(3) [3 pts] Which of the following is/are forms of regularization in neural networks.

- Weight decay
- L1 regularization
- L2 regularization
- Dropout

(4) [3 pts] We are given a classifier that computes probabilities for two classes (positive and negative). The following is always true about the ROC curve, and the area under the ROC curve (AUC):

- An AUC of 0.5 represents a classifier that performs worse than random.
- The ROC curve allows us to visualize the tradeoff between true positive and false positive classifications.
- We generate an ROC curve by varying the discriminative threshold of our classifier.
- The ROC curve monotonically increases.

(5) [3 pts] The K-means algorithm:

- Requires the dimension of the feature space to be no bigger than the number of samples
- number of clusters
- Has the smallest value of the objective function when $K = 1$
- Converges to the global optimum if and only if the initial means are chosen as some of the samples themselves
- Minimizes the within class variance for a given
- None of the above

(6) [3 pts] Suppose when you are training your convolutional neural network, you find that the training loss just doesn't go down after initialization. What could you try to fix this problem?

- Change the network architecture
- Find a better model
- Change learning rates
- Normalize the inputs to the network
- Ensure training data is being read correctly
- Add a regularization term

(7) [3 pts] Logistic regression:

- Minimizes cross-entropy loss
- Has a simple, closed form analytical solution
- Models the log-odds as a linear function
- Is a classification method to estimate class posterior probabilities

(8) [3 pts] Select all the true statements.

- The first principal component is unique up to a sign change.
- The last principal component is unique up to a sign change.
- All principal components are unique up to a sign change.
- If some features are linearly dependent, at least one singular value is zero.
- If some features are correlated, at least one singular value is zero.

(9) [3 pts] Select all the choices that make the following statement true:

In (a), the training error does not increase as (b) increases.

- a: K-means,
b: number of iterations
- a: Training neural nets with back propagation using *batch* gradient decent,
b: number of iterations
- a: Training neural nets with back propagation using *stochastic* gradient decent,
b: number of iterations
- a: Regression Trees with square loss,
b: depth of the tree
- a: Random Forest Classifier,
b: number of trees in the forest
- a: Least squares,
b: number of features

(10) [3 pts] Neural networks:

- Optimize a convex objective function
- Can only be trained with stochastic gradient descent
- Can use a mix of different activation functions
- Can be made to perform well even when the number of parameters/weights is much greater than the number of data points.

Q3. [15 pts] Nearest Neighbors and Bayes risk

In this problem, we want to investigate whether given enough training examples, the Bayes decision rule gives more accurate results than nearest neighbors.

A life insurance company needs to estimate whether a client is at risk of dying in the year to come, based on his age and blood pressure. We call $\mathbf{x} = [A, B]$ (A =Age, B =Blood pressure) the two dimensional input vector and y the outcome ($y = 1$ if the client dies and $y = -1$ otherwise). The insurance company has a lot of data, enough to estimate accurately with Parzen windows the posterior probability $P(y = 1|\mathbf{x})$. This is represented in a diagram in Figure 1.

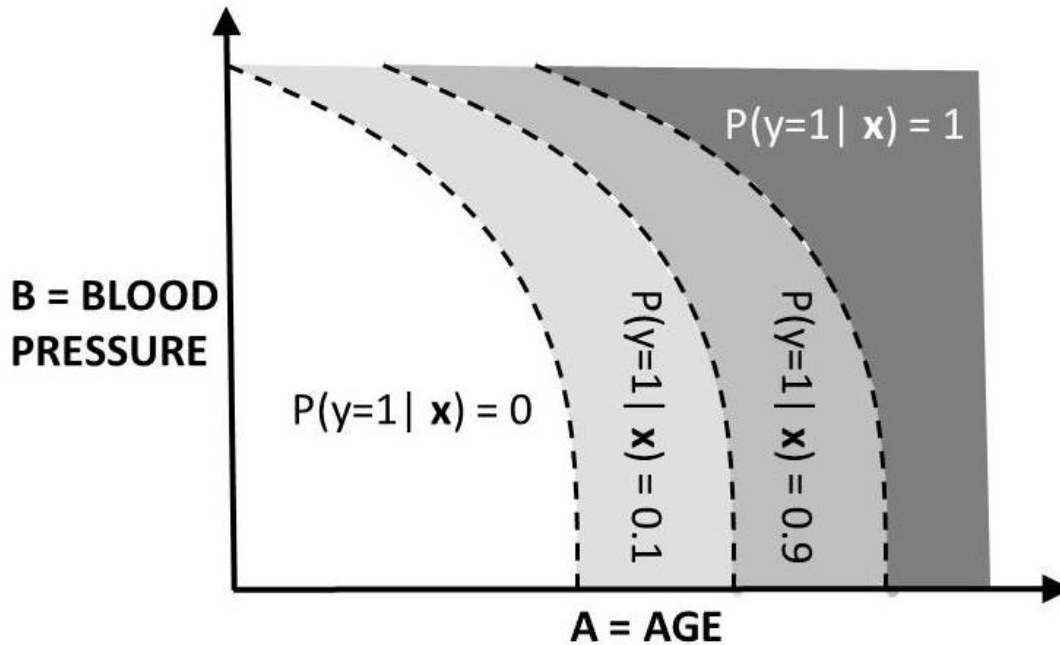


Figure 1: Draw your answer to the Nearest Neighbor and Bayes risk problem. Note that the distribution $P(x)$ is assumed to be uniform across the area shown in the figure.

Note: No worries, nobody died, this is a fictitious example. For simplicity we have 4 regions in which $P(y = 1|\mathbf{x})$ is constant, and the distribution $P(x)$ is assumed to be uniform across the area shown in the figure.

Let us name the different regions:

- $R1 : P(y = 1|\mathbf{x}) = 0$
- $R2 : P(y = 1|\mathbf{x}) = 0.1$
- $R3 : P(y = 1|\mathbf{x}) = 0.9$
- $R4 : P(y = 1|\mathbf{x}) = 1$

- (1) [2 pts] Draw on the figure the Bayes optimum decision boundary with a thick line.
- (2) [2 pts] What is the Bayes risk in each of the four regions (the Bayes risk is the probability of error of the optimum Bayes classifier).
 - R1: $E_{\text{Bayes}} = 0$
 - R2: $E_{\text{Bayes}} = 0.1$
 - R3: $E_{\text{Bayes}} = 0.1$
 - R4: $E_{\text{Bayes}} = 0$
- (3) [4 pts] Assume we have lots and lots of samples due to which we can assume that the nearest neighbor of any sample lies in the same region as that sample. Now consider any sample, say, \mathbf{x} which falls in region R_i . For $i \in \{1, 2, 3, 4\}$, find the probability that \mathbf{x} and its nearest neighbor belong to different classes (that is, have

different labels):

1. If they both fall in R1: 0
2. If they both fall in R2: $x \cdot 0.9 (+1) \times 0.1 (+1) + x \cdot 0.1 (+1) \times 0.9 (+1) \Rightarrow 2 * 0.1 * 0.9 = 0.18$
3. If they both fall in R3: same as R2: $2 * 0.1 * 0.9$
4. If they both fall in R4: 0

(4) [2 pts] What is the nearest neighbor error rate ENN in each region:

- R1: ENN = 0
R2: ENN = 0.18
R3: ENN = 0.18
R4: ENN = 0

(5) [5 pts] Now let us generalize the previous results to the case where the posterior probabilities are:

$$\begin{aligned}R1 : P(y = 1|\mathbf{x}) &= 0 \\R2 : P(y = 1|\mathbf{x}) &= p \\R3 : P(y = 1|\mathbf{x}) &= (1 - p) \\R4 : P(y = 1|\mathbf{x}) &= 1\end{aligned}$$

where p is a number between 0 and 0.5.

After recalculating the results of the previous questions, give an upper bound and a lower bound of ENN in terms of EBayes.

$$\begin{aligned}R1 : EBayes &= 0 \text{ ENN} = 0 \\R2 : EBayes &= p \text{ ENN} = 2p(1 - p) \\R3 : EBayes &= p \text{ ENN} = 2p(1 - p) \\R4 : EBayes &= 0 \text{ ENN} = 0\end{aligned}$$

$$EBayes \leq ENN \leq 2 EBayes(1-EBayes)$$

Q4. [11 pts] Curse of dimensionality

When the dimension of input space d is large, the performance of the nearest neighbor algorithm (and other local methods such as “Parzen windows”) tends to deteriorate. This phenomenon is known as the “curse of dimensionality”. In the following questions, we will assume that we have a training set of fixed size N and that all features are uniformly distributed on $[0, 1]$. Associated with each test example \mathbf{x} is a predicted response y corresponding to the average of the responses associated to the training examples that are near \mathbf{x} .

- (1) [1 pt] Suppose we have only one feature ($d = 1$) and we want to make prediction using only training examples that are within 10% of the input range. For instance, to predict the response y of $x = 0.6$, we will use the training examples that fall in the range $[0.55, 0.65]$ only. On average, what fraction of the training examples will we use to make each prediction?

10%

- (2) [1 pt] Now suppose that we have two features ($d = 2$) and we want to predict using only training examples that are within 10% of the input range in both dimensions. For instance, to predict the response y of $\mathbf{x} = (0.6, 0.35)$, we will use the training examples that fall in the range $[0.55, 0.65]$ for the first feature and $[0.3, 0.4]$ for the second one. On average, what fraction of the training examples will we use to make each prediction?

1%

- (3) [4 pts] Generalize your response for the general case of any dimension d . Argue that a drawback of methods based on nearest neighbors is that, when d is large, there are very few training examples near any test example.

$\lim_{d \rightarrow \infty} (0.1)^d = 0$

- (4) [5 pts] Now suppose that we wish to make a prediction of a test example \mathbf{x} by creating a d -dimensional hypercube centered around \mathbf{x} that contains on average 10% of the training examples. For $d=1, 2, 3$, and 100, what is the length of each side of the hypercube? Explain the implication of your answer on the performance of the nearest neighbors algorithm.

$d = 1$: 0.1, $d = 2$: 0.3, $d = 5$: 0.5, $d = 100$: 0.98. In high dimensions, you end up having to look at all the points.

Q5. [22 pts] Decision Trees and Random Forests

Consider constructing a decision tree on data with d features and n training points where each feature is real-valued and each label takes one of m possible values. The splits are two-way, and are chosen to maximize the information gain. We only consider splits that form a linear boundary parallel to one of the axes. In parts (a), (b) and (c) we will consider a standalone decision tree and not a random forest, so no randomization.

(1) [4 pts] Prove or give a counter-example: For every value of $m > 3$, there exists some probability distribution on m objects such that its entropy is less than -1 . **False. The entropy is always non-negative since $-p \log p$ is non-negative when $p \in [0, 1]$.**

(2) [4 pts] Prove or give a counter-example: In any path from the root split to a leaf, the same feature will never be split on twice.

False. Example: one dimensional feature space with training points of two classes x and o arranged as xxxooooxxx.

(3) [4 pts] Prove or give a counter-example: The information gain at the root is at least as much as the information gain at any other node.

False. Example: the XOR function.

(4) [4 pts] One may be concerned that the randomness introduced in random forests may cause trouble, for instance, some features or samples may not be considered at all. We will investigate this phenomenon in the next two parts.

Consider n training points in a feature space of d dimensions. Consider building a random forest with t binary trees, each having exactly h internal nodes. Let f be the number of features randomly selected at each node. In order to simplify our calculations, we will let $f = 1$. For this setting, compute the probability that a certain feature (say, the first feature) is never considered for splitting.

The probability that it is not considered for splitting in a particular node of a particular tree is $1 - \frac{1}{d}$. The subsampling of $f = 1$ features at each node is independent of all others. There are a total of th nodes and hence the final answer is $(1 - \frac{1}{d})^{th}$.

(5) [3 pts] Now let us investigate the concern regarding the random selection of the samples. Suppose each tree employs n bootstrapped training samples. Compute the probability that a particular sample (say, the first sample) is never considered in any of the trees.

The probability that it is not considered in one of the trees is $(1 - \frac{1}{n})^n$. Since the choice for every tree is independent, the probability that it is not considered in any of the trees is $(1 - \frac{1}{n})^{nt}$.

(6) [3 pts] Compute the values of the probabilities you obtained in the previous two parts for the case when there are $n = 2$ training points, $d = 2$ dimensions, $t = 10$ trees of depth $h = 4$ (you may leave your answer in a fraction and exponentiated form, e.g., as $(\frac{51}{100})^2$). What conclusions can you draw from your answer with regard to the concern mentioned in the beginning of the problem?

$\frac{1}{2^{150}}$ and $\frac{1}{2^{20}}$. It is quite unlikely that a feature or a sample will be missed.

Q6. [12 pts] Elastic net regularization

A powerful method for regularizing linear regression is called elastic net regularization, which combines ridge regression (L2 regularization) and Lasso (L1 regularization).

Observe that linear regression can be probabilistically modeled as $P(y^{(k)}|\mathbf{x}^{(k)}, \mathbf{w}, \sigma^2) \sim \mathcal{N}(\mathbf{w}^T \mathbf{x}^{(k)}, \sigma^2)$. This means $P(y^{(k)}|\mathbf{x}^{(k)}, \mathbf{w}, \sigma^2) =$

$$\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(k)} - \mathbf{w}^T \mathbf{x}^{(k)})^2}{2\sigma^2}\right)$$

It is then possible to show that ridge regression is equivalent to MAP estimation with a Gaussian prior, and Lasso is equivalent to MAP estimation with a Laplace prior.

Let us assume a different prior distribution. Assume each weight w_j is i.i.d, drawn from a distribution such that $P(w_j) = q \exp(-\alpha_1|w_j| - \alpha_2 w_j^2)$, where q, α_1, α_2 are fixed constants. Our training set is $(\mathbf{x}^{(k)}, y^{(k)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})$.

- (1) [6 pts] Show that the MAP estimate for \mathbf{w} is equivalent to minimizing the following risk function, for some choice of constants λ_1, λ_2 :

$$R(\mathbf{w}) = \sum_{k=1}^n (y^{(k)} - \mathbf{w}^T \mathbf{x}^{(k)})^2 + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2$$

The posterior of \mathbf{w} is:

$$P(\mathbf{w}|\mathbf{x}^{(k)}, y^{(k)}) \propto \left(\prod_{k=1}^n \mathcal{N}(y^{(k)}|\mathbf{w}^T \mathbf{x}^{(k)}, \sigma^2)\right) \cdot P(\mathbf{w}) = \left(\prod_{k=1}^n \mathcal{N}(y^{(k)}|\mathbf{w}^T \mathbf{x}^{(k)}, \sigma^2)\right) \cdot \prod_{j=1}^D P(w_j)$$

Taking the log-probability, we want to maximize:

$$\begin{aligned} l(\mathbf{w}) &= \sum_{k=1}^n \log \mathcal{N}(y^{(k)}|\mathbf{w}^T \mathbf{x}^{(k)}, \sigma^2) + \sum_{j=1}^D \log P(w_j) \\ &= \sum_{k=1}^n \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(k)} - \mathbf{w}^T \mathbf{x}^{(k)})^2}{2\sigma^2}\right)\right) + \sum_{j=1}^D \log q \exp(-\alpha_1|w_j| - \alpha_2 w_j^2) \\ &= \sum_{k=1}^n -\frac{(y^{(k)} - \mathbf{w}^T \mathbf{x}^{(k)})^2}{2\sigma^2} - \alpha_1 \sum_{j=1}^D |w_j| - \alpha_2 \sum_{j=1}^D w_j^2 + n \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + D \log(q) \\ &= -\sum_{k=1}^n (y^{(k)} - \mathbf{w}^T \mathbf{x}^{(k)})^2 - 2\sigma^2 \alpha_1 \|\mathbf{w}\|_1 - 2\sigma^2 \alpha_2 \|\mathbf{w}\|_2^2 + n \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) + D \log(q) \end{aligned}$$

This is equivalent to minimizing the following function:

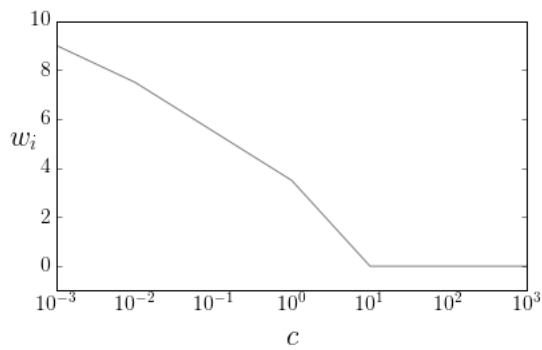
$$R(\mathbf{w}) = \sum_{k=1}^n (y^{(k)} - \mathbf{w}^T \mathbf{x}^{(k)})^2 + \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \|\mathbf{w}\|_2^2$$

where $\lambda_1 = 2\sigma^2 \alpha_1, \lambda_2 = 2\sigma^2 \alpha_2$.

- (2) [2 pts] Suppose we scale both λ_1 and λ_2 by a positive constant c . The graph below represents the value of a single one of the weights, w_i graphed against the value of c . Out of the following set of values for λ_1, λ_2 , which best corresponds to the graph (select exactly one option)?

$\lambda_1 = 1, \lambda_2 = 0$

$\lambda_1 = 0, \lambda_2 = 1$



- (3) [2 pts] Explain why your choice in (b) results in the graph.

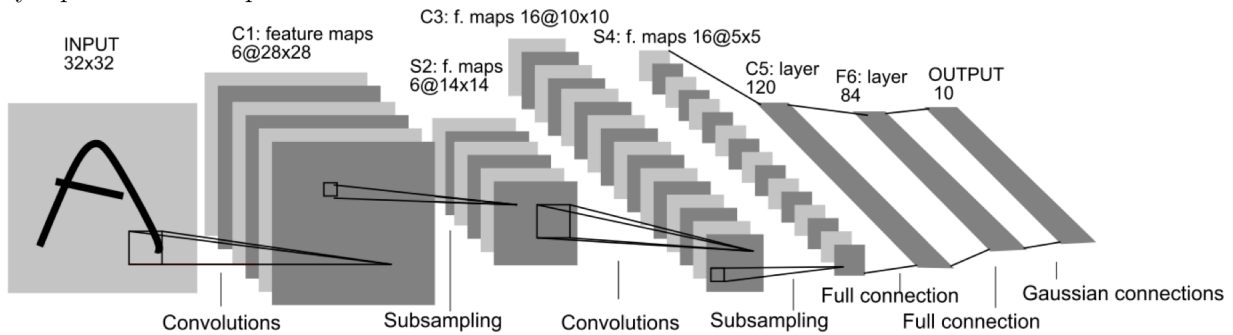
That choice is equivalent to just doing Lasso, which induces sparsity.

- (4) [2 pts] What is the advantage of using Elastic net regularization over using a single regularization term of $\|\mathbf{w}\|_p$, where $1 < p < 2$?

Elastic net gives us the option of inducing sparsity.

Q7. [14 pts] Neural Networks

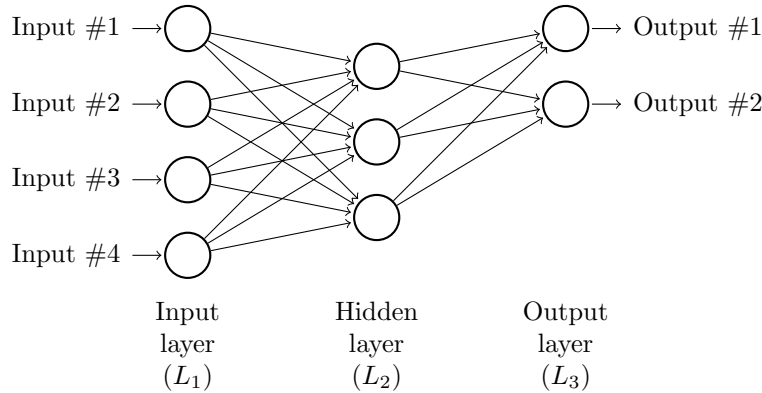
- (1) [6 pts] Here is the historical *LeNet* Convolutional Neural Network architecture of Yann LeCun et al. for digit classification that we've discussed in class. Here, the INPUT layer takes in a 32x32 image, and the OUTPUT layer produces 10 outputs. The notation 6@28x28 means 6 matrices of size 28x28.



If the parameters of a given layer are the weights that connect to its inputs,

- Given that the input size is 32x32, and the Layer 1 size is 28x28, what's the size of the convolutional filter in the first layer (i.e. how many inputs is each neuron connected to)? 5×5
- How many independent parameters (weight and bias) are in layer C1? $5 \times 5 \times 6 \times 1 + 6 = 156$
- How many independent parameters (weight and bias) are in layer C3? $5 \times 5 \times 16 \times 6 + 16 = 2416$
- How many independent parameters (weight and bias) are in layer F6? $120 \times 84 + 84 = 10164$

- (2) [8 pts] Consider a three layer fully-connected network with n_1, n_2, n_3 neurons in three layers respectively. Inputs are fed into the first layer. The loss is mean squared error E , and the non-linearity is a sigmoid function. Let the label vector be t of size n_3 . Let each layer output vector be y_i and input vector be z_i , both of size n_i . Let the weight between layer i and layer $i + 1$ be W_{ii+1} . The j -th element in y_i is defined by y_i^j , same for z_i^j . The weight connecting k -th and l -th neuron in $i, i + 1$ layers is defined by W_{ii+1}^{kl} (You don't need to consider bias in this problem).



Here is a summary of our notation:

- σ denotes the activation function for L_2 and L_3 , $\sigma(x) = \frac{1}{1+e^{-x}}$. There is no activation applied to the input layer.
- $z_i^{(j)} = \sum_{k=1}^P W_{i-1i}^{kj} x_{i-1}^{(k)}$
- $y_i^{(j)} = \sigma\left(\sum_{k=1}^P W_{i-1i}^{kj} x_{i-1}^{(k)}\right)$

Now solve the following problems.

- Find $\frac{\partial E}{\partial z_3^j}$ in terms of y_3^j .
 $-2y_3^j(1 - y_3^j)(t^j - y_3^j)$
- Find $\frac{\partial E}{\partial y_2^k}$ in terms of elements in W_{23} and $\frac{\partial E}{\partial z_3^j}$.
 $\sum_{j=1}^{n_3} W_{23}^{kj} \frac{\partial E}{\partial z_3^j}$
- Find $\frac{\partial E}{\partial W_{23}^{kj}}$ in terms of y_2^k , y_3^j and t^j .
 $y_2^k \frac{\partial E}{\partial z_3^j} = -2y_3^j(1 - y_3^j)(t^j - y_3^j)y_2^k$
- If the input to a neuron in max-pooling layer is x and the output is $y = \max(x)$, derive $\frac{\partial y}{\partial x_i}$.
 $\frac{\partial y}{\partial x_i} = 1$ if and only if $x_i = \max(x)$, otherwise $\frac{\partial y}{\partial x_i} = 0$.

Q8. [16 pts] The dimensions are high! And possibly hard too.

In this problem, we will derive a famous result called the “Johnson-Lindenstrauss” lemma. Suppose you are given n arbitrary vectors $x^1, \dots, x^n \in \mathbb{R}^{d \times 1}$. Let $k = 320 \log n$. Now consider a matrix $A \in \mathbb{R}^{k \times d}$ that is obtained randomly in the following manner: every entry of the matrix is chosen independently at random from $\mathcal{N}(0, 1)$. Define vectors $z^1, \dots, z^n \in \mathbb{R}^{k \times 1}$ as $z^i = \frac{1}{\sqrt{k}} Ax^i$ for every $i \in \{1, \dots, n\}$.

- (1) [4 pts] For any given $i \in \{1, \dots, n\}$, what is the distribution of the random vector Ax^i ? Your answer should be in terms of the vector x^i . To simplify notation, let $v = x^i$. Clearly, Av is a zero-mean jointly Gaussian vector. Let us compute the covariance: $E[(Av)(Av)^T]$. Letting a_j^T denote the j^{th} row of A , we have that the j^{th} entry of vector Av is $a_j^T v$, and hence the $(i, j)^{\text{th}}$ entry of $(Av)(Av)^T$ is $(a_i^T v v^T a_j)$. It follows that the $(i, j)^{\text{th}}$ entry of $E[(Av)(Av)^T]$ is $E[a_i^T v v^T a_j] = E[v^T a_i a_j^T v] = v^T E[a_i a_j^T] v$. Now we have $E[a_i a_j^T] = I$ if $i = j$ and 0 otherwise. Thus the covariance matrix is a diagonal matrix with each entry on the diagonal equal to $\|x^i\|_2^2$.

- (2) [4 pts] For any distinct $i, j \in \{1, \dots, n\}$, derive a relation between $\mathbb{E}[\|A(x^i - x^j)\|_2^2]$ and the value of $\|x^i - x^j\|_2^2$? More points for deriving the relation using your answer from part (1) above. Without using part 1: $\mathbb{E}[\|A(x^i - x^j)\|_2^2] = \mathbb{E}[(x^i - x^j)^T A^T A (x^i - x^j)] = (x^i - x^j)^T \mathbb{E}[A^T A] (x^i - x^j)$. $\mathbb{E}[A^T A] = kI$ and hence the answer is $k\|x^i - x^j\|_2^2$.

Using part 1: Now let $v = x^i - x^j$. Observe that $\mathbb{E}[\|Av\|_2^2]$ is simply the sum of the variances of each entry of the vector Av . We computed these variances in part (1) as being equal to $\|v\|_2^2$. Since vector Av has length k , the sum of the variances equals $k\|v\|_2^2$.

- (3) [4 pts] It can be shown that for any fixed vector v , the random matrix A has the property that

$$\frac{3}{4}\|v\|_2^2 \leq \|Av\|_2^2 \leq \frac{5}{4}\|v\|_2^2$$

with probability at least $1 - \frac{1}{n^4}$. Using this fact, show that with probability at least $1 - \frac{1}{n^2}$, every pair (z^i, z^j) simultaneously satisfies $\frac{3}{4}\|x^i - x^j\|_2^2 \leq \|z^i - z^j\|_2^2 \leq \frac{5}{4}\|x^i - x^j\|_2^2$. (Think of how you would bound probabilities of multiple events. Only requires a very basic fact about probability and a little thought.) Use the simple fact that: $P(A \text{ or } B) \leq P(A) + P(B)$.

- (4) [4 pts] Describe, in at most two sentences, the usefulness of this result. (Think of n and d as having very large values, for instance, several billions). Helps in reducing the dimensionality of the feature space in problems where only the pairwise distances need to be preserved.

- You have 3 hours for the exam.
- The exam is closed book, closed notes except your one-page (two sides) or two-page (one side) crib sheet.
- Please use non-programmable calculators only.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.
- For true/false questions, fill in the *True/False* bubble.
- For multiple-choice questions, fill in the bubbles for **ALL CORRECT CHOICES** (in some cases, there may be more than one). For a question with p points and k choices, every false positive will incur a penalty of $p/(k - 1)$ points.
- For short answer questions, **unnecessarily long explanations and extraneous data will be penalized**. Please try to be terse and precise and do the side calculations on the scratch papers provided.
- Please **draw a bounding box around your answer** in the Short Answers section. A missed answer without a bounding box will not be regraded.

First name	
Last name	
SID	

For staff use only:

Q1.	True/False	/23
Q2.	Multiple Choice Questions	/36
Q3.	Short Answers	/26
	Total	/85

Q1. [23 pts] True/False

- (a) [1 pt] Solving a non linear separation problem with a hard margin Kernelized SVM (Gaussian RBF Kernel) might lead to overfitting.
 True False
- (b) [1 pt] In SVMs, the sum of the Lagrange multipliers corresponding to the positive examples is equal to the sum of the Lagrange multipliers corresponding to the negative examples.
 True False
- (c) [1 pt] SVMs directly give us the posterior probabilities $P(y = 1|x)$ and $P(y = -1|x)$.
 True False
- (d) [1 pt] $V(X) = E[X]^2 - E[X^2]$
 True False
- (e) [1 pt] In the discriminative approach to solving classification problems, we model the conditional probability of the labels given the observations.
 True False
- (f) [1 pt] In a two class classification problem, a point on the Bayes optimal decision boundary x^* always satisfies $P(y = 1|x^*) = P(y = 0|x^*)$.
 True False
- (g) [1 pt] Any linear combination of the components of a multivariate Gaussian is a univariate Gaussian.
 True False
- (h) [1 pt] For any two random variables $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$, $X + Y \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.
 True False
- (i) [1 pt] Stanford and Berkeley students are trying to solve the same logistic regression problem for a dataset. The Stanford group claims that their initialization point will lead to a much better optimum than Berkeley's initialization point. Stanford is correct.
 True False
- (j) [1 pt] In logistic regression, we model the odds ratio ($\frac{p}{1-p}$) as a linear function.
 True False
- (k) [1 pt] Random forests can be used to classify infinite dimensional data.
 True False
- (l) [1 pt] In boosting we start with a Gaussian weight distribution over the training samples.
 True False
- (m) [1 pt] In Adaboost, the error of each hypothesis is calculated by the ratio of misclassified examples to the total number of examples.
 True False
- (n) [1 pt] When $k = 1$ and $N \rightarrow \infty$, the kNN classification rate is bounded above by twice the Bayes error rate.
 True False
- (o) [1 pt] A single layer neural network with a sigmoid activation for binary classification with the cross entropy loss is exactly equivalent to logistic regression.
 True False

- (p) [1 pt] The loss function for LeNet5 (the convolutional neural network by LeCun et al.) is convex.
 True False
- (q) [1 pt] Convolution is a linear operation i.e. $(\alpha f_1 + \beta f_2) * g = \alpha f_1 * g + \beta f_2 * g$.
 True False
- (r) [1 pt] The k-means algorithm does coordinate descent on a non-convex objective function.
 True False
- (s) [1 pt] A 1-NN classifier has higher variance than a 3-NN classifier.
 True False
- (t) [1 pt] The single link agglomerative clustering algorithm groups two clusters on the basis of the maximum distance between points in the two clusters.
 True False
- (u) [1 pt] The largest eigenvector of the covariance matrix is the direction of minimum variance in the data.
 True False
- (v) [1 pt] The eigenvectors of AA^T and $A^T A$ are the same.
 True False
- (w) [1 pt] The non-zero eigenvalues of AA^T and $A^T A$ are the same.
 True False

Q2. [36 pts] Multiple Choice Questions

(a) [4 pts] In linear regression, we model $P(y|x) \sim \mathcal{N}(w^T x + w_0, \sigma^2)$. The irreducible error in this model is _____.

- σ^2
 $E[(y - E[y|x])|x]$
 $E[(y - E[y|x])^2|x]$
 $E[y|x]$

(b) [4 pts] Let S_1 and S_2 be the set of support vectors and w_1 and w_2 be the learnt weight vectors for a linearly separable problem using hard and soft margin linear SVMs respectively. Which of the following are correct?

- $S_1 \subset S_2$
 S_1 may not be a subset of S_2
 $w_1 = w_2$
 w_1 may not be equal to w_2 .

(c) [4 pts] Ordinary least-squares regression is equivalent to assuming that each data point is generated according to a linear function of the input plus zero-mean, constant-variance Gaussian noise. In many systems, however, the noise variance is itself a positive linear function of the input (which is assumed to be non-negative, i.e., $x \geq 0$). Which of the following families of probability models correctly describes this situation in the univariate case?

- $P(y|x) = \frac{1}{\sigma\sqrt{2\pi x}} \exp\left(-\frac{(y-(w_0+w_1x))^2}{2x\sigma^2}\right)$
 $P(y|x) = \frac{1}{\sigma\sqrt{2\pi x}} \exp\left(-\frac{(y-(w_0+(w_1+\sigma^2)x))^2}{2\sigma^2}\right)$
 $P(y|x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-(w_0+w_1x))^2}{2\sigma^2}\right)$
 $P(y|x) = \frac{1}{\sigma x\sqrt{2\pi}} \exp\left(-\frac{(y-(w_0+w_1x))^2}{2x^2\sigma^2}\right)$

(d) [3 pts] The left singular vectors of a matrix A can be found in _____.

- Eigenvectors of AA^T
 Eigenvectors of A^2
 Eigenvectors of $A^T A$
 Eigenvalues of AA^T

(e) [3 pts] Averaging the output of multiple decision trees helps _____.

- Increase bias
 Increase variance
 Decrease bias
 Decrease variance

(f) [4 pts] Let A be a symmetric matrix and S be the matrix containing its eigenvectors as column vectors, and D a diagonal matrix containing the corresponding eigenvalues on the diagonal. Which of the following are true:

- $AS = SD$
 $SA = DS$
 $AS = DS$
 $AS = DS^T$

(g) [4 pts] Consider the following dataset: $A = (0, 2)$, $B = (0, 1)$ and $C = (1, 0)$. The k-means algorithm is initialized with centers at A and B . Upon convergence, the two centers will be at

- A and C
 C and the midpoint of AB
 A and the midpoint of BC
 A and B

(h) [3 pts] Which of the following loss functions are convex?

Misclassification loss

Hinge loss

Logistic loss

Exponential Loss ($e^{-yf(x)}$)

(i) [3 pts] Consider T_1 , a decision stump (tree of depth 2) and T_2 , a decision tree that is grown till a maximum depth of 4. Which of the following is/are correct?

$Bias(T_1) < Bias(T_2)$

$Variance(T_1) < Variance(T_2)$

$Bias(T_1) > Bias(T_2)$

$Variance(T_1) > Variance(T_2)$

(j) [4 pts] Consider the problem of building decision trees with k -ary splits (split one node into k nodes) and you are deciding k for each node by calculating the entropy impurity for different values of k and optimizing simultaneously over the splitting threshold(s) and k . Which of the following is/are true?

The algorithm will always choose $k = 2$

There will be $k - 1$ thresholds for a k -ary split

The algorithm will prefer high values of k

This model is strictly more powerful than a binary decision tree.

Q3. [26 pts] Short Answers

- (a) [5 pts] Given that (x_1, x_2) are jointly normally distributed with $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$ ($\sigma_{21} = \sigma_{12}$), give an expression for the mean of the conditional distribution $p(x_1|x_2 = a)$.

This can be solved by writing $p(x_1|x_2 = a) = \frac{p(x_1, x_2 = a)}{p(x_2 = a)}$. x_2 being a component of a multivariate Gaussian is a univariate Gaussian with $x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$. Write out the Gaussian densities and simplify (complete squares) to see the following:

$$x_1|x_2 = a \sim \mathcal{N}(\bar{\mu}, \bar{\sigma}^2), \quad \bar{\mu} = \mu_1 + \frac{\sigma_{12}}{\sigma_2^2}(a - \mu_2)$$

- (b) [4 pts] The logistic function is given by $\sigma(x) = \frac{1}{1+e^{-x}}$. Show that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} \cdot \frac{e^{-x}}{(1+e^{-x})} = \left(\frac{1}{1+e^{-x}} \right) \left(1 - \frac{1}{1+e^{-x}} \right) = \sigma(x)(1 - \sigma(x))$$

- (c) Let X have a uniform distribution

$$p(x; \theta) = \begin{cases} \frac{1}{\theta} & 0 \leq x \leq \theta \\ 0 & \text{otherwise} \end{cases}$$

Suppose that n samples x_1, \dots, x_n are drawn independently according to $p(x; \theta)$.

- (i) [5 pts] The maximum likelihood estimate of θ is $x_{(n)} = \max(x_1, x_2, \dots, x_n)$. Show that this estimate of θ is biased.

Biased estimator: $\hat{\theta}$ (the sample estimate) is a biased estimator of θ (the population distribution parameter) if $E[\hat{\theta}] \neq \theta$.

Here $\hat{\theta} = x_{(n)}$. And $E[x_{(n)}] = \frac{n}{n+1}\theta \neq \theta$. The steps for finding $E[x_{(n)}]$ are given in the solutions of Homework 2, problem 5(c).

- (ii) [2 pts] Give an expression for an unbiased estimator of θ .

$$\hat{\theta}_{unbiased} = \frac{n+1}{n}x_{(n)}$$

$$E[\hat{\theta}_{unbiased}] = E\left[\frac{n+1}{n}x_{(n)}\right] = \frac{n+1}{n}E[x_{(n)}] = \frac{n+1}{n} \times \frac{n}{n+1}\theta = \theta$$

- (d) [5 pts] Consider the problem of fitting the following function to a dataset of 100 points $\{(x_i, y_i)\}, i = 1 \dots 100$:

$$y = \alpha \cos(x) + \beta \sin(x) + \gamma$$

This problem can be solved using the least squares method with a solution of the form:

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = (X^T X)^{-1} X^T Y$$

What are X and Y ?

$$X = \begin{bmatrix} \cos(x_1) & \sin(x_1) & 1 \\ \cos(x_2) & \sin(x_2) & 1 \\ \vdots & \vdots & \vdots \\ \cos(x_{100}) & \sin(x_{100}) & 1 \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{100} \end{bmatrix}$$

- (e) [5 pts] Consider the problem of binary classification using the Naive Bayes classifier. You are given two dimensional features (X_1, X_2) and the categorical class conditional distributions in the tables below. The entries in the tables correspond to $P(X_1 = x_1 | C_i)$ and $P(X_2 = x_2 | C_i)$ respectively. The two classes are *equally likely*.

$X_1 =$ \backslash Class	C_1	C_2
-1	0.2	0.3
0	0.4	0.6
1	0.4	0.1

$X_2 =$ \backslash Class	C_1	C_2
-1	0.4	0.1
0	0.5	0.3
1	0.1	0.6

Given a data point $(-1, 1)$, calculate the following posterior probabilities:

$$P(C_1 | X_1 = -1, X_2 = 1) = \text{Using Bayes' Rule and conditional independence assumption of Naive Bayes}$$

$$\frac{P(X_1=-1, X_2=1 | C_1) P(C_1)}{P(X_1=-1, X_2=1)} = \frac{P(X_1=-1 | C_1) P(X_2=1 | C_1) P(C_1)}{P(X_1=-1 | C_1) P(X_2=1 | C_1) P(C_1) + P(X_1=-1 | C_2) P(X_2=1 | C_2) P(C_2)} = 0.1$$

$$P(C_2 | X_1 = -1, X_2 = 1) = 1 - P(C_1 | X_2 = -1, X_1 = 1) = 0.9$$

SCRATCH PAPER

SCRATCH PAPER

- You have 3 hours for the exam.
- The exam is closed book, closed notes except your one-page crib sheet.
- Please use non-programmable calculators only.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation.
- For true/false questions, fill in the *True/False* bubble.
- For multiple-choice questions, fill in the bubbles for **ALL CORRECT CHOICES** (in some cases, there may be more than one). We have introduced a negative penalty for false positives for the multiple choice questions such that the expected value of randomly guessing is 0. Don't worry, for this section, your score will be the maximum of your score and 0, thus you cannot incur a negative score for this section.

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

For staff use only:

Q1. True or False	/9
Q2. Multiple Choice	/24
Q3. Softmax regression	/11
Q4. PCA and least squares	/10
Q5. Mixture of linear regressions	/10
Q6. Training set augmentation	/10
Q7. Kernel PCA	/12
Q8. Autoencoder	/14
Total	/100

Q1. [9 pts] True or False

- (a) [1 pt] The singular value decomposition of a real matrix is unique.
 True False
- (b) [1 pt] A multiple-layer neural network with linear activation functions is equivalent to one single-layer perceptron that uses the same error function on the output layer and has the same number of inputs.
 True False
- (c) [1 pt] The maximum likelihood estimator for the parameter θ of a uniform distribution over $[0, \theta]$ is unbiased.
 True False
- (d) [1 pt] The k-means algorithm for clustering is guaranteed to converge to a local optimum.
 True False
- (e) [1 pt] Increasing the depth of a decision tree cannot increase its training error.
 True False
- (f) [1 pt] There exists a one-to-one feature mapping ϕ for every valid kernel k .
 True False
- (g) [1 pt] For high-dimensional data, k-d trees can be slower than brute force nearest neighbor search.
 True False
- (h) [1 pt] If we had infinite data and infinitely fast computers, kNN would be the only algorithm we would study in CS 189.
 True False
- (i) [1 pt] For datasets with high label noise (many data points with incorrect labels), random forests would generally perform better than boosted decision trees.
 True False

Q2. [24 pts] Multiple Choice

(a) [2 pts] In Homework 4, you fit a logistic regression model on spam and ham data for a Kaggle Competition. Assume you had a very good score on the public test set, but when the GSIs ran your model on a private test set, your score dropped a lot. This is likely because you overfitted by submitting multiple times and changing the following between submissions:

- λ , your penalty term
- ϵ , your convergence criterion
- η , your step size
- Fixing a random bug

(b) [2 pts] Given d -dimensional data $\{\mathbf{x}_i\}_{i=1}^N$, you run principle component analysis and pick P principle components. Can you always reconstruct any data point \mathbf{x}_i for $i \in \{1 \dots N\}$ from the P principle components with zero reconstruction error?

- Yes, if $P < d$
- Yes, if $P = d$
- Yes, if $P < n$
- No, always

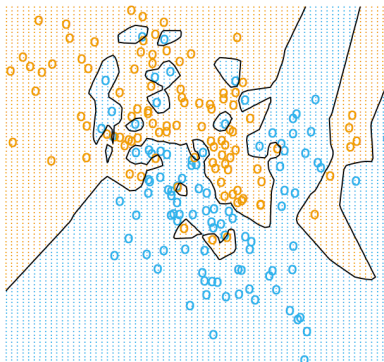
(c) [2 pts] Putting a standard Gaussian prior on the weights for linear regression ($\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) will result in what type of posterior distribution on the weights?

- Laplace
- Uniform
- Poisson
- None of the above

(d) [2 pts] Suppose we have N instances of d -dimensional data. Let h be the amount of data storage necessary for a histogram with a fixed number of ticks per axis, and let k be the amount of data storage necessary for kernel density estimation. Which of the following is true about h and k ?

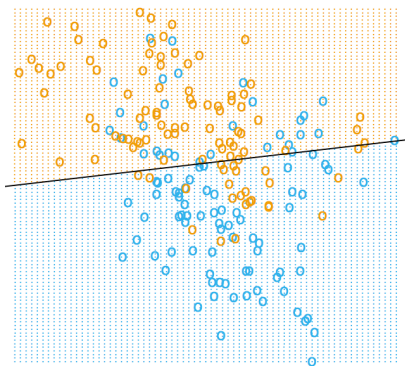
- h and k grow linearly with N
- h grows exponentially with d , and k grows linearly N
- h and k grow exponentially with d
- h grows linearly with N , and k grows exponentially with d

(e) [2 pts] Which of these classifiers could have generated this decision boundary?



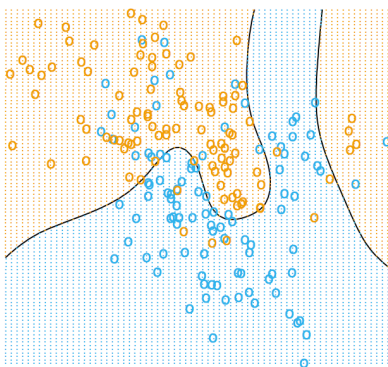
- Linear SVM
- 1-NN
- Logistic regression
- None of the above

(f) [2 pts] Which of the these classifiers could have generated this decision boundary?



- Linear SVM
- 1-NN
- Logistic regression
- None of the above

(g) [2 pts] Which of the these classifiers could have generated this decision boundary?



- Linear SVM
- 1-NN
- Logistic regression
- None of the above

(h) [2 pts] You want to cluster this data into 2 clusters. Which of the these algorithms would work well?



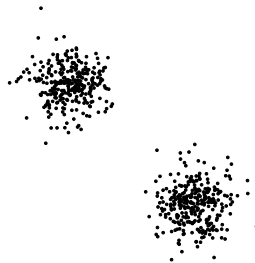
- K-means
- GMM clustering
- Mean shift clustering

(i) [2 pts] You want to cluster this data into 2 clusters. Which of these algorithms would work well?



- K-means GMM clustering Mean shift clustering

(j) [2 pts] You want to cluster this data into 2 clusters. Which of these algorithms would work well?



- K-means GMM clustering Mean shift clustering

The following questions are about how to help CS 189 TA Jonathan Snow to solve the homework.

(k) [2 pts] Jonathan just trained a decision tree for a digit recognition. He notices an extremely low training error, but an abnormally large test error. He also notices that an SVM with a linear kernel performs much better than his tree. What could be the cause of his problem?

- Decision tree is too deep Decision tree is overfitting
 Learning rate too high There is too much training data

(l) [2 pts] Jonathan has now switched to multilayer neural networks and notices that the training error is going down and converges to a local minimum. Then when he tests on the new data, the test error is abnormally high. What is probably going wrong and what do you recommend him to do?

- The training data size is not large enough. Collect a larger training data and retrain it. Play with learning rate and add regularization term to the objective function.
 Use a different initialization and train the network several times. Use the average of predictions from all nets to predict test data. Use the same training data but add two more hidden layers.

Q3. [11 pts] Softmax regression

Recall the setup of logistic regression: We assume that the posterior probability is of the form

$$p(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\beta}^T \mathbf{x}}}$$

This assumes that $Y|\mathbf{X}$ is a Bernoulli random variable. We now turn to the case where $Y|\mathbf{X}$ is a multinomial random variable over K outcomes. This is called softmax regression, because the posterior probability is of the form

$$p(Y = k|\mathbf{x}) = \mu_k(\mathbf{x}) = \frac{e^{\boldsymbol{\beta}_k^T \mathbf{x}}}{\sum_{j=1}^K e^{\boldsymbol{\beta}_j^T \mathbf{x}}}$$

which is called the softmax function. Assume we have observed data $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$. Our goal is to learn the weight vectors $\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K$.

(a) [3 pts] Find the negative log likelihood of the data $l(\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K)$.

$$\begin{aligned} -\log \mathbb{P}(Y|X) &= -\log \prod_{i=1}^N \mathbb{P}(y_i|x_i) = -\log \prod_{i=1}^N \prod_{k=1}^K \left(\frac{e^{\boldsymbol{\beta}_k^T x_i}}{\sum_{j=1}^K e^{\boldsymbol{\beta}_j^T x_i}} \right)^{1\{y_i=k\}} \\ &= -\sum_{i=1}^N \sum_{k=1}^K 1\{y_i = k\} \left(\boldsymbol{\beta}_k^T x_i - \log \left(\sum_{j=1}^K e^{\boldsymbol{\beta}_j^T x_i} \right) \right) \\ &= -\sum_{i=1}^N \sum_{k=1}^K 1\{y_i = k\} \boldsymbol{\beta}_k^T x_i + \sum_{i=1}^N \log \left(\sum_{j=1}^K e^{\boldsymbol{\beta}_j^T x_i} \right) \end{aligned}$$

(b) [2 pts] We want to minimize the negative log likelihood. To combat overfitting, we put a regularizer on the objective function. Find the gradient w.r.t. $\boldsymbol{\beta}_k$ of the regularized objective

$$l(\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K) + \lambda \sum_{k=1}^K \|\boldsymbol{\beta}_k\|^2$$

$$\nabla_{\boldsymbol{\beta}_k} -\log \mathbb{P}(Y|X) = 2\lambda \boldsymbol{\beta}_k - \sum_{i=1}^N 1\{y_i = k\} x_i + \sum_{i=1}^N \frac{e^{\boldsymbol{\beta}_k^T x_i}}{\sum_{j=1}^K e^{\boldsymbol{\beta}_j^T x_i}} x_i$$

Note that we can use the definition of $\mu_k(x_i)$ here to save a bunch of writing.

$$= 2\lambda \boldsymbol{\beta}_k + \sum_{i=1}^N (\mu_k(x_i) - 1\{y_i = k\}) x_i$$

(c) [4 pts] State the gradient updates for both batch gradient descent and stochastic gradient descent.

Batch gradient descent:

$$\boldsymbol{\beta}_k^{(t+1)} = \boldsymbol{\beta}_k^{(t)} - \eta \left(2\lambda \boldsymbol{\beta}_k^{(t)} + \sum_{i=1}^N (\mu_k(x_i) - 1\{y_i = k\}) x_i \right)$$

Stochastic gradient descent:

$$\boldsymbol{\beta}_k^{(t+1)} = \boldsymbol{\beta}_k^{(t)} - \eta \left(2\lambda \boldsymbol{\beta}_k^{(t)} + (\mu_k(x_i) - 1\{y_i = k\}) x_i \right)$$

(d) [2 pts] There are times when we'd like to consider the multiclass case to be a 1-vs.-all scenario with K binary classifiers, and there are times when we'd like to attack the multiclass case with a multiclass classifier such as softmax regression.

When would you want to use a softmax regression as opposed to K 1-vs.-all logistic regressions?

- When the classes are mutually exclusive
- When the classes are not linearly separable
- When the classes are not mutually exclusive
- Both work equally well

Q4. [10 pts] PCA and least squares

Recall that PCA transforms (zero-mean) data into low-dimensional reconstructions that lie in the span of the top k eigenvectors of the sample covariance matrix. Let \mathbf{U}_k denote the $d \times k$ matrix of the top k eigenvectors of the covariance matrix (\mathbf{U}_k is a truncated version of \mathbf{U} , which is the matrix of eigenvectors of the covariance matrix).

There are two approaches to computing the low-dimensional reconstruction $\mathbf{w} \in \mathbb{R}^k$ of a data point $\mathbf{x} \in \mathbb{R}^d$:

1. Solve a least squares problem to minimize the reconstruction error
2. Project \mathbf{x} onto the span of the columns of \mathbf{U}_k

In this problem, you will show that these approaches are equivalent.

- (a) [5 pts] Formulate the least squares problem in terms of \mathbf{U}_k , \mathbf{x} , and the variable \mathbf{w} .
(Hint: This optimization problem should resemble linear regression.)

We want to find the weights such that when we weight the columns of U_k , we will minimize the residual error. Thus, the objective function is $\|U_k w - x_i\|^2$. Here is our least squares problem:

$$\min_w \|U_k w - x_i\|^2$$

- (b) [5 pts] Show that the solution of the least squares problem is equal to $\mathbf{U}_k^\top \mathbf{x}$, which is the projection of \mathbf{x} onto the span of the columns of \mathbf{U}_k .

Recall the normal equations (which is the most important equation in the class!). For some unconstrained least squares problem in the form

$$\min_x \|Ax - y\|^2$$

the solution of the minimizer is

$$x^* = (A^T A)^{-1} A^T y$$

only when A is full rank. This applies here since U_k is full rank by definition. Plugging in our least squares problem, we have

$$w^* = (U_k^T U_k)^{-1} U_k^T x_i$$

If you don't remember this you can easily derive this by taking the gradient of the objective function and setting it to 0. We note that $U_k^T U_k = I_k$, (I_k = the identity matrix in k dimensions) thus

$$w^* = (U_k^T U_k)^{-1} U_k^T x_i = (I_k)^{-1} U_k^T x_i = U_k^T x_i$$

since the inverse of I is simply I .

Q5. [10 pts] Mixture of linear regressions

In class, you learned that K -means partitions points into K clusters by minimizing an objective that encourages points to be close to their cluster centers. K -means minimizes this objective in a coordinate descent fashion, alternating between computing cluster assignments and cluster centers until convergence.

In this problem, you will devise an algorithm, in the spirit of K -means, that fits an entirely different kind of mixture model. You are given a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. You know that this dataset is a mixture of realizations of K different linear regression models

$$y = \mathbf{w}_k^\top \mathbf{x} + \mathcal{N}(0, \sigma^2)$$

parameterized by K weight vectors $\mathbf{w}_1, \dots, \mathbf{w}_K \in \mathbb{R}^d$.

Your algorithm will jointly determine the following:

- A partition S_1, \dots, S_K of the dataset such that $(\mathbf{x}_i, y_i) \in S_k$ iff (\mathbf{x}_i, y_i) comes from model k
 - The model weights $\mathbf{w}_1, \dots, \mathbf{w}_K$
- (a) [4 pts] Write an objective function $f(S_1, \dots, S_K, \mathbf{w}_1, \dots, \mathbf{w}_K)$ to be minimized to solve this problem. Use the penalty $\|\mathbf{w}_k^\top \mathbf{x} - y\|^2$ if the point (\mathbf{x}, y) is assigned to model k . Your objective should be a sum of N terms, and each data point should show up in exactly one of these terms.

$$f(S_1, \dots, S_K, w_1, \dots, w_K) = \sum_{k=1}^K \sum_{(x,y) \in S_k} \|w_k^\top x - y\|^2$$

- (b) [3 pts] What is coordinate descent update for f with $\mathbf{w}_1, \dots, \mathbf{w}_K$ fixed? In other words, to which of the K models should a point (\mathbf{x}, y) be assigned?
- Assign the point (x, y) to S_k if $k = \arg \min_k \|w_k^\top x - y\|^2$.
- (c) [3 pts] Write the coordinate descent update for f with S_1, \dots, S_K fixed.

For a set S of (\mathbf{x}, y) values, you should use the notation \mathbf{X}_S to denote the design matrix whose rows are the \mathbf{x} -values of the elements of S , and \mathbf{y}_S to denote the column vector of y -values of the elements of S .

$$w_k = (X_{S_k}^\top X_{S_k})^{-1} X_{S_k}^\top y_{S_k}$$

Q6. [10 pts] Training set augmentation

In class, you learned that one way to encourage invariance of a model to certain transformations is to augment the training set with extra examples perturbed according to those transformations. In this problem, you will examine the behavior of a certain type of input perturbation for a probabilistic linear regression setting.

Consider the following general generative model for regression:

- $\mathbf{x} \sim p(\mathbf{x})$, where $p(\mathbf{x})$ is a distribution over input vectors $\mathbf{x} \in \mathbb{R}^d$
- $y|\mathbf{x} \sim p(y|\mathbf{x})$, where $p(y|\mathbf{x})$ is a distribution over output scalars $y \in \mathbb{R}$ given \mathbf{x}

Assume that the relationship between y and \mathbf{x} is well-modeled by a linear function $y = \mathbf{w}^\top \mathbf{x}$, where $\mathbf{w} \in \mathbb{R}^d$, so that in the infinite dataset limit, the objective to be minimized for this regression problem is:

$$\mathcal{L}_0(\mathbf{w}) = \mathbb{E} [(\mathbf{w}^\top \mathbf{x} - y)^2]$$

Now suppose the inputs are perturbed by zero-mean Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \lambda \mathbf{I})$, which is independent of the training data. The new objective is

$$\mathcal{L}(\mathbf{w}) = \mathbb{E} [(\mathbf{w}^\top (\mathbf{x} + \epsilon) - y)^2]$$

- (a) [9 pts] Compute and simplify $\mathcal{L}(\mathbf{w})$. Show all your work in detail, and write your answer in terms of \mathcal{L}_0 .

$$\begin{aligned} \mathcal{L}(w) &= \mathbb{E} [(w^\top (x + \epsilon) - y)^2] \\ &= \mathbb{E} [((w^\top x - y) + w^\top \epsilon)^2] \\ &= \mathbb{E} [(w^\top x - y)^2 + 2(w^\top x - y)w^\top \epsilon + w^\top \epsilon \epsilon^\top w] \\ &= \mathbb{E} [(w^\top x - y)^2] + \mathbb{E} [2(w^\top x - y)w^\top] \mathbb{E}[\epsilon] + w^\top \mathbb{E} [\epsilon \epsilon^\top] w \end{aligned}$$

Substituting $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon \epsilon^\top] = \lambda I$, this simplifies to

$$= \mathcal{L}_0(w) + \lambda \|w\|^2$$

- (b) [1 pt] Is there a relationship between this particular type of input perturbation and some type of regularization? If so, what kind of regularizer is involved?

In this setting, regression assuming this type of input perturbation turns out to be equivalent to regression with an L_2 regularizer.

Q7. [12 pts] Kernel PCA

You are given d -dimensional real-valued data $\{\mathbf{x}_i\}_{i=1}^N$ and a feature mapping $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^m$. In the following questions, you will investigate how to do PCA in feature space on the feature vectors $\{\phi(\mathbf{x}_i)\}_{i=1}^N$. Assume that the data is centered in feature space; that is, $\sum_{i=1}^N \phi(\mathbf{x}_i) = 0$.

In the following, Φ is a design matrix whose i^{th} row is $\phi(\mathbf{x}_i)$.

- (a) [1 pt] Recall that as a part of PCA, we must solve the eigenvalue problem $\mathbf{S}\mathbf{v} = \lambda\mathbf{v}$, where \mathbf{S} is proportional to the sample covariance matrix. For PCA in feature space, we have $\mathbf{S} = \sum_{i=1}^N \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top = \Phi^\top\Phi$. Why is this a problem if m is large?

Working in feature space directly is too expensive if m is large. The covariance matrix $\Phi^\top\Phi$ is $m \times m$, which is too large to compute and work with.

- (b) [3 pts] Now, you are given a kernel function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$. Define the kernel matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Show that if $\lambda \neq 0$, then λ is an eigenvalue of \mathbf{S} if and only if λ is also an eigenvalue of \mathbf{K} (in other words, finding feature-space principal components can be done by finding eigenvectors of \mathbf{K}).

Notice that $K = \Phi\Phi^\top$. The nonzero eigenvalues of $K = \Phi\Phi^\top$ and $S = \Phi^\top\Phi$ are the same.

- (c) [4 pts] Let \mathbf{v} be an eigenvector of \mathbf{S} with nonzero eigenvalue λ . Show that \mathbf{v} can be written as $\mathbf{v} = \Phi^\top\alpha_v$, where α_v is an eigenvector of \mathbf{K} with eigenvalue λ .

First, write $Sv = \Phi^\top\Phi v = \lambda v$. Multiplying this equation by Φ gives $K\alpha = \lambda\alpha$, where $\alpha = \Phi v$. Then, since $\Phi^\top\alpha = \Phi^\top\Phi v = Sv = \lambda v$, we have $v = \Phi^\top\alpha/\lambda$. Choosing $\alpha_v = \alpha/\lambda$ gives the desired result.

- (d) [4 pts] You are given a new data point $\mathbf{x} \in \mathbb{R}^d$. Find the scalar projection of its feature representation $\phi(\mathbf{x})$ onto $\mathbf{v}/\|\mathbf{v}\|$ (with \mathbf{v} defined as above).

Write your answer in terms of α_v and λ . Use the kernel k , and do not explicitly use ϕ . You should use the notation $\mathbf{k}_x = [k(\mathbf{x}_1, \mathbf{x}) \cdots k(\mathbf{x}_n, \mathbf{x})]^\top$.

First, let's calculate the squared norm of v :

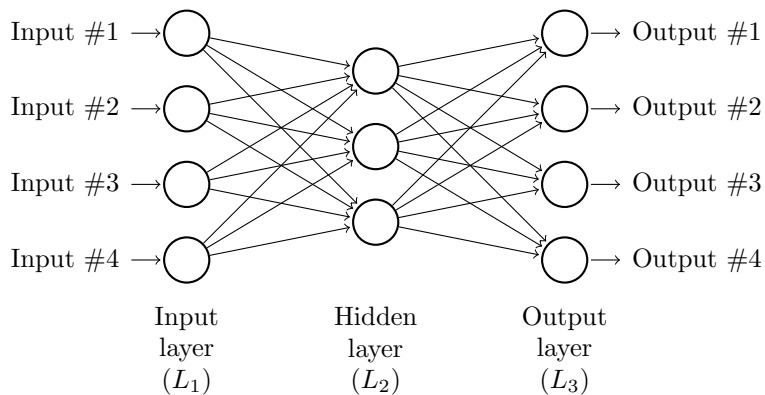
$$\|v\|^2 = v^\top v = \alpha_v^\top \Phi\Phi^\top \alpha_v = \alpha_v^\top K \alpha_v = \lambda \|\alpha_v\|^2$$

Then, the projection is:

$$\frac{v^\top \phi(x)}{\|v\|} = \frac{\alpha_v^\top \Phi \phi(x)}{\sqrt{\lambda} \|\alpha_v\|} = \frac{\alpha_v^\top k_x}{\sqrt{\lambda} \|\alpha_v\|}$$

Q8. [14 pts] Autoencoder

An autoencoder is a neural network designed to learn feature representations in an unsupervised manner. Unlike a standard multi-layer network, an autoencoder has the same number of nodes in its output layer as its input layer. An autoencoder is not trained to predict some target value y given input \mathbf{x} ; rather, it is trained to reconstruct its own input \mathbf{x} , i.e. to minimize the reconstruction error. An autoencoder is shown below.



Suppose the input is a set of P -dimensional unlabeled data $\{\mathbf{x}^{(i)}\}_{i=1}^N$. Consider an autoencoder with H hidden units in L_2 . We will use the following notation for this autoencoder:

- \mathbf{W}^e denotes the $P \times H$ weight matrix between L_1 and L_2
- \mathbf{W}^d denotes the $H \times P$ weight matrix between L_2 and L_3
- σ denotes the activation function for L_2 and L_3
- $s_j^{(i)} = \sum_{k=1}^P W_{kj}^e x_k^{(i)}$
- $z_j^{(i)} = \sigma \left(\sum_{k=1}^P W_{kj}^e x_k^{(i)} \right)$
- $t_j^{(i)} = \sum_{k=1}^H W_{kj}^d z_k^{(i)}$
- $\hat{x}_j^{(i)} = \sigma \left(\sum_{k=1}^H W_{kj}^d z_k^{(i)} \right)$
- $J(\mathbf{W}^e, \mathbf{W}^d)^{(i)} = \|\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}\|_2^2 = \sum_{j=1}^P (x_j^{(i)} - \hat{x}_j^{(i)})^2$ is the reconstruction error for example $\mathbf{x}^{(i)}$
- $J(\mathbf{W}^e, \mathbf{W}^d) = \sum_{i=1}^N J(\mathbf{W}^e, \mathbf{W}^d)^{(i)}$ is the total reconstruction error

(We add element 1 to the input layer and hidden layer so that no bias term has to be considered.)

- (a) [8 pts] Fill in the following derivative equations for \mathbf{W}^e and \mathbf{W}^d . Use the notation defined above; there should be no new notation needed.

$$\begin{aligned} \frac{\partial J^{(i)}}{\partial W_{kl}^d} &= \sum_{j=1}^P \left(\boxed{2(\hat{x}_j^{(i)} - x_j^{(i)})} \cdot \frac{\partial \hat{x}_j^{(i)}}{\partial W_{kl}^d} \right) \\ \frac{\partial \hat{x}_j^{(i)}}{\partial W_{kl}^d} &= \sigma' \left(\sum_{k=1}^P W_{kj}^e x_k^{(i)} \right) \cdot \boxed{z_k^{(i)}} \\ \frac{\partial J^{(i)}}{\partial W_{kl}^e} &= \frac{\partial J^{(i)}}{\partial s_j^{(i)}} \cdot \boxed{\frac{\partial s_j^{(i)}}{\partial W_{kl}^e} = x_k^{(i)}} \\ \frac{\partial J^{(i)}}{\partial s_j^{(i)}} &= \sum_{k=1}^H \left(\frac{\partial J^{(i)}}{\partial t_k^{(i)}} \cdot \boxed{W_{jk}^d} \cdot \sigma'(s_j^{(i)}) \right) \end{aligned}$$

- (b) [4 pts] To limit the number of activated hidden units, we add a sparsity penalty to the problem. The reconstruction error is formulated as

$$J_{sparse}(\mathbf{W}^e, \mathbf{W}^d) = J(\mathbf{W}^e, \mathbf{W}^d) + \beta \sum_{j=1}^H \text{KL}(\rho \| \hat{\rho}_j)$$

where $\hat{\rho}_j = \frac{1}{N} \sum_{i=1}^N z_j^{(i)}$, and ρ and β are hyperparameters. KL divergence is defined as

$$\text{KL}(\rho \| \hat{\rho}_j) = \rho \log \left(\frac{\rho}{\hat{\rho}_j} \right) + (1 - \rho) \log \left(\frac{1 - \rho}{1 - \hat{\rho}_j} \right)$$

Write the following derivative updates for \mathbf{W}^e and \mathbf{W}^d .

$$\begin{aligned} \frac{\partial J_{sparse}}{\partial W_{kl}^d} &= \frac{\partial J}{\partial W_{kl}^d} + \boxed{0} \\ \frac{\partial J_{sparse}}{\partial W_{kl}^e} &= \frac{\partial J}{\partial W_{kl}^e} + \beta \cdot \sum_{j=1}^H \left(\boxed{-\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j}} \cdot \frac{1}{N} \sum_{i=1}^N (x_k^{(i)} \sigma'(\sum_{s=1}^P W_{sj}^e x_s^{(i)})) \right) \end{aligned}$$

- (c) [2 pts] State some relations between autoencoders and PCA.

They are both feature representation learning methods. PCA is only linear transformation to the subspace while autoencoder is nonlinear transformation to the hidden units. If the autoencoder's activation functions are linear, it is very similar to PCA method.

- You have 2 hours 50 minutes for the exam.
- The exam is closed book, closed notes except your one-page (two-sided) cheat sheet.
- No calculators or electronic items.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation and state your assumptions.
- For true/false questions, fill in the *True/False* bubble.
- For multiple-choice questions, fill in the bubble for **EXACTLY ONE** choice that represents the best answer to the question.

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

For staff use only:

Q1. True or False	/44
Q2. Multiple Choice	/33
Q3. Decision Theory	/9
Q4. Parameter Estimation	/8
Q5. Locally Weighted Logistic Regression	/14
Q6. Decision Trees	/7
Q7. Convolutional Neural Nets	/11
Q8. Streaming k-means	/9
Q9. Low Dimensional Decompositions	/15
Total	/150

Q1. [44 pts] True or False

- (a) [2 pts] A neural network with multiple hidden layers and sigmoid nodes can form non-linear decision boundaries.
 True False
- (b) [2 pts] All neural networks compute non-convex functions of their parameters.
 True False
- (c) [2 pts] For logistic regression, with parameters optimized using a stochastic gradient method, setting parameters to 0 is an acceptable initialization.
 True False
- (d) [2 pts] For arbitrary neural networks, with weights optimized using a stochastic gradient method, setting weights to 0 is an acceptable initialization.
 True False
- (e) [2 pts] Given a design matrix $X \in \mathbb{R}^{n \times d}$, where $d \ll n$, if we project our data onto a k dimensional subspace using PCA where k equals the rank of X , we recreate a perfect representation of our data with no loss.
 True False
- (f) [2 pts] Hierarchical clustering methods require a predefined number of clusters, much like k -means.
 True False
- (g) [2 pts] Given a predefined number of clusters k , globally minimizing the k -means objective function is NP-hard.
 True False
- (h) [2 pts] Using cross validation to select hyperparameters will guarantee that our model does not overfit.
 True False
- (i) [2 pts] A random forest is an ensemble learning method that attempts to lower the bias error of decision trees.
 True False
- (j) [2 pts] Bagging algorithms attach weights $w_1 \dots w_n$ to a set of N weak learners. They re-weight the learners and convert them into strong ones. Boosting algorithms draw N sample distributions (usually with replacement) from an original data set for learners to train on.
 True False
- (k) [2 pts] Given any matrix X , its singular values are the eigenvalues of XX^T and $X^T X$.
 True False
- (l) [2 pts] Given any matrix X , $(XX^T + \lambda I)^{-1}$ for $\lambda \neq 0$ always exists.
 True False
- (m) [2 pts] Backpropagation is motivated by utilizing Chain Rule and Dynamic Programming to conserve mathematical calculations.
 True False
- (n) [2 pts] An infinite depth binary Decision Tree can always achieve 100% training accuracy, provided that no point is mislabeled in the training set.
 True False
- (o) [2 pts] In One vs All Multi-Class Classification in SVM, we are trying to classify an input data point X as one of the N classes ($C_1 \dots C_n$), each of which has a parameter vector $\vec{w}_1 \dots \vec{w}_n$. We classify point X as the class C_i which maximizes the inner product of X and \vec{w}_i . True False

- (p) [2 pts] The number of parameters in a parametric model is fixed, while the number of parameters in a non-parametric model grows with the amount of training data.
 True False
- (q) [2 pts] As model complexity increases, bias will decrease while variance will increase.
 True False
- (r) [2 pts] Consider a cancer diagnosis classification problem where almost all of the people being diagnosed don't have cancer. The probability of correct classification is the most important metric to optimize.
 True False
- (s) [2 pts] For the 1-Nearest Neighbors algorithm, as the number of data points increases to infinity in our dataset, the error of our algorithm is guaranteed to be bounded by twice the Bayes Risk.
 True False
- (t) [2 pts] Increasing the dimensionality of our data always decreases our misclassification rate.
 True False
- (u) [2 pts] It is possible to represent a XOR function with a neural network without a hidden layer.
 True False
- (v) [2 pts] At high dimensionality, the KD tree speedup to the nearest neighbor can be slower than the naive nearest neighbor implementation.
 True False

Q2. [33 pts] Multiple Choice

(a) [3 pts] Given a Neural Net with N input nodes, no hidden layers, one output node, with Entropy Loss and Sigmoid Activation Functions, which of the following algorithms (with the proper hyper-parameters and initialization) can be used to find the global optimum?

- Simulated Annealing (Gradient Descent with restarts)
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent
- Batch Gradient Descent
- All of the above
- None of the above

(b) [3 pts] Given function $f(x) = |x^2 + 3| - 1$ defined on \mathbb{R} :

- Newtons Method on minimizing gradients will always converge to the global optimum in one iteration from any starting location
- Stochastic Gradient Descent will always converge to the global optimum in one iteration
- The problem is nonconvex, so it not feasible to find a solution.
- All of the above
- None of the above

(c) [3 pts] Daniel wants to minimize a convex loss function $f(x)$ using stochastic gradient descent. Given a random starting point, mark the condition that would guarantee that stochastic gradient descent will converge to the global optimum. Let $\eta_t =$ step size at iteration t .

- $\eta_t < 0$
- Constant step size η_t
- Decreasing step size $\eta_t = \frac{1}{\sqrt{t}}$
- Decreasing step size $\eta_t = \frac{1}{t^2}$
- All of the above
- None of the above

(d) [3 pts] Which of the following is true of logistic regression?

- It can be motivated by "log odds"
- The optimal weight vector can be found using MLE.
- It can be used with L1 regularization
- All of the above
- None of the above

(e) [3 pts] You've just finished training a decision tree for spam classification, and it is getting abnormally bad performance on both your training and test sets. You know that your implementation has no bugs, so what could be causing the problem?

- Your decision trees are too shallow.
- You need to increase the learning rate.
- You are overfitting.
- All of the above.

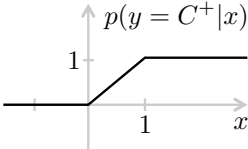
(f) [3 pts] The numerical output of a sigmoid node in a neural network:

- Is unbounded, encompassing all real numbers.
- Is unbounded, encompassing all integers.
- Is bounded between 0 and 1.
- Is bounded between -1 and 1.

- (g) [3 pts] If n is the number of points in the training set, regular nearest neighbor (without KD trees, hashing, etc) has a classification runtime of:
- $O(1)$
 $O(n)$
 $O(\log n)$
 $O(n^2)$
- (h) [3 pts] Consider the p -norm of a vector x defined using the notation $\|x\|_p$. Also note that α is a scalar. Which of the following is true?
- $\|x\|_p + \|y\|_p \geq \|x + y\|_p$.
 $\|x\|_p = 0$ implies x is the zero vector.
 $\|\alpha x\|_p = |\alpha| \|x\|_p$.
 All of the above.
- (i) [3 pts] What are some practical problems with the sigmoidal activation function in neural nets?
- It is convex, and convex functions cannot solve nonconvex problems
 It can have negative values
 It does not work well with the entropy loss function
 Gradients are small for values away from 0, leading to the "Vanishing Gradient" problem for large or recurrent neural nets
- (j) [3 pts] In Homework 4, you fit a logistic regression model on spam and ham data for a Kaggle Competition. Assume you had a very good score on the public test set, but when the GSIs ran your model on a private test set, your score dropped a lot. This is likely because you overfitted by submitting multiple times and changing which of the following between submissions: A) λ , your penalty term; B) η , your step size; C) ϵ , your convergence criterion; or D) Fixing a random bug:
- A
 A, B, and C
 B
 C and D
 A and B
 A, B, C, and D
- (k) [3 pts] With access to an n -by- n matrix of pairwise data distances, but no access to the data itself, we can use which of the following clustering techniques: A) k -means; B) k -medoids; C) hierarchical clustering:
- A
 A and B
 B
 B and C
 C
 A, B, and C
- (l) [0 pts] What was your favorite class of the ~~semester~~ year all time?
- CS 189 - Introduction to Machine Learning
 CS 189 - Kaggle Competitions for Dummies
 CS 189 - Classify EVERYTHING
 All of the above
 CS 189 - Advanced MATLAB and Numpy
 None of the above (Choose this if you dare...)

Q3. [9 pts] Decision Theory

We are given a test that is designed to predict whether a patient y has cancer C^+ or not C^- . The test returns a value $x \in \mathcal{R}$ and we know the probability of the patient having cancer given the test results:

$$p(y = C^+|x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } 0 \leq x < 1 \\ 1, & \text{if } 1 \leq x \end{cases}$$


We also know that it is three times more costly to have a false negative than a false positive. Specifically, the loss matrix is:

Predicted:

	C^-	C^+
Truth: C^-	0	10
C^+	30	0

Suppose that we choose a fixed value x^* , and we predict C^+ if the test result is greater than x^* and C^- otherwise.

(a) [2 pts] What is the decision boundary (value of x) that minimizes the misclassification probability?

$x = 1/2$ minimizes the classification boundary.

We want to choose x such that the probability of C^+ given x is the same as the probability of C^- given x :

$$\begin{aligned} P(y = C^+|x) &= P(y = C^-|x) \\ x &= 1 - x \\ x &= 1/2 \end{aligned}$$

(b) [3 pts] What is the decision boundary (value of x^*) that minimizes the risk?

$x^* = 1/4$ minimizes the risk.

Choose x^* such that the risk of choosing C^+ given x is the same as the risk of choosing of C^- given x :

$$\begin{aligned} R(\hat{y} = C^+|x^*) &= R(\hat{y} = C^-|x^*) \\ \ell(C^+, C^+)P(y = C^+|x^*) + \ell(C^+, C^-)P(y = C^-|x^*) &= \ell(C^-, C^+)P(y = C^+|x^*) + \ell(C^-, C^-)P(y = C^-|x^*) \\ \ell(C^+, C^-)P(y = C^-|x^*) &= \ell(C^-, C^+)P(y = C^+|x^*) \\ 10(1 - x^*) &= 30x^* \\ x^* &= 1/4 \end{aligned}$$

(c) [4 pts] If the test result is uniformly distributed in the interval $[-1, 1]$, what is the value of the minimum risk? Write your answer in terms of x^* (to avoid loss of points if your x^* is incorrect).

$$\begin{aligned}
\mathbb{E}\ell(\hat{y}, y) &= \mathbb{E}\mathbb{E}[\ell(\hat{y}, y)|x] \\
&= \int_{-\infty}^{\infty} \mathbb{1}[\hat{y} = C^-] \ell(C^-, C^+) P(y = C^+, x) + \mathbb{1}[\hat{y} = C^+] \ell(C^+, C^-) P(y = C^-, x) dx \\
&= \int_{-\infty}^{\infty} \mathbb{1}[\hat{y} = C^-] \ell(C^-, C^+) P(y = C^+ | x) P(x) + \mathbb{1}[\hat{y} = C^+] \ell(C^+, C^-) P(y = C^- | x) P(x) dx \\
&= \int_0^{x^*} \frac{1}{2} \ell(C^-, C^+) P(y = C^+ | x) dx + \int_{x^*}^1 \frac{1}{2} \ell(C^+, C^-) P(y = C^- | x) dx \\
&= \frac{1}{2} \int_0^{x^*} 30x dx + \int_{x^*}^1 10(1-x) dx \\
&= \frac{1}{2} \left(15x^2 \Big|_0^{x^*} + 10x - 5x^2 \Big|_{x^*}^1 \right) \\
&= \frac{1}{2} (15x^{*2} + 5 - 10x^* + 5x^{*2}) \\
&= 10x^{*2} - 5x^* + 2.5 = \frac{15}{8}
\end{aligned}$$

Q4. [8 pts] Parameter Estimation

Suppose you are given n observations, X_1, \dots, X_n , independent and identically distributed with a $\text{Gamma}(\alpha, \lambda)$ distribution. The following information might be useful for one or more parts of the problem.

- If $X \sim \text{Gamma}(\alpha, \lambda)$, then $\mathbb{E}[X] = \frac{\alpha}{\lambda}$ and $\mathbb{E}[X^2] = \frac{\alpha(\alpha + 1)}{\lambda^2}$
- The probability density function of $X \sim \text{Gamma}(\alpha, \lambda)$ is $f_X(x) = \frac{1}{\Gamma(\alpha)} \lambda^\alpha x^{\alpha-1} e^{-\lambda x}$ where the function Γ is only dependent on α and not λ .

The following notation might be useful for one or more parts of the problem: $\bar{X}_1 = \frac{1}{n} \sum_{i=1}^n X_i$ and $\bar{X}_2 = \frac{1}{n} \sum_{i=1}^n X_i^2$.

- (a) [4 pts] Find the estimators for α and λ using the method of moments. Remember you are trying to write α and λ as functions of the data.

First, we take the sample moments from the data, \bar{X}_1, \bar{X}_2 and equate that to the theoretical moments, $\mathbb{E}[X], \mathbb{E}[X^2]$.

$$\begin{aligned}\bar{X}_1 &= \frac{\alpha}{\lambda} \\ \bar{X}_2 &= \frac{\alpha(\alpha + 1)}{\lambda^2}\end{aligned}$$

Solving for α and λ from this system of equations, we get

$$\begin{aligned}\hat{\alpha}_{\text{MM}} &= \frac{\bar{X}_1^2}{\bar{X}_2 - \bar{X}_1^2} = \frac{\bar{X}_1^2}{\hat{\sigma}^2} \\ \hat{\lambda}_{\text{MM}} &= \frac{\bar{X}_1}{\bar{X}_2 - \bar{X}_1^2} = \frac{\bar{X}_1}{\hat{\sigma}^2}\end{aligned}$$

- (b) [4 pts] Suppose, we are given a known, fixed value for α . Compute the maximum likelihood estimator for λ .

We first write the likelihood function.

$$L(\lambda|X_1, \dots, X_n) = \prod_{i=1}^n \frac{1}{\Gamma(\alpha)} \lambda^\alpha X_i^{\alpha-1} e^{-\lambda X_i}$$

The log-likelihood function is given as follows.

$$l(\lambda|X_1, \dots, X_n) = -n \log(\Gamma(\alpha)) + n\alpha \log \lambda + \sum_{i=1}^n (\alpha - 1) \log X_i - \lambda X_i$$

Next, we take the gradient with respect to λ and set it equal to 0.

$$\nabla_\lambda l(\lambda|X_1, \dots, X_n) = \frac{n\alpha}{\lambda} - \sum_{i=1}^n X_i = 0$$

Solving for λ , we get,

$$\hat{\lambda}_{\text{MLE}} = \frac{n\alpha}{\sum_{i=1}^n X_i} = \frac{\alpha}{\frac{1}{n} \sum_{i=1}^n X_i} = \frac{\alpha}{\bar{X}_1}$$

Q5. [14 pts] Locally Weighted Logistic Regression

In this problem, we consider solving the problem of locally weighted logistic regression. Given data $\{(x_i, y_i) \in \mathbb{R}^d \times \{0, 1\}\}_{i=1}^n$ and a query point x , we choose a parameter vector θ to minimize the loss (which is simply the negative log likelihood, weighted appropriately):

$$l(\theta; x) = - \sum_{i=1}^n w_i(x) [y_i \log(\mu(x_i)) + (1 - y_i) \log(1 - \mu(x_i))]$$

where

$$\mu(x_i) = \frac{1}{1 + e^{-\theta \cdot x_i}}, \quad w_i(x) = \exp\left(-\frac{\|x - x_i\|^2}{2\tau}\right)$$

where τ is a hyperparameter that must be tuned. Note that whenever we receive a new query point x , we must solve the entire problem again with these new weights $w_i(x)$.

Hint: the derivative of the logistic regression log likelihood with respect to θ is: $\sum_{i=1}^n (y_i - \mu(x_i))x_i$

(a) [4 pts] Given a data point x , derive the gradient of $l(\theta; x)$ with respect to θ .

The derivation is extremely similar to the derivation of logistic regression as in the slides. The answer is

$$-X^\top z = - \sum_{i=1}^n w_i(x) (y_i - \mu(x_i)) x_i$$

where X is the design matrix (i.e. every row in X is a data point), and $z_i = w_i(x)(y_i - \mu(x_i))$.

(b) [4 pts] Given a data point x , derive the Hessian of $l(\theta; x)$ with respect to θ .

The derivation is extremely similar to the derivation of logistic regression as in the slides. The answer is

$$X^\top DX = \sum_{i=1}^n w_i(x) \mu(x_i) (1 - \mu(x_i)) x_i x_i^\top$$

(c) [2 pts] Given a data point x , write the update formula for gradient descent. Use the symbol η for an arbitrary step size.

$$\theta^{(t+1)} = \theta^{(t)} + \eta X^\top z$$

(d) [2 pts] Given a data point x , write the update formula for Newton's method.

$$\theta^{(t+1)} = \theta^{(t)} + [X^\top DX]^{-1} X^\top z$$

(e) [2 pts] Locally Weighted Logistic Regression is a

Parametric method

Nonparametric method

Q6. [7 pts] Decision Trees

Answer the following questions related to decision trees.

- (a) [3 pts] In Homework 5, you first implemented a decision tree and then implemented a decision forest, which uses an ensemble method called bagging.

For neural network classification, it is typical to train k networks and average the results. Why not run your decision tree (using all of the data) k times and then average the results?

Decision trees return the same result each time you run them.

- (b) [2 pts] True or false: Selecting the decision tree split (at each node as you move down the tree) that *minimizes classification error* will guarantee an optimal decision tree.

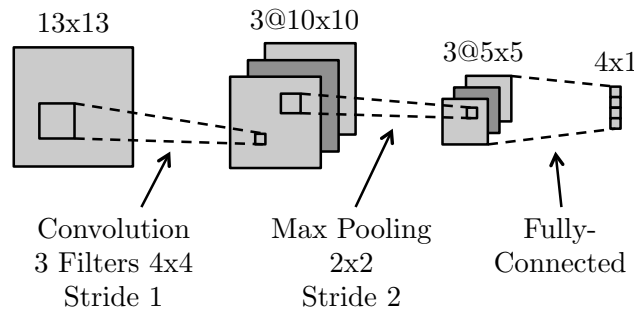
True False

- (c) [2 pts] True or false: Selecting the decision tree split (at each node as you move down the tree) that *maximizes information gain* will guarantee an optimal decision tree.

True False

Q7. [11 pts] Convolutional Neural Nets

Below is a diagram of a small convolutional neural network that converts a 13x13 image into 4 output values. The network has the following layers/operations from input to output: convolution with 3 filters, max pooling, ReLu, and finally a fully-connected layer. For this network we will not be using any bias/offset parameters (b). Please answer the following questions about this network.



- (a) [2 pts] How many weights in the convolutional layer do we need to learn?
48 weights. Three filters with 4x4=16 weights each.
- (b) [2 pts] How many ReLu operations are performed on the forward pass?
75 ReLu operations. ReLu is performed after the pooling step. ReLu is performed on each pixel of the three 5x5 feature images.
- (c) [2 pts] How many weights do we need to learn for the entire network?
348 weights. 48 for the convolutional layer. Fully-connected has 3x5x5=75 pixels each connected to four outputs, which is 300 weights. Pooling layer does not have any weights.
- (d) [2 pts] True or false: A fully-connected neural network with the same size layers as the above network (13x13 → 3x10x10 → 3x5x5 → 4x1) can represent any classifier that the above convolutional network can represent.
 True False
- (e) [3 pts] What is the disadvantage of a fully-connected neural network compared to a convolutional neural network with the same size layers?
Too many weights to effectively learn.

Q8. [9 pts] Streaming k-means

The standard k-means algorithm loads all data points altogether into the memory. In practice, data usually comes in a stream, such that they are sequentially processed and dropped (not stored in memory). The advantage of streaming algorithms is that their memory requirement is independent of the stream length. Thus, streaming algorithms are very useful in processing data that cannot fit into the memory.

In this problem, we will explore how to extend the k-means algorithm to process streaming data. Suppose that there are k clusters. The cluster centers are randomly initialized. Once the processor receives a data point $x \in \mathbb{R}^d$, it does the following:

1. Find the cluster whose center is the closest to x (in Euclidean distance), then add x to the cluster
2. Adjust the cluster center so that it equals the mean of all cluster members.

The algorithm outputs the k cluster centres after processing all data points in the stream.

According to the above algorithm specification, complete the streaming algorithm for k-means. Note that the algorithm's memory requirement should be independent of the stream length.

- (a) [3 pts] List the variables that are stored in the memory and their initial values. Which variables should be the output of the algorithm?

Two sets of variables need to be stored:

- c_i : the center of the i -th cluster ($i = 1, \dots, k$). It is a d -dimensional vector and should be randomly initialized in \mathbb{R}^d .
- n_i : the number of data points that belong to the i -th cluster. It should be initialized by 0.

The algorithm's output should be c_i ($i = 1, \dots, k$).

- (b) [3 pts] When the processor receives a data point x , state the updates that are made on the variables.

The algorithm executes the following three steps:

1. Find the index i which minimizes $\|x - c_i\|_2$ for $i = 1, \dots, k$.
2. Update $c_i \leftarrow (n_i c_i + x)/(n_i + 1)$.
3. Update $n_i \leftarrow n_i + 1$.

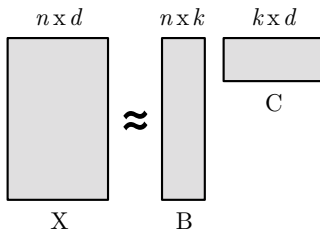
- (c) [3 pts] In each iteration, suppose the processor receives a data point x along with its weight $w > 0$. We want the cluster center to be the weighted average of all cluster members. How do you modify the updates in question (b) to process weighted data?

With weighted data, the algorithm executes the following three steps:

1. Find the index i which minimizes $\|x - c_i\|_2$ for $i = 1, \dots, k$.
2. Update $c_i \leftarrow (n_i c_i + wx)/(n_i + w)$.
3. Update $n_i \leftarrow n_i + w$.

Q9. [15 pts] Low Dimensional Decompositions

Given a design matrix $X \in \mathbb{R}^{n \times d}$ with $n > d$, we can create a low dimensional decomposition approximation $\tilde{X} = BC$, where $\tilde{X} \in \mathbb{R}^{n \times d}$, $B \in \mathbb{R}^{n \times k}$, $C \in \mathbb{R}^{k \times d}$, and $k < d$. The following figure shows a diagram of X approximated by B times C :



We can formulate several low dimensional techniques from CS 189 as solving the following optimization, subject to various constraints:

$$\min_{B,C} \|X - BC\|_F^2, \quad (1)$$

where $\|\cdot\|_F^2$ denotes the squared Frobenius norm of a matrix, that is, the sum of its squared entries.

- (a) [2 pts] Which machine learning technique corresponds to solving (1) with constraint \mathcal{C}_1 : each row of B is a vector e_i (a vector of all zeros, except a one in position i)?

k -means k -medoids SVD of X

- (b) [3 pts] Describe the B and C matrices that result from solving (1) with constraint \mathcal{C}_1 .

The rows of C are the cluster centers (the means) and the rows of B indicate which cluster each point belongs to.

- (c) [2 pts] Which machine learning technique corresponds to solving (1) with constraint \mathcal{C}_2 : each column of B has norm equal to one?

k -means k -medoids SVD of X

- (d) [3 pts] Describe the B and C matrices that result from solving (1) with constraint \mathcal{C}_2 .

B is the first k left singular vectors of X and C is the transpose of the first k right singular vectors of X scaled by the first k singular values of X .

$$X = U\Sigma V^T, \quad B = U_k, \quad \text{and} \quad C = \Sigma_k V_k^T.$$

- (e) [2 pts] Which machine learning technique corresponds to solving (1) with the constraints \mathcal{C}_3 : each row of C is one of the rows from X and each row of B is a vector e_i (a vector of all zeros, except a one in position i)?

k -means k -medoids SVD of X

- (f) [3 pts] Describe the B and C matrices that result from solving (1) with constraints \mathcal{C}_3 .

The rows of C are the medoids (points from X representing the cluster centers) and the rows of B indicate which cluster each point belongs to.

- Please do not open the exam before you are instructed to do so.
- The exam is closed book, closed notes except your two-page cheat sheet.
- **Electronic devices are forbidden on your person**, including cell phones, iPods, headphones, and laptops. Turn your cell phone off and **leave all electronics at the front of the room**, or **risk getting a zero** on the exam.
- You have 3 hours.
- Please write your initials at the top right of each page (e.g., write “JS” if you are Jonathan Shewchuk). Finish this by the end of your 3 hours.
- Mark your answers on **front** of each page, **not** the back. We will not scan the backs of each page, but you may use them as scratch paper. Do **not** attach any extra sheets.
- The total number of points is 150. There are 30 multiple choice questions worth 3 points each, and 6 written questions worth a total of 60 points.
- For multiple-choice questions, fill in the boxes 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-choice questions: the set of all correct answers must be checked.

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

Q1. [90 pts] Multiple Choice

Check the boxes for **ALL CORRECT CHOICES**. Every question should have at least one box checked. **NO PARTIAL CREDIT**: the set of all correct answers (only) must be checked.

(1) [3 pts] What strategies can help reduce overfitting in decision trees?

- Pruning
- Make sure each leaf node is one pure class
- Enforce a minimum number of samples in leaf nodes
- Enforce a maximum depth for the tree

(2) [3 pts] Which of the following are true of convolutional neural networks (CNNs) for image analysis?

- Filters in earlier layers tend to include edge detectors
- Pooling layers reduce the spatial resolution of the image
- They have more parameters than fully-connected networks with the same number of layers and the same numbers of neurons in each layer
- A CNN can be trained for unsupervised learning tasks, whereas an ordinary neural net cannot

(3) [3 pts] Neural networks

- optimize a convex cost function
- can be used for regression as well as classification
- always output values between 0 and 1
- can be used in an ensemble

(4) [3 pts] Which of the following are true about generative models?

- They model the joint distribution $P(\text{class} = C \text{ AND sample} = \mathbf{x})$
- They can be used for classification
- The perceptron is a generative model
- Linear discriminant analysis is a generative model

(5) [3 pts] Lasso can be interpreted as least-squares linear regression where

- weights are regularized with the ℓ_1 norm
- weights are regularized with the ℓ_2 norm
- the weights have a Gaussian prior
- the solution algorithm is simpler

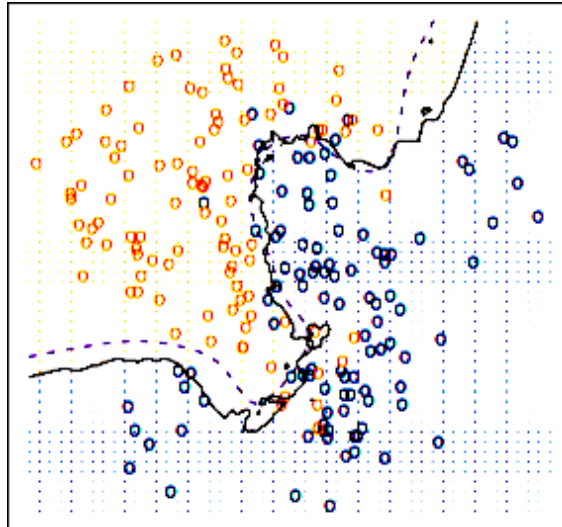
(6) [3 pts] Which of the following methods can achieve zero training error on *any* linearly separable dataset?

- Decision tree
- Hard-margin SVM
- 15-nearest neighbors
- Perceptron

(7) [3 pts] The kernel trick

- can be applied to every classification algorithm
- is commonly used for dimensionality reduction
- changes ridge regression so we solve a $d \times d$ linear system instead of an $n \times n$ system, given n sample points with d features
- exploits the fact that in many learning algorithms, the weights can be written as a linear combination of input points

- (15) [3 pts] Which value of k in the k -nearest neighbors algorithm generates the solid decision boundary depicted here? There are only 2 classes. (Ignore the dashed line, which is the Bayes decision boundary.)



- $k = 1$
 $k = 2$
 $k = 10$
 $k = 100$
- (16) [3 pts] Consider one layer of weights (edges) in a convolutional neural network (CNN) for grayscale images, connecting one layer of units to the next layer of units. Which type of layer has the fewest parameters to be learned during training? (Select one.)
- A convolutional layer with 10 3×3 filters
 A convolutional layer with 8 5×5 filters
 A max-pooling layer that reduces a 10×10 image to 5×5
 A fully-connected layer from 20 hidden units to 4 output units
- (17) [3 pts] In the kernelized perceptron algorithm with learning rate $\epsilon = 1$, the coefficient a_i corresponding to a training example x_i represents the weight for $K(x_i, x)$. Suppose we have a two-class classification problem with $y_i \in \{1, -1\}$. If $y_i = 1$, which of the following can be true for a_i ?
- $a_i = -1$
 $a_i = 1$
 $a_i = 0$
 $a_i = 5$
- (18) [3 pts] Suppose you want to split a graph G into two subgraphs. Let L be G 's Laplacian matrix. Which of the following could help you find a good split?
- The eigenvector corresponding to the second-largest eigenvalue of L
 The left singular vector corresponding to the second-largest singular value of L
 The eigenvector corresponding to the second-smallest eigenvalue of L
 The left singular vector corresponding to the second-smallest singular value of L
- (19) [3 pts] Which of the following are properties that a kernel matrix always has?
- Invertible
 All the entries are positive
 At least one negative eigenvalue
 Symmetric

(20) [3 pts] How does the bias-variance decomposition of a ridge regression estimator compare with that of ordinary least squares regression? (Select one.)

- Ridge has larger bias, larger variance
- Ridge has smaller bias, larger variance
- Ridge has larger bias, smaller variance
- Ridge has smaller bias, smaller variance

(21) [3 pts] Both PCA and Lasso can be used for feature selection. Which of the following statements are true?

- Lasso selects a subset (not necessarily a strict subset) of the original features
- PCA and Lasso both allow you to specify how many features are chosen
- PCA produces features that are linear combinations of the original features
- PCA and Lasso are the same if you use the kernel trick

(22) [3 pts] Which of the following are true about forward subset selection?

- $O(2^d)$ models must be trained during the algorithm, where d is the number of features
- It finds the subset of features that give the lowest test error
- It greedily adds the feature that most improves cross-validation accuracy
- Forward selection is faster than backward selection if few features are relevant to prediction

(23) [3 pts] You've just finished training a random forest for spam classification, and it is getting abnormally bad performance on your validation set, but good performance on your training set. Your implementation has no bugs. What could be causing the problem?

- Your decision trees are too deep
- You have too few trees in your ensemble
- You are randomly sampling too many features when you choose a split
- Your bagging implementation is randomly sampling sample points *without* replacement

(24) [3 pts] Consider training a decision tree given a design matrix $X = \begin{bmatrix} 6 & 3 \\ 2 & 7 \\ 9 & 6 \\ 4 & 2 \end{bmatrix}$ and labels $y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$. Let f_1 denote feature 1, corresponding to the first column of X , and let f_2 denote feature 2, corresponding to the second column. Which of the following splits at the root node gives the highest information gain? (Select one.)

- $f_1 > 2$
- $f_2 > 3$
- $f_1 > 4$
- $f_2 > 6$

(25) [3 pts] In terms of the bias-variance decomposition, a 1-nearest neighbor classifier has _____ than a 3-nearest neighbor classifier.

- higher variance
- higher bias
- lower variance
- lower bias

(26) [3 pts] Which of the following are true about bagging?

In bagging, we choose random subsamples of the input points with replacement

Bagging is ineffective with logistic regression, because all of the learners learn exactly the same decision boundary

The main purpose of bagging is to decrease the bias of learning algorithms.

If we use decision trees that have one sample point per leaf, bagging never gives lower training error than one ordinary decision tree

(27) [3 pts] An advantage of searching for an approximate nearest neighbor, rather than the exact nearest neighbor, is that

it sometimes makes exhaustive search much faster

it sometimes makes searching in a k -d tree much faster

the nearest neighbor classifier is sometimes much more accurate

you find all the points within a distance of $(1 + \epsilon)r$ from the query point, where r is the distance from the query point to its nearest neighbor

(28) [3 pts] In the derivation of the spectral graph partitioning algorithm, we *relax* a combinatorial optimization problem to a continuous optimization problem. This relaxation has the following effects.

The combinatorial problem requires an exact bisection of the graph, but the continuous algorithm can produce (after rounding) partitions that aren't perfectly balanced

The combinatorial problem cannot be modified to accommodate vertices that have different masses, whereas the continuous problem can

The combinatorial problem requires finding eigenvectors, whereas the continuous problem requires only matrix multiplication

The combinatorial problem is NP-hard, but the continuous problem can be solved in polynomial time

(29) [3 pts] The firing rate of a neuron

determines how strongly the dendrites of the neuron stimulate axons of neighboring neurons

only changes very slowly, taking a period of several seconds to make large adjustments

is more analogous to the output of a unit in a neural net than the output voltage of the neuron

can sometimes exceed 30,000 action potentials per second

(30) [3 pts] In algorithms that use the kernel trick, the Gaussian kernel

gives a regression function or predictor function that is a linear combination of Gaussians centered at the sample points

is less prone to oscillating than polynomials, assuming the variance of the Gaussians is large

is equivalent to lifting the d -dimensional sample points to points in a space whose dimension is exponential in d

has good properties in theory but is rarely used in practice

(31) **3 bonus points!** The following Berkeley professors were cited in this semester's lectures (possibly self-cited) for specific research contributions they made to machine learning.

David Culler

Jitendra Malik

Anca Dragan

Michael Jordan

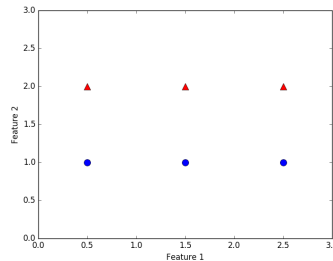
Leo Breiman

Jonathan Shewchuk

Q2. [8 pts] Feature Selection

A newly employed former CS 189/289A student trains the latest Deep Learning classifier and obtains state-of-the-art accuracy. However, the classifier uses too many features! The boss is overwhelmed and asks for a model with fewer features.

Let's try to identify the most important features. Start with a simple dataset in \mathbb{R}^2 .

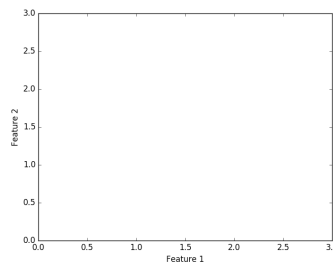


- (1) [4 pts] Describe the training error of a Bayes optimal classifier that can see only the first feature of the data. Describe the training error of a Bayes optimal classifier that can see only the second feature.

The first feature yields a training error of 50% (like random guessing). The second feature offers a training error of zero.

- (2) [4 pts] Based on this toy example, the student decides to fit a classifier on each feature individually, then rank the features by their classifier's accuracy, take the best k features, and train a new classifier on those k features. We call this approach *variable ranking*. Unfortunately, the classifier trained on the best k features obtains horrible accuracy, unless k is very close to d , the original number of features!

Construct a toy dataset in \mathbb{R}^2 for which variable ranking fails. In other words, a dataset where a variable is useless by itself, but potentially useful alongside others. Use + for data points in Class 1, and O for data points in Class 2.



An XOR Dataset is unpredictable with either feature. (This extends to n -dimensions, with the n -bit parity string.)

Q3. [10 pts] Gradient Descent for k -means Clustering

Recall the loss function for k -means clustering with k clusters, sample points x_1, \dots, x_n , and centers μ_1, \dots, μ_k :

$$L = \sum_{j=1}^k \sum_{x_i \in S_j} \|x_i - \mu_j\|^2,$$

where S_j refers to the set of data points that are closer to μ_j than to any other cluster mean.

- (1) [4 pts] Instead of updating μ_j by computing the mean, let's minimize L with **batch** gradient descent while holding the sets S_j fixed. Derive the update formula for μ_1 with learning rate (step size) ϵ .

$$\begin{aligned} \frac{\partial L}{\partial \mu_1} &= \frac{\partial}{\partial \mu_1} \sum_{x_i \in S_1} (x_i - \mu_1)^\top (x_i - \mu_1) \\ &= \sum_{x_i \in S_1} 2(\mu_1 - x_i). \end{aligned}$$

Therefore the update formula is

$$\mu_1 \leftarrow \mu_1 + \epsilon \sum_{x_i \in S_1} (x_i - \mu_1).$$

(Note: writing 2ϵ instead of ϵ is fine.)

- (2) [2 pts] Derive the update formula for μ_1 with **stochastic** gradient descent on a single sample point x_i . Use learning rate ϵ .

$\mu_1 \leftarrow \mu_1 + \epsilon(x_i - \mu_1)$ if $x_i \in S_1$, otherwise no change.

- (3) [4 pts] In this part, we will connect the batch gradient descent update equation with the standard k -means algorithm. Recall that in the update step of the standard algorithm, we assign each cluster center to be the mean (centroid) of the data points closest to that center. It turns out that a particular choice of the learning rate ϵ (which may be different for each cluster) makes the two algorithms (batch gradient descent and the standard k -means algorithm) have identical update steps. Let's focus on the update for the first cluster, with center μ_1 . Calculate the value of ϵ so that both algorithms perform the same update for μ_1 . (If you do it right, the answer should be very simple.)

In the standard algorithm, we assign $\mu_1 \leftarrow \sum_{x_i \in S_1} \frac{1}{|S_1|} x_i$.

Comparing to the answer in (1), we set $\sum_{x_i \in S_1} \frac{1}{|S_1|} x_i = \mu_1 + \epsilon \sum_{x_i \in S_1} (x_i - \mu_1)$ and solve for ϵ .

$$\begin{aligned} \sum_{x_i \in S_1} \frac{1}{|S_1|} x_i - \sum_{x_i \in S_1} \frac{1}{|S_1|} \mu_1 &= \epsilon \sum_{x_i \in S_1} (x_i - \mu_1) \\ \sum_{x_i \in S_1} \frac{1}{|S_1|} (x_i - \mu_1) &= \epsilon \sum_{x_i \in S_1} (x_i - \mu_1). \end{aligned}$$

Thus $\epsilon = \frac{1}{|S_1|}$.

(Note: answers that differ by a constant factor are fine if consistent with answer for (1).)

Q4. [10 pts] Kernels

- (1) [2 pts] What is the primary motivation for using the kernel trick in machine learning algorithms?

If we want to map sample points to a very high-dimensional feature space, the kernel trick can save us from having to compute those features explicitly, thereby saving a lot of time.

(Alternative solution: the kernel trick enables the use of infinite-dimensional feature spaces.)

- (2) [4 pts] Prove that for every design matrix $X \in \mathbb{R}^{n \times d}$, the corresponding kernel matrix is positive semidefinite.

For every vector $\mathbf{z} \in \mathbb{R}^n$,

$$\mathbf{z}^\top K \mathbf{z} = \mathbf{z}^\top X X^\top \mathbf{z} = |X^\top \mathbf{z}|^2,$$

which is clearly nonnegative.

- (3) [2 pts] Suppose that a regression algorithm contains the following line of code.

$$\mathbf{w} \leftarrow \mathbf{w} + X^\top M X X^\top \mathbf{u}$$

Here, $X \in \mathbb{R}^{n \times d}$ is the design matrix, $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, $M \in \mathbb{R}^{n \times n}$ is a matrix unrelated to X , and $\mathbf{u} \in \mathbb{R}^n$ is a vector unrelated to X . We want to derive a dual version of the algorithm in which we express the weights \mathbf{w} as a linear combination of samples X_i (rows of X) and a dual weight vector \mathbf{a} contains the coefficients of that linear combination. Rewrite the line of code in its dual form so that it updates \mathbf{a} correctly (and so that \mathbf{w} does not appear).

$$\mathbf{a} \leftarrow \mathbf{a} + M X X^\top \mathbf{u}$$

- (4) [2 pts] Can this line of code for updating \mathbf{a} be kernelized? If so, show how. If not, explain why.

Yes:

$$\mathbf{a} \leftarrow \mathbf{a} + M K \mathbf{u}$$

Q5. [12 pts] Let's PCA

You are given a design matrix $X = \begin{bmatrix} 6 & -4 \\ -3 & 5 \\ -2 & 6 \\ 7 & -3 \end{bmatrix}$. Let's use PCA to reduce the dimension from 2 to 1.

- (1) [6 pts] Compute the covariance matrix for the sample points. (Warning: Observe that X is not centered.) Then compute the **unit** eigenvectors, and the corresponding eigenvalues, of the covariance matrix. *Hint:* If you graph the points, you can probably guess the eigenvectors (then verify that they really are eigenvectors).

The covariance matrix is $X^T X = \begin{bmatrix} 82 & -80 \\ -80 & 82 \end{bmatrix}$.

Its unit eigenvectors are $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$ with eigenvalue 2 and $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$ with eigenvalue 162. (Note: either eigenvector can be replaced with its negation.)

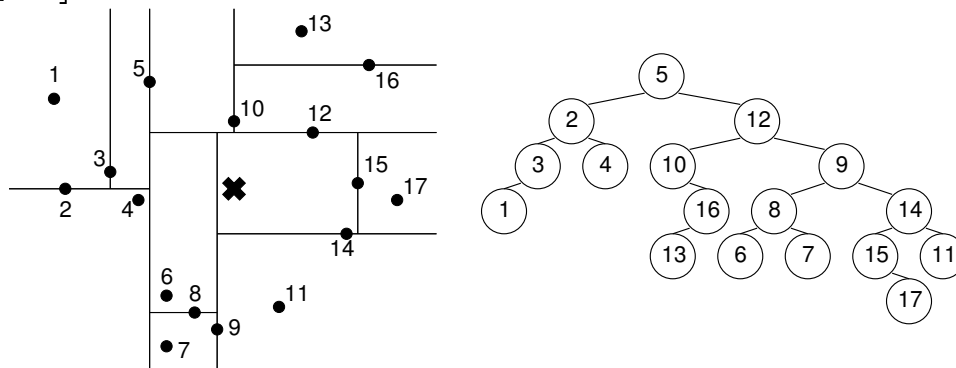
- (2) [3 pts] Suppose we use PCA to project the sample points onto a one-dimensional space. What one-dimensional subspace are we projecting onto? For each of the four sample points in X (not the centered version of X !), write the coordinate (in principal coordinate space, not in \mathbb{R}^2) that the point is projected to.

We are projecting onto the subspace spanned by $\begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$. (Equivalently, onto the space spanned by $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Equivalently, onto the line $x + y = 0$.) The projections are $(6, -4) \rightarrow \frac{10}{\sqrt{2}}$, $(-3, 5) \rightarrow -\frac{8}{\sqrt{2}}$, $(-2, 6) \rightarrow -\frac{8}{\sqrt{2}}$, $(7, -3) \rightarrow \frac{10}{\sqrt{2}}$.

- (3) [3 pts] Given a design matrix X that is taller than it is wide, prove that every right singular vector of X with singular value σ is an eigenvector of the covariance matrix with eigenvalue σ^2 .

If v is a right singular vector of X , then there is a singular value decomposition $X = UDV^T$ such that v is a column of V . Here each of U and V has orthonormal columns, V is square, and D is square and diagonal. The covariance matrix is $X^T X = VDU^TUDV^T = VD^2V^T$. This is an eigendecomposition of $X^T X$, so each singular vector in V with singular value σ is an eigenvector of $X^T X$ with eigenvalue σ^2 .

Q6. [10 pts] Trees



- (1) [5 pts] Above, we have two depictions of the same k -d tree, which we have built to solve nearest neighbor queries. Each node of the tree at right represents a rectangular box at left, and also stores one of the sample points that lie inside that box. (The root node represents the whole plane \mathbb{R}^2 .) If a treenode stores sample point i , then the line passing through point i (in the diagram at left) determines which boxes the child treenodes represent.

Simulate running an exact 1-nearest neighbor query, where the bold X is the query point. Recall that the query algorithm visits the treenodes in a smart order, and keeps track of the nearest point it has seen so far.

- Write down the numbers of all the sample points that serve as the “nearest point seen so far” sometime while the query algorithm is running, in the order they are encountered.
- Circle all the subtrees in the k -d tree at upper right that are never visited during this query. (This is why k -d tree search is usually faster than exhaustive search.)

Nearest point seen so far: first 5, then 12, then 10.

The unvisited subtrees are rooted at 2, 13, 7, and 17.

- (2) [5 pts] We are building a decision tree for a 2-class classification problem. We have n training points, each having d real-valued features. At each node of the tree, we try every possible univariate split (i.e. for each feature, we try every possible splitting value for that feature) and choose the split that maximizes the information gain.

Explain why it is possible to build the tree in $O(ndh)$ time, where h is the depth of the tree’s deepest node. Your explanation should include an analysis of the time to choose one node’s split. Assume that we can radix sort real numbers in linear time.

Consider choosing the split at a node whose box contains n' sample points. For each of the d features, we can sort the sample points in $O(n'd)$ time. Then we can compute the entropy for the first split (separating the first sample in the sorted list from the others) in $O(n')$ time, then we can walk through the list and update the entropy for each successive split in $O(1)$ time, summing to a total of $O(n')$ time for each of the d features. So it takes $O(n'd)$ time overall to choose a split.

Each sample point participates in at most h treenodes, so each sample point contributes at most dh to the running time, for a total running time of at most $O(ndh)$.

Q7. [10 pts] Self-Driving Cars and Backpropagation

You want to train a neural network to drive a car. Your training data consists of grayscale 64×64 pixel images. The training labels include the human driver's steering wheel angle in degrees and the human driver's speed in miles per hour. Your neural network consists of an input layer with $64 \times 64 = 4,096$ units, a hidden layer with 2,048 units, and an output layer with 2 units (one for steering angle, one for speed). You use the ReLU activation function for the hidden units and no activation function for the outputs (or inputs).

- (1) [2 pts] Calculate the number of parameters (weights) in this network. You can leave your answer as an expression. Be sure to account for the bias terms.

$$4097 \times 2048 + 2049 \times 2$$

- (2) [3 pts] You train your network with the cost function $J = \frac{1}{2}|\mathbf{y} - \mathbf{z}|^2$. Use the following notation.

- \mathbf{x} is a training image (input) vector with a 1 component appended to the end, \mathbf{y} is a training label (input) vector, and \mathbf{z} is the output vector. All vectors are column vectors.
- $r(\gamma) = \max\{0, \gamma\}$ is the ReLU activation function, $r'(\gamma)$ is its derivative (1 if $\gamma > 0$, 0 otherwise), and $r(\mathbf{v})$ is $r(\cdot)$ applied component-wise to a vector.
- \mathbf{g} is the vector of hidden unit values before the ReLU activation functions are applied, and $\mathbf{h} = r(\mathbf{g})$ is the vector of hidden unit values after they are applied (but we append a 1 component to the end of \mathbf{h}).
- V is the weight matrix mapping the input layer to the hidden layer; $\mathbf{g} = V\mathbf{x}$.
- W is the weight matrix mapping the hidden layer to the output layer; $\mathbf{z} = W\mathbf{h}$.

Derive $\partial J / \partial W_{ij}$.

$$\begin{aligned} \frac{\partial J}{\partial W_{ij}} &= (\mathbf{z} - \mathbf{y})^\top \frac{\partial \mathbf{z}}{\partial W_{ij}} \\ &= (\mathbf{z}_i - \mathbf{y}_i) \mathbf{h}_j \end{aligned}$$

- (3) [1 pt] Write $\partial J / \partial W$ as an outer product of two vectors. $\partial J / \partial W$ is a matrix with the same dimensions as W ; it's just like a gradient, except that W and $\partial J / \partial W$ are matrices rather than vectors.

$$\frac{\partial J}{\partial W} = (\mathbf{z} - \mathbf{y}) \mathbf{h}^\top$$

- (4) [4 pts] Derive $\partial J / \partial V_{ij}$.

$$\begin{aligned} \frac{\partial J}{\partial V_{ij}} &= (\mathbf{z} - \mathbf{y})^\top \frac{\partial \mathbf{z}}{\partial V_{ij}} \\ &= (\mathbf{z} - \mathbf{y})^\top W \frac{\partial \mathbf{h}}{\partial V_{ij}} \\ &= (\mathbf{z} - \mathbf{y})^\top W [0, \dots, r'(\mathbf{g}_i) \mathbf{x}_j, \dots, 0]^\top \\ &= ((\mathbf{z} - \mathbf{y})^\top W)_i r'(\mathbf{g}_i) \mathbf{x}_j. \end{aligned}$$

- Please do not open the exam before you are instructed to do so.
- The exam is closed book, closed notes except your two page cheat sheet.
- **Electronic devices are forbidden on your person**, including cell phones, iPods, headphones, and laptops. Turn your cell phone off and **leave all electronics at the front of the room**, or **risk getting a zero** on the exam.
- You have 3 hours.
- Please write your initials at the top right of each odd-numbered page (e.g., write “JS” if you are Jonathan Shewchuk). Finish this by the end of your 3 hours.
- Mark your answers on the exam itself in the space provided. Do **not** attach any extra sheets.
- The total number of points is 150. There are 26 multiple choice questions worth 3 points each, and 7 written questions worth a total of 72 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	
First and last name of student to your left	
First and last name of student to your right	

Q1. [78 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.

(1) [3 pts] Which of the following are NP-hard problems? Let $X \in \mathbb{R}^{n \times d}$ be a design matrix, let $y \in \mathbb{R}^n$ be a vector of labels, let L be the Laplacian matrix of some n -vertex graph, and let $\mathbf{1} = [1 \ 1 \ \dots \ 1]^\top$.

- $\min_{\mu, y} \sum_{i=1}^k \sum_{y_j=i} |X_j - \mu_i|^2$ where each μ_i is the mean of sample points assigned class i
- $\min_{y \in \mathbb{R}^n} \frac{1}{4} y^\top L y$ subject to $|y|^2 = n; \mathbf{1}^\top y = 0$
- $\min_y \sum_{i=1}^k \sum_{y_j=i} |X_j - \mu_i|^2$ with each μ_i fixed
- $\min_{y \in \mathbb{R}^n} \frac{1}{4} y^\top L y$ subject to $\forall j, y_j \in \{-1, +1\}; \mathbf{1}^\top y = 0$

(2) [3 pts] Which clustering algorithms permit you to decide the number of clusters after the clustering is done?

- k -means clustering
- a k -d tree used for divisive clustering
- agglomerative clustering with single linkage
- spectral graph clustering with 3 eigenvectors

(3) [3 pts] For which of the following does normalizing your input features influence the predictions?

- decision tree (with usual splitting method)
- neural network
- Lasso
- soft-margin support vector machine

(4) [3 pts] With the SVD, we write $X = UDV^\top$. For which of the following matrices are the eigenvectors the columns of U ?

- $X^\top X$
- XX^\top
- $X^\top XX^\top X$
- $XX^\top XX^\top$

(5) [3 pts] Why is PCA sometimes used as a preprocessing step before regression?

- To reduce overfitting by removing poorly predictive dimensions.
- To make computation faster by reducing the dimensionality of the data.
- To expose information missing from the input data.
- For inference and scientific discovery, we prefer features that are not axis-aligned.

(6) [3 pts] Consider the matrix $X = \sum_{i=1}^r \alpha_i u_i v_i^\top$ where each α_i is a scalar and each u_i and v_i is a vector. It is possible that the rank of X might be

- $r + 1$
- $r - 1$
- r
- 0

(7) [3 pts] Why would we use a random forest instead of a decision tree?

- For lower training error.
- To better approximate posterior probabilities.
- To reduce the variance of the model.
- For a model that is easier for a human to interpret.

(8) [3 pts] What tends to be true about increasing the k in k -nearest neighbors?

- The decision boundary tends to get smoother.
- The bias tends to increase.
- The variance tends to increase.
- As the number of sample points approaches infinity (with $n/k \rightarrow \infty$), the error rate approaches less than twice the Bayes risk (assuming training and test points are drawn independently from the same distribution).

(9) [3 pts] Which of the following statements are true about the entropy of a discrete probability distribution?

- It is a useful criterion for picking splits in decision trees.
- It is maximized when the probability distribution is uniform.
- It is a convex function of the class probabilities.
- It is minimized when the probability distribution is uniform.

(10) [3 pts] A low-rank approximation of a matrix can be useful for

- removing noise.
- filling in unknown values.
- discovering latent categories in the data.
- matrix compression.

(11) [3 pts] Let L be the Laplacian matrix of a graph with n vertices. Let

$$\beta = \min_{\substack{y \in \mathbb{R}^n \\ \forall i, y_i \in \{-1, +1\} \\ \mathbf{1}^T y = 0}} y^T L y \quad \text{and} \quad \gamma = \min_{\substack{y \in \mathbb{R}^n \\ |y|^2 = n \\ \mathbf{1}^T y = 0}} y^T L y.$$

Which of the following are true for every Laplacian matrix L ?

- $\beta \geq \gamma$
- $\beta \leq \gamma$
- $\beta > \gamma$
- $\beta < \gamma$

(12) [3 pts] Which of the following are true about decision trees?

- They can be used only for classification.
- All the leaves must be pure.
- The tree depth never exceeds $O(\log n)$ for n sample points.
- Pruning usually achieves better test accuracy than stopping early.

(13) [3 pts] Which of the following is an effective way of reducing overfitting in neural networks?

- Augmenting the training data with similar synthetic examples
- Increasing the number of layers
- Weight decay (i.e., ℓ_2 regularization)
- Dropout

(14) [3 pts] If the VC dimension of a hypothesis class \mathcal{H} is an integer $D < \infty$ (i.e., $\text{VC}(\mathcal{H}) = D$), this means

- there exists some set of D points shattered by \mathcal{H} .
- no set of $D + 1$ points is shattered by \mathcal{H} .
- all sets of D points are shattered by \mathcal{H} .
- $\Pi_{\mathcal{H}}(D) = 2^D$.

- (15) [3 pts] Consider the minimizer w^* of the ℓ_2 -regularized least squares objective $J(w) = \|Xw - y\|^2 + \lambda \|w\|^2$ with $\lambda > 0$. Which of the following are true?
- $Xw^* = y$
 - w^* exists if and only if $X^T X$ is nonsingular
 - $w^* = X^+ y$, where X^+ is the pseudoinverse of X
 - The minimizer w^* is unique
- (16) [3 pts] You are training a neural network, but the training error is high. Which of the following, if done in isolation, has a better-than-tiny chance of reducing the training error?
- Adding another hidden layer
 - Adding more units to hidden layers
 - Normalizing the input data
 - Training on more data
- (17) [3 pts] Filters in the **late** layers of a convolutional neural network designed to classify objects in photographs likely represent
- edge detectors.
 - concepts such as “this image contains wheels.”
 - concepts such as “there is an animal.”
 - concepts such as “Jen is flirting with Dan.”
- (18) [3 pts] Which of the following techniques usually speeds up the training of a sigmoid-based neural network on a classification task?
- Using batch descent instead of stochastic
 - Having a good initialization of the weights
 - Increasing the learning rate with every iteration
 - Using the cross-entropy loss instead of the mean squared error
- (19) [3 pts] In a soft-margin support vector machine, decreasing the slack penalty term C causes
- more overfitting.
 - a smaller margin.
 - less overfitting.
 - less sensitivity to outliers.
- (20) [3 pts] The shortest distance from a point z to a hyperplane $w^T x = 0$ is
- $w^T z$
 - $\frac{w^T z}{|w|^2}$
 - $\frac{w^T z}{|w|}$
 - $|w| \cdot |z|$
- (21) [3 pts] The Bayes decision rule
- does the best a classifier can do, in expectation
 - chooses the class with the greatest posterior probability, if we use the 0-1 risk function
 - can be computed exactly from a large sample
 - minimizes the risk functional
- (22) [3 pts] Which of the following are techniques commonly used in training neural nets?
- linear programming
 - Newton’s method
 - backpropagation
 - cross-validation

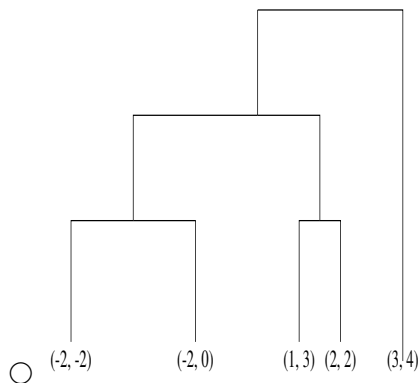
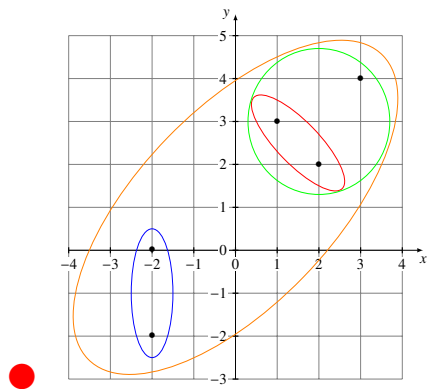
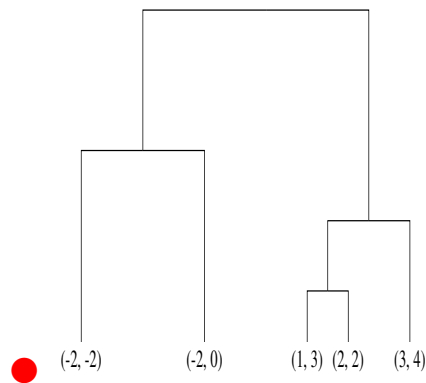
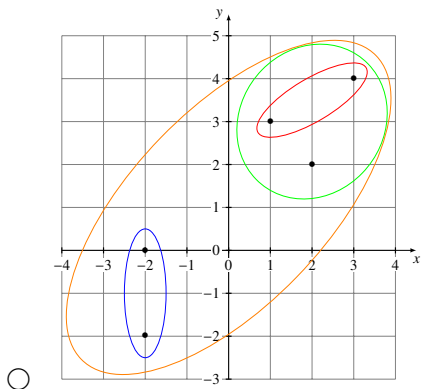
(23) [3 pts] Which of these statements about learning theory are correct?

- The VC dimension of halfplanes is 3.
- For a fixed set of training points, the more dichotomies Π we have, the higher the probability that the training error is close to the true risk.
- The VC dimension of halfspaces in 3D is ∞ .
- For a fixed hypothesis class \mathcal{H} , the more training points we have, the higher the probability that the training error is close to the true risk.

(24) [3 pts] Which of the following statements are true for a design matrix $X \in \mathbb{R}^{n \times d}$ with $d > n$? (The rows are n sample points and the columns represent d features.)

- Least-squares linear regression computes the weights $w = (X^T X)^{-1} X^T y$.
- X has exactly $d - n$ eigenvectors with eigenvalue zero.
- The sample points are linearly separable.
- At least one principal component direction is orthogonal to a hyperplane that contains all the sample points.

(25) [3 pts] Which of the following visuals accurately represent the clustering produced by greedy agglomerative hierarchical clustering with centroid linkage on the set of feature vectors $\{(-2, -2), (-2, 0), (1, 3), (2, 2), (3, 4)\}$?



(26) [3 pts] Which of the following statements is true about the standard k -means clustering algorithm?

- The random partition initialization method usually outperforms the Forgy method.
- After a sufficiently large number of iterations, the clusters will stop changing.
- It is computationally infeasible to find the optimal clustering of $n = 15$ points in $k = 3$ clusters.
- You can use the metric $d(x, y) = \frac{x \cdot y}{|x| \cdot |y|}$.

Q2. [9 pts] A Miscellany

- (a) [3 pts] Consider a convolutional neural network for reading the handwritten MNIST letters, which are 28×28 images. Suppose the first hidden layer is a convolutional layer with 20 different 5×5 filters, applied to the input image with a stride of 1 (i.e., every filter is applied to *every* 5×5 patch of the image, with patches allowed to overlap). Each filter has a bias weight. How many weights (parameters) does this layer use?

$$20 \times (5 \times 5 + 1) = 520.$$

- (b) [3 pts] Let X be an $n \times d$ design matrix representing n sample points in \mathbb{R}^d . Let $X = UDV^T$ be the singular value decomposition of X . We stated in lecture that row i of the matrix UD gives the coordinates of sample point X_i in principal coordinates space, i.e., $X_i \cdot v_j$ for each j , where X_i is the i th row of X and v_j is the j th column of V . Show that this is true.

As V is an orthogonal matrix, we can write $XV = UDV^T V = UD$.

By the definition of matrix multiplication, $(UD)_{ij} = X_i \cdot v_j$.

- (c) [3 pts] Let $x, y \in \mathbb{R}^d$ be two points (e.g., sample or test points). Consider the function $k(x, y) = x^T \text{rev}(y)$ where $\text{rev}(y)$ reverses the order of the components in y . For example, $\text{rev} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}$. Show that k *cannot* be a valid kernel function.

Hint: remember how the kernel function is defined, and show a simple two-dimensional counterexample.

We have that $k((-1, 1), (-1, 1)) = -2$, but this is impossible as, if k is a valid kernel, then there is some function Φ such that $k(x, x) = \Phi(x)^T \Phi(x) \geq 0$.

Q3. [10 pts] Maximum Likelihood Estimation for Reliability Testing

Suppose we are reliability testing n units taken randomly from a population of identical appliances. We want to estimate the mean failure time of the population. We assume the failure times come from an exponential distribution with parameter $\lambda > 0$, whose probability density function is $f(x) = \lambda e^{-\lambda x}$ (on the domain $x \geq 0$) and whose cumulative distribution function is $F(x) = \int_0^x f(x) dx = 1 - e^{-\lambda x}$.

- (a) [6 pts] In an ideal (but impractical) scenario, we run the units until they all fail. The failure times are t_1, t_2, \dots, t_n .

Formulate the likelihood function $\mathcal{L}(\lambda; t_1, \dots, t_n)$ for our data. Then find the maximum likelihood estimate $\hat{\lambda}$ for the distribution's parameter.

$$\begin{aligned} \mathcal{L}(\lambda; t_1, \dots, t_n) &= \prod_{i=1}^n f(t_i) = \prod_{i=1}^n \lambda e^{-\lambda t_i} = \lambda^n e^{-\lambda \sum_{i=1}^n t_i} \\ \ln \mathcal{L}(\lambda) &= n \ln \lambda - \lambda \sum_{i=1}^n t_i \\ \frac{\partial}{\partial \lambda} \ln \mathcal{L}(\lambda) &= \frac{n}{\lambda} - \sum_{i=1}^n t_i = 0 \\ \hat{\lambda} &= \frac{n}{\sum_{i=1}^n t_i} \end{aligned}$$

- (b) [4 pts] In a more realistic scenario, we run the units for a fixed time T . We observe r unit failures, where $0 \leq r \leq n$, and there are $n - r$ units that survive the entire time T without failing. The failure times are t_1, t_2, \dots, t_r .

Formulate the likelihood function $\mathcal{L}(\lambda; n, r, t_1, \dots, t_r)$ for our data. Then find the maximum likelihood estimate $\hat{\lambda}$ for the distribution's parameter.

Hint 1: What is the probability that a unit will not fail during time T ? *Hint 2:* It is okay to define $\mathcal{L}(\lambda)$ in a way that includes contributions (densities and probability masses) that are not commensurate with each other. Then the constant of proportionality of $\mathcal{L}(\lambda)$ is meaningless, but that constant is irrelevant for finding the best-fit parameter $\hat{\lambda}$. *Hint 3:* If you're confused, for part marks write down the likelihood that r units fail and $n - r$ units survive; then try the full problem. *Hint 4:* If you do it right, $\hat{\lambda}$ will be the number of observed failures divided by the sum of unit test times.

$$\begin{aligned} \mathcal{L}(\lambda; n, r, t_1, \dots, t_r) &\propto \left(\prod_{i=1}^r f(t_i) \right) (1 - F(T))^{n-r} \\ &= \left(\prod_{i=1}^r \lambda e^{-\lambda t_i} \right) (e^{-\lambda T})^{n-r} \\ &= \lambda^r e^{-\lambda \sum_{i=1}^r t_i} e^{-\lambda(n-r)T} \\ \ln \mathcal{L}(\lambda) &= r \ln \lambda - \lambda \sum_{i=1}^r t_i - \lambda(n-r)T + \text{constant} \\ \frac{\partial}{\partial \lambda} \ln \mathcal{L}(\lambda) &= \frac{r}{\lambda} - \sum_{i=1}^r t_i - (n-r)T = 0 \\ \hat{\lambda} &= \frac{r}{\sum_{i=1}^r t_i + (n-r)T} \end{aligned}$$

Q4. [10 pts] Decision Trees

Consider the design matrix

$$\begin{bmatrix} 4 & 6 & 9 & 1 & 7 & 5 \\ 1 & 6 & 5 & 2 & 3 & 4 \end{bmatrix}^T$$

representing 6 sample points, each with two features f_1 and f_2 .

The labels for the data are

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$$

In this question, we build a decision tree of depth 2 by hand to classify the data.

- (a) [2 pts] What is the entropy at the root of the tree?

$$-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

- (b) [3 pts] What is the rule for the first split? Write your answer in a form like $f_1 \geq 4$ or $f_2 \geq 3$. Hint: you should be able to eyeball the best split without calculating the entropies.

If we sort by f_1 , the features and the corresponding labels are

$$\begin{bmatrix} 1 & 4 & 5 & 6 & 7 & 9 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

If we sort by f_2 , we have

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

The best split is $f_1 \geq 7$.

- (c) [3 pts] For each of the two treenodes after the first split, what is the rule for the second split?

For the treenode with labels (1, 1), there's no need to split again.

For the treenode with labels (0, 1, 0, 0), if we sort by f_1 , we have

$$\begin{bmatrix} 1 & 4 & 5 & 6 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

If we sort using f_2 , we get

$$\begin{bmatrix} 1 & 2 & 4 & 6 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

We easily see we should choose $f_2 \geq 2$.

- (d) [2 pts] Let's return to the root of the tree, and suppose we're incompetent tree builders. Is there a (not trivial) split at the root that would have given us an information gain of zero? Explain your answer.

Yes. The rules $f_1 \geq 5$, $f_2 \geq 3$, or $f_2 \geq 5$ would all fail to reduce the weighted average entropy below 1.

Q5. [11 pts] Bagging and Random Forests

We are building a random forest for a 2-class classification problem with t decision trees and bagging. The input is a $n \times d$ design matrix X representing n sample points in \mathbb{R}^d (quantitative real-valued features only). For the i th decision tree we create an n -point training set $X^{(i)}$ through standard bagging. At each node of each tree, we randomly select k of the features (this random subset is selected independently for each treenode) and choose the single-feature split that maximizes the information gain, compared to all possible single-feature splits on those k features. Assume that we can radix sort real numbers in linear time, and we can randomly select an item from a set in constant time.

- (a) [3 pts] Remind us how bagging works. How do we generate the data sets $X^{(i)}$? What do we do with duplicate points?

For each training set $X^{(i)}$, we select n sample points from X uniformly at random **with replacement**. Duplicate points have proportionally greater weight in the entropy (or other cost function) calculations. (We will accept an answer that states that duplicate points are treated as if they were separate points infinitesimally close together.)

- (b) [3 pts] Fill in the blanks to derive the overall running time to construct a random forest with bagging and random subset selection. Let h be the height/depth (they're the same thing) of the tallest/deepest tree in the forest. You must use the tightest bounds possible with respect to n , d , t , k , h , and n' .

Consider choosing the split at a treenode whose box contains n' sample points. We can choose the best split for these n' sample points in $O(\underline{\hspace{2cm}})$ time. Therefore, the running time **per sample point in that node** is $O(\underline{\hspace{2cm}})$.

Each sample point in $X^{(i)}$ participates in at most $O(\underline{\hspace{2cm}})$ treenodes, so each sample point contributes at most $O(\underline{\hspace{2cm}})$ to the time. Therefore, the total running time for one tree is $O(\underline{\hspace{2cm}})$.

We have t trees, so the total running time to create the random forest is $O(\underline{\hspace{2cm}})$.

The blanks in order: $O(n'k)$, $O(k)$, $O(h)$, $O(kh)$, $O(nkh)$, $O(nkht)$. They are each worth half a point.

- (c) [2 pts] If we instead use a support vector machine to choose the split in each treenode, how does that change the asymptotic **query** time to classify a test point?

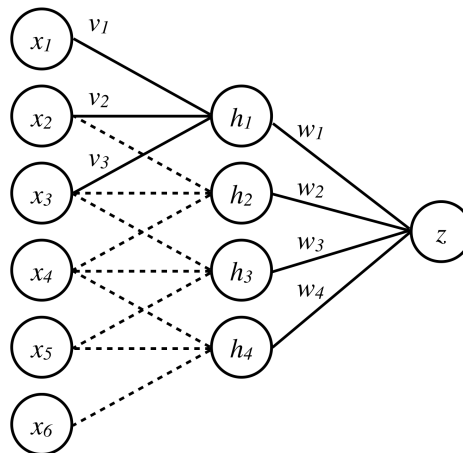
It slows queries down by a factor of $\Theta(d)$ or $\Theta(k)$ (depending whether you run the SVM on k features or all d features—we'll accept either interpretation), because it is necessary to inspect all d (or k) features of the query point at each treenode.

- (d) [3 pts] Why does bagging by itself (without random subset selection) tend **not** to improve the performance of decision trees as much as we might expect?

It is common that the same few features tend to dominate in all of the subsets, so almost all the trees will tend to have very similar early splits, and therefore all the trees will produce very similar estimates. The models are not decorrelated enough.

Q6. [11 pts] One-Dimensional ConvNet Backprop

Consider this convolutional neural network architecture.



In the first layer, we have a one-dimensional convolution with a single filter of size 3 such that $h_i = s\left(\sum_{j=1}^3 v_j x_{i+j-1}\right)$. The second layer is fully connected, such that $z = \sum_{i=1}^4 w_i h_i$. The hidden units' activation function $s(x)$ is the logistic (sigmoid) function with derivative $s'(x) = s(x)(1 - s(x))$. The output unit is linear (no activation function). We perform gradient descent on the loss function $R = (y - z)^2$, where y is the training label for x .

- (a) [1 pt] What is the total number of parameters in this neural network? Recall that convolutional layers share weights. There are no bias terms.

The answer is 7. There are 3 parameters in layer 1 and 4 parameters in layer 2.

- (b) [4 pts] Compute $\partial R / \partial w_i$.

$$\frac{\partial R}{\partial w_i} = -2(y - z)h_i$$

- (c) [1 pt] Vectorize the previous expression—that is, write $\partial R / \partial w$.

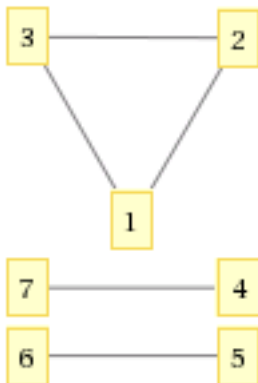
$$\frac{\partial R}{\partial w} = -2(y - z)h$$

- (d) [5 pts] Compute $\partial R / \partial v_j$.

$$\frac{\partial R}{\partial v_j} = -2(y - z) \frac{\partial z}{\partial v_j} = -2(y - z) \sum_{i=1}^4 \frac{\partial z}{\partial h_i} \frac{\partial h_i}{\partial v_j} = -2(y - z) \sum_{i=1}^4 w_i h_i (1 - h_i) x_{i+j-1}$$

Q7. [11 pts] Spectral Graph Partitioning

(a) [3 pts] Write down the Laplacian matrix L_G of the following graph G . Every edge has weight 1.



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

(b) [2 pts] Find three orthogonal eigenvectors of L_G , all having eigenvalue 0.

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, y = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

(c) [2 pts] Use two of those three eigenvectors (it doesn't matter which two) to assign each vertex of G a *spectral vector* in \mathbb{R}^2 . Draw these vectors in the plane, and explain how they partition G into three clusters. (Optional alternative: if you can draw 3D figures well, you are welcome to use all three eigenvectors and assign each vertex a spectral vector in \mathbb{R}^3 .)

The eigenvectors x and y give the embedding

$$1, 2, 3 \mapsto \begin{bmatrix} 1 \\ 0 \end{bmatrix}, 4, 7 \mapsto \begin{bmatrix} 0 \\ 1 \end{bmatrix}, 5, 6 \mapsto \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Each of the three clusters is mapped to a single point in \mathbb{R}^2 .

(d) [3 pts] Let K_n be the complete graph on n vertices (every pair of vertices is connected by an edge of weight 1) and let L_{K_n} be its Laplacian matrix. The eigenvectors of L_{K_n} are $v_1 = \mathbf{1} = [1 \ \dots \ 1]^T$ and every vector that is orthogonal to $\mathbf{1}$. What are the eigenvalues of L_{K_n} ?

$$\lambda_1 = 0 \text{ and } \lambda_2 = \dots = \lambda_n = n.$$

(e) [1 pt] What property of these eigenvalues gives us a hint that the complete graph does not have any good partitions?

λ_2 is large, so there is no low-sparsity cut. (The optimal cut has sparsity $\geq \lambda_2/2$.)

Q8. [10 pts] We Hope You Learned This

Consider learning closed intervals on the real line. Our hypothesis class \mathcal{H} consists of all intervals of the form $[a, b]$ where $a < b$ and $a, b \in \mathbb{R}$. We interpret an interval (hypothesis) $[a, b] \in \mathcal{H}$ as a classifier that identifies a point x as being in class C if $a \leq x \leq b$, and identifies x as not being in class C if $x < a$ or $x > b$.

- (a) [2 pts] Consider a set containing two distinct points on the real line. Which such sets can be shattered by \mathcal{H} ?

All sets of two distinct points can be shattered.

- (b) [2 pts] Show that no three points can be shattered by \mathcal{H} .

Let $X = \{x_1, x_2, x_3\}$ with $x_1 \leq x_2 \leq x_3$. No interval can contain x_1 and x_3 without containing x_2 . (I.e., suppose x_1 and x_3 are in class C, but x_2 is not.)

- (c) [2 pts] Write down the shatter function $\Pi_{\mathcal{H}}(n)$. Explain your answer.

$$\Pi_{\mathcal{H}}(n) = \binom{n}{2} + n + 1 = \frac{n^2 + n}{2} + 1.$$

There are $\binom{n}{2}$ dichotomies with two or more points (imagine choosing the first and the last sample point in class C), n dichotomies with one point, and one dichotomy with no points.

- (d) [2 pts] Consider another hypothesis class \mathcal{H}_2 . Each hypothesis in \mathcal{H}_2 is a union of two intervals. \mathcal{H}_2 is the set of all such hypotheses (i.e., every union of two intervals on the number line). For example, $[3, 7] \cup [8.5, 10] \in \mathcal{H}_2$; that's the set of all points x such that $3 \leq x \leq 7$ or $8.5 \leq x \leq 10$.

What is the largest number of distinct points that \mathcal{H}_2 can shatter? Explain why no larger number can be shattered.

Four. If you have five distinct points, \mathcal{H}_2 cannot include the first, third, and fifth points while excluding the second and fourth.

- (e) [2 pts] Which hypothesis class has a greater sample complexity, \mathcal{H} or \mathcal{H}_2 ? Explain why.

\mathcal{H}_2 , because its shatter function grows faster (quartic rather than quadratic) and its VC dimension is greater.

- Please do not open the exam before you are instructed to do so.
- **Electronic devices are forbidden on your person**, including cell phones, iPods, headphones, and laptops. Turn your cell phone off and **leave all electronics at the front of the room**, or **risk getting a zero** on the exam.
- 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 3 hours.
- Mark your answers on the exam itself in the space provided. Do **not** attach any extra sheets.
- The total number of points is 150. There are 26 multiple choice questions worth 3 points each, and 5 written questions worth a total of 72 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	
First and last name of student to your left	
First and last name of student to your right	

Q1. [78 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.

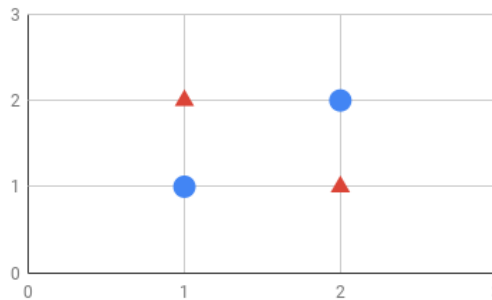
(a) [3 pts] Which of the following algorithms can learn nonlinear decision boundaries? The decision trees use only axis-aligned splits.

- A depth-five decision tree
- AdaBoost with depth-one decision trees
- Quadratic discriminant analysis (QDA)
- Perceptron

The solutions are obvious other than AdaBoost with depth-one decision trees, where you can form non-linear boundaries due to the final classifier not actually being a linear combination of the linear weak learners.

(b) [3 pts] Which of the following classifiers are capable of achieving 100% training accuracy on the data below? The decision trees use only axis-aligned splits.

- Logistic regression
- AdaBoost with depth-one decision trees
- A neural network with one hidden layer
- AdaBoost with depth-two decision trees



top left: Each weak learner will either classify the points from each pair in different classes, or classify every point in the same class. Since the meta classifier is a weighted sum of all of these weak classifiers, each which has a 50% training accuracy, the meta classifier cannot have 100% accuracy.

top right: A neural network with one hidden layer (with enough units) is a universal function approximator.

lower left: Logistic regression finds a linear decision boundary, which cannot separate the data.

lower right: A depth two decision tree can fully separate the data.

(c) [3 pts] Which of the following are true of support vector machines?

- Increasing the hyperparameter C tends to decrease the training error
- Increasing the hyperparameter C tends to decrease the margin
- The hard-margin SVM is a special case of the soft-margin with the hyperparameter C set to zero
- Increasing the hyperparameter C tends to decrease the sensitivity to outliers

Top left: True, from the lecture notes.

Bottom left: False, Hard-margin SVM is where C tends towards infinity.

Top right: false, perceptron is trained using gradient descent and SVM is trained using a quadratic program.

Bottom right: True: slack becomes less expensive, so you allow data points to be farther on the wrong side of the margin and make the margin bigger. Doing this will never reduce the number of data points inside the margin.

(d) [3 pts] Let $r(x)$ be a decision rule that minimizes the risk for a three-class classifier with labels $y \in \{0, 1, 2\}$ and an asymmetric loss function. What is true about $r(\cdot)$?

- $\forall y \in \{0, 1, 2\}, \exists x : r(x) = y$
- If we don't have access to the underlying data distribution $P(X)$ or $P(Y|X)$, we cannot exactly compute the risk of $r(\cdot)$
- $\forall x, r(x)$ is a class y that maximizes the posterior probability $P(Y = y|X = x)$
- If $P(X = x)$ changes but $P(Y = y|X = x)$ remains the same for all x, y , $r(X)$ still minimizes the risk

top left: it is possible that $r(X)$ is the same for all X .

top right: no, because the risk is asymmetric

lower left: by definition of risk we need to be able to compute expectations over these two distributions.

lower right: Given that $r(X)$ has no constraint, it can pick the y that minimizes risk for every $X = x$ without trade-offs. Therefore, if only the marginals change, that choice is not affected.

(e) [3 pts] Which of the following are true about two-class Gaussian discriminant analysis? Assume you have estimated the parameters $\hat{\mu}_C, \hat{\Sigma}_C, \hat{\pi}_C$ for class C and $\hat{\mu}_D, \hat{\Sigma}_D, \hat{\pi}_D$ for class D.

- If $\hat{\mu}_C = \hat{\mu}_D$ and $\hat{\pi}_C = \hat{\pi}_D$, then the LDA and QDA classifiers are identical
- If $\hat{\Sigma}_C = \hat{\Sigma}_D, \hat{\pi}_C = 1/6$, and $\hat{\pi}_D = 5/6$, then the LDA and QDA classifiers are identical
- If $\hat{\Sigma}_C = I$ (the identity matrix) and $\hat{\Sigma}_D = 5I$, then the LDA and QDA classifiers are identical
- If the LDA and QDA classifiers are identical, then the posterior probability $P(Y = C|X = x)$ is linear in x

Top left: false, the covariance matrices might differ, making the QDA decision function nonlinear.

Bottom left: false, the QDA decision function is nonlinear.

Top right: correct.

Bottom right: no, the posterior is a logistic function.

(f) [3 pts] Consider an $n \times d$ design matrix X with labels $y \in \mathbb{R}^n$. What is true of fitting this data with dual ridge regression with the polynomial kernel $k(X_i, X_j) = (X_i^T X_j + 1)^p = \Phi(X_i)^T \Phi(X_j)$ and regularization parameter $\lambda > 0$?

- If the polynomial degree is high enough, the polynomial will fit the data exactly
- The algorithm solves an $n \times n$ linear system
- The algorithm computes $\Phi(X_i)$ and $\Phi(X_j)$ in $O(d^p)$ time
- When n is very large, this dual algorithm is more likely to overfit than the primal algorithm with degree- p polynomial features

Top left: see definition of dual ridge regression

Lower left: both give the same solution, no matter n !

Top right: The dual method problem of ridge regression is indeed recommended only when $d > n$. But in this case we use a Kernel, so in fact we have a number of features of $d' = d^p$!

Top right: no need! just their dot product, which can easily be obtained with $(X_i^T X_j + 1)^p$.

(g) [3 pts] Consider the kernel perceptron algorithm on an $n \times d$ design matrix X . We choose a matrix $M \in \mathbb{R}^{D \times d}$ and define the feature map $\Phi(x) = Mx \in \mathbb{R}^D$ and the kernel $k(x, z) = \Phi(x) \cdot \Phi(z)$. Which of the following are always true?

The kernel matrix is $XM^T MX^T$

The kernel matrix is $MX^T XM^T$

If the primal perceptron algorithm terminates, then the kernel perceptron algorithm terminates

If the kernel perceptron algorithm terminates, then the primal perceptron algorithm terminates

Top left: $k(x, y) = \langle Mx, My \rangle = x^T M^T M y$ is a valid kernel function since $M^T M$ is positive semidefinite for all matrices M .

Bottom left: Yes, because $K = \Phi(X)\Phi(X)^T$ and $\Phi(X) = XM^T$.

Top right: Counterexample: let one class have $(-2, 1), (1, 1)$ and the other class have $(-1, -1), (2, -1)$ and let M project the points onto the x-axis. The raw data is linearly separable but the projected data is not.

Bottom right: If w linearly separates the transformed points Mx_i , then $M^T w$ linearly separates the original points since $w^T(Mx_i) = (M^T w)^T x_i$.

(h) [3 pts] Which of the following are true of decision trees? Assume splits are binary and are done so as to maximize the information gain.

If there are at least two classes at a given node, there exists a split such that information gain is strictly positive

The deeper the decision tree is, the more likely it is to overfit

As you go down any path from the root to a leaf, the information gain at each level is non-increasing

Random forests are less likely to overfit than decision trees

Top left: false, recall example from section.

Bottom left: false, recall example from section.

Top right: correct.

Bottom right: correct.

(i) [3 pts] While solving a classification problem, you use a pure, binary decision tree constructed by the standard greedy procedure we outlined in class. While your training accuracy is perfect, your validation accuracy is unexpectedly low. Which of the following, in isolation, is likely to improve your validation accuracy in most real-world applications?

Lift your data into a quadratic feature space

Normalize each feature to have variance 1

Select a random subset of the features and use only those in your tree

Prune the tree, using validation to decide how to prune

Top left: False, lifting to a more complex feature space will not generally stop you from overfitting.

Bottom left: False, an ensemble of standard decision trees fit to the same data-set will not learn

Top right: The small change in split criterion will not generally stop you from overfitting.

Bottom right: Correct, lowering depth defends against overfitting.

(j) [3 pts] For the sigmoid activation function and the ReLU activation function, which of the following are true in general?

Both activation functions are monotonically non-decreasing

Compared to the sigmoid, the ReLU is more computationally expensive

Both functions have a monotonic first derivative

The sigmoid derivative $s'(\gamma)$ is quadratic in $s(\gamma)$

- a True. Simply graph the activation functions
- b False. Sigmoid has non-monotonic derivative
- c False. ReLU is simpler as all positives have derivative 1 and all negatives have 0. While we have to calculate exponential for Sigmoid
- d True. as ReLU vanish all the negatives into zero, implicitly introducing sparsity in the network
- e True. Unlike Sigmoid, the product of gradients of ReLU function doesn't end up converging to 0 as the value is either 0 or 1

(k) [3 pts] Which of the following are true in general for backpropagation?

- It is a dynamic programming algorithm
- The weights are initially set to zero
- Some of the derivatives cannot be fully computed until the backward pass
- Its running time grows exponentially in the number of layers

- a False. As it is not a model, but a quick algorithm to compute derivatives in the network
- b True. We have a forward pass and a backward pass
- c False. Linear time complexity instead
- d False. The weights set randomly
- e True. In the backward pass

(l) [3 pts] Facets of neural networks that have (reasonable, though not perfect) analogs in human brains include

- backpropagation
- convolutional masks applied to many patches
- linear combinations of input values
- edge detectors

(m) [3 pts] Which of the following are true of the vanishing gradient problem for sigmoid units?

- Deeper neural networks tend to be more susceptible to vanishing gradients
- Using ReLU units instead of sigmoid units can reduce this problem
- If a unit has the vanishing gradient problem for one training point, it has the problem for every training point
- Networks with sigmoid units don't have this problem if they're trained with the cross-entropy loss function

Top left: false, as the number of layers goes up, the gradient is more likely to vanish during backpropagation. If one node yields a gradient close to zero, the gain of the nodes in the previous layers will also be very low.

Bottom left: true, ReLU is generally better since its gradient does not go to zero as the input goes to zero.

Top right: false, if gradients are vanishing, the weights have already effectively stopped changing their values.

Bottom right: true, this is the incentive for ResNets.

(n) [3 pts] Suppose our input is two-dimensional sample points, with ten non-exclusive classes those points may belong to (i.e., a point can belong to more than one class). To train a classifier, we build a fully-connected neural network (with bias terms) that has a single hidden layer of twenty units and an output layer of ten units (one for each class). Which statements apply?

- For the output units, softmax activations are more appropriate than sigmoid activations
- For the hidden units, ReLU activations are more appropriate than linear activations
- This network will have 240 trainable parameters
- This network will have 270 trainable parameters

Softmax will create a valid probability distribution across all the outputs, making it well suited to predicting the single class a point is most likely to belong to but not to predicting whether or not the point is in each class. Sigmoid will give us a valid in-class probability for each class independently, allowing us to perform multiclass predictions.

There are $2 * 20 + 20 = 60$ parameters in the first layer and $20 * 10 + 10 = 210$ in the second, giving us a total of 270 parameters.

With a linear activation on the hidden layer, this network reduces to a perceptron and cannot model non-linear decision boundaries.

Randomly initializing the weight terms breaks the symmetry of the network; its okay (and in fact standard practice) to initialize the bias terms to zero.

(o) [3 pts] Which of the following can lead to valid derivations of PCA?

- Fit the mean and covariance matrix of a Gaussian distribution to the sample data with maximum likelihood estimation
- Find the direction w that minimizes the sum of projection distances
- Find the direction w that minimizes the sample variance of the projected data
- Find the direction w that minimizes the sum of squares of projection distances

This is best explained by the lecture notes - in particular, lecture note 20 from the Spring 2019 iteration of the course.

(p) [3 pts] Write the SVD of an $n \times d$ design matrix X (with $n \geq d$) as $X = UDV^T$. Which of the following are true?

- The components of D are all nonnegative
- The columns of V all have unit length and are orthogonal to each other
- If X is a real, symmetric matrix, the SVD is always the same as the eigendecomposition
- The columns of D are orthogonal to each other

Top left: false, the columns of V can be used for PCA.

Bottom left: false, let the spectral decomposition be $Q\Lambda Q^{-1}$. Λ may have negative values.

Top right: correct.

Bottom right: False, a subset of the columns of V correspond to the null space of X .

(q) [3 pts] Which of the following is true about Lloyd's algorithm for k -means clustering?

- It is a supervised learning algorithm
- If run for long enough, it will always terminate
- It never returns to a particular assignment of classes to sample points after changing to another one
- No algorithm (Lloyd's or any other) can always find the optimal solution

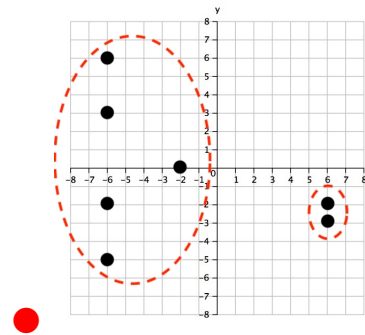
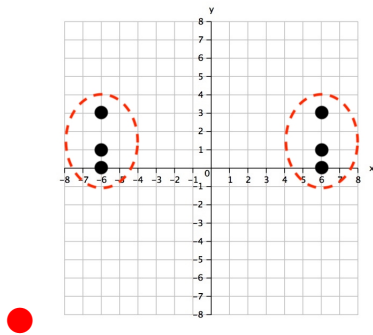
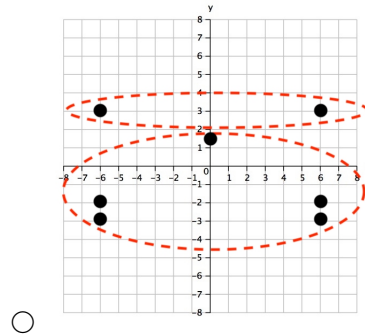
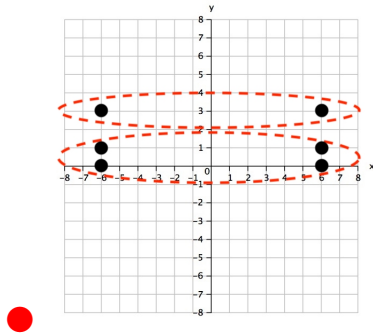
k -means is an unsupervised learning algorithm. The number of clusters k is a hyperparameter.

(r) [3 pts] Which of the following are advantages of using k -medoid clustering instead of k -means?

- k -medoids is less sensitive to outliers
- Medoids are faster to compute than means
- Medoids make more sense than means for non-Euclidean distance metrics
- The k -medoids algorithm with the Euclidean distance metric has no hyperparameters, unlike k -means

Both k means and k medoids have k as a hyperparameter. Medoids are much more expensive to compute than means (calculating all pairwise distances, rather than just summing all points and averaging).

(s) [3 pts] We wish to cluster 2-dimensional points into two clusters, so we run Lloyd's algorithm for k -means clustering until convergence. Which of the following clusters could it produce? (The points inside an oval belong to the same cluster).



(t) [3 pts] Which of the following are true of hierarchical clustering?

○ The number k of clusters is a hyperparameter

● The greedy agglomerative clustering algorithm repeatedly fuses the two clusters that minimize the distance between clusters

○ Complete linkage works only with the Euclidean distance metric

● During agglomerative clustering, single linkage is more sensitive to outliers than complete linkage

Top left: Part of the point of hierarchy is so you don't have to guess k in advance

Bottom left: Correct

Top right: Complete linkage is compatible with any distance function

Bottom right: Single linkage is very sensitive to outliers

(u) [3 pts] Which of the following are true of spectral clustering?

○ The Fiedler vector is the eigenvector associated with the second largest eigenvalue of the Laplacian matrix

● Nobody knows how to find the sparsest cut in polynomial time

● The relaxed optimization problem for partitioning a graph involves minimizing the Rayleigh quotient of the Laplacian matrix and an indicator vector (subject to a constraint)

○ The Laplacian matrix of a graph is invertible

Top left: The Fiedler vector corresponds to the second smallest eigenvalue.

Bottom left: The relaxed optimization problem minimizes the rayleigh quotient with constraints.

Top right: It's NP-hard.

Bottom right: the laplacian is never invertible; $\mathbf{1}$ is always in the nullspace.

(v) [3 pts] For binary classification, which of the following statements are true of AdaBoost?

- It can be applied to neural networks
- It uses the majority vote of learners to predict the class of a data point
- The metalearner provides not just a classification, but also an estimate of the posterior probability
- The paper on AdaBoost won a Gödel Prize

(w) [3 pts] For binary classification, which of the following statements are true of AdaBoost with decision trees?

- It usually has lower bias than a single decision tree
- It is popular because it usually works well even before any hyperparameter tuning
- To use the weight w_i of a sample point X_i when training a decision tree G , we scale the loss function $L(G(X_i), y_i)$ by w_i
- It can train multiple decision trees in parallel

(x) [3 pts] Which of the following are reasons one might choose latent factor analysis (LFA) over k -means clustering to group together n data points in \mathbb{R}^d ?

- LFA is not sensitive to how you initialize it, whereas Lloyd's algorithm is
- LFA allows us to consider points as belonging to multiple "overlapping" clusters, whereas in k -means, each point belongs to only one cluster
- In market research, LFA can distinguish different consumer types, whereas k -means cannot
- k -means requires you to guess k in advance, whereas LFA makes it easier to infer the right number of clusters after the computation

The first choice is true due to the curse of dimensionality. The second one is false: LFA is more expensive than k -means. For I iterations and k clusters, k -means runs in $O(nkID)$ time, whereas SVD takes $O(\min(dn^2, d^2n))$ time. The third one is true. We can measure how much a user vector belongs to a particular cluster by taking its inner product with the corresponding singular vector. The fourth one is false because of the above application.

(y) [3 pts] Which of the following are true for k -nearest neighbor classification?

- It is more likely to overfit with $k = 1$ (1-NN) than with $k = 1,000$ (1,000-NN)
- In very high dimensions, exhaustively checking every training point is often faster than any widely used competing exact k -NN query algorithm
- If you have enough training points drawn from the same distribution as the test points, k -NN can achieve accuracy almost as good as the Bayes decision rule
- The optimal running time to classify a point with k -NN grows linearly with k

Top left: correct; smaller k 's overfit more.

Bottom left: empirical fact.

Top right: true; Fix & Hodges, 1951.

Bottom right: false, it's poly.

(z) [3 pts] Suppose we use the k -d tree construction and query algorithms described in class to find the *approximate* nearest neighbor of a query point among n sample points. Select the true statements.

- It is possible to guarantee that the tree has $O(\log n)$ depth by our choice of splitting rule at each treenode
- Sometimes we permit the k -d tree to be unbalanced so we can choose splits with better information gain
- Querying the k -d tree is faster than querying a Voronoi diagram for sample points in \mathbb{R}^2
- Sometimes the query algorithm declines to search inside a box that's closer to the query point than the nearest neighbor it's found so far

Q2. [17 pts] Getting Down(hill) with the Funk Function

The Netflix Prize was an open competition for the best *collaborative filtering* algorithm to predict user ratings for films. Competitors were given an $n \times d$ ratings matrix R ; entry R_{jk} is user j 's rating of movie k . Because users only watch a small fraction of the movies, most entries in R are unobserved, hence filled with a default value of zero. Latent factor analysis attempts to predict missing ratings by replacing R with a low-rank approximation, which is a truncated singular value decomposition (SVD).

- (a) [4 pts] Given the SVD $R = UDV^T$, write a formula for the rank- r truncated SVD R' for comparison; make sure you explain your notation. Then write the standard restrictions (imposed by the definition of SVD) on U , D , and V .

The rank- r truncated SVD of R is $R' = \sum_{i=1}^r \delta_i u_i v_i^T$, where u_i is column i of U and v_i is column i of V . The standard restrictions are $U^T U = I$, $V^T V = I$, and D is a diagonal matrix with nonnegative components.

LFA leaves plenty of room for improvement. Simon Funk (a pseudonym, but a real person), who at one point was ranked third in the competition, developed a method called "Funk SVD." Recall that the rank- r truncated SVD R' minimizes the Frobenius norm $\|R - R'\|_F$, subject to the constraint that R' has rank r . Mr. Funk modified this approach to learn two matrices $A \in \mathbb{R}^{n \times r}$ and $B \in \mathbb{R}^{r \times d}$ such that $AB \approx R$. The rank of AB cannot exceed r . Let a_j be the j th row of A , let b_k be the k th column of B , and observe that $(AB)_{jk} = a_j \cdot b_k$. Mr. Funk solves the problem of finding matrices A and B that minimize the objective function

$$L(A, B) = \sum_{j,k: R_{jk} \neq 0} (R_{jk} - a_j \cdot b_k)^2.$$

The key difference between this objective function and the one optimized by the truncated SVD is that the summation is over **only nonzero** components of R . Instead of computing an SVD, Mr. Funk minimizes this objective with gradient descent.

- (b) [2 pts] Explain why the optimal solution is not unique; that is, there is more than one pair of optimal matrices (A, B) .

If $AB = R$, then $(2A)(\frac{1}{2}B) = R$ too.

- (c) [5 pts] Although Mr. Funk uses stochastic gradient descent, we will derive a batch gradient descent algorithm. It turns out to be easiest to write the update rule for A one row at a time. State the gradient descent rule for updating row a_j during the minimization of Mr. Funk's objective function $L(A, B)$. Use some step size $\epsilon > 0$. (Be careful that you sum only the correct terms!) (Note: there is a symmetric rule for updating b_k ; the algorithm must update both A and B .)

$$\nabla_{a_j} L(A, B) = -2 \sum_{k: R_{jk} \neq 0} (R_{jk} - a_j \cdot b_k) b_k.$$

Hence the gradient descent update is

$$a_j \leftarrow a_j + 2\epsilon \sum_{k: R_{jk} \neq 0} (R_{jk} - a_j \cdot b_k) b_k.$$

(You may omit the factor of 2.)

- (d) [3 pts] What will happen if you initialize Funk SVD by setting $A \leftarrow 0$ and $B \leftarrow 0$? Suggest a better initialization.

If the matrices are initialized to zero, the gradient descent rule cannot make them nonzero.

There are many better initializations. You could use $A \leftarrow UD$ and $B \leftarrow V^T$, or better yet, $A \leftarrow UD^{1/2}$ and $B \leftarrow D^{1/2}V^T$. A random initialization will generally work okay.

Note that a choice in which all the components of a_j are the same and all the components of b_k are the same will not work.

- (e) [3 pts] Consider the special case where $r = 1$ and the matrix R has no zero entries. In this case, what is the relationship between an optimal solution A, B and the rank-one truncated singular value decomposition?

$$AB = \delta_1 u_1 v_1^T.$$

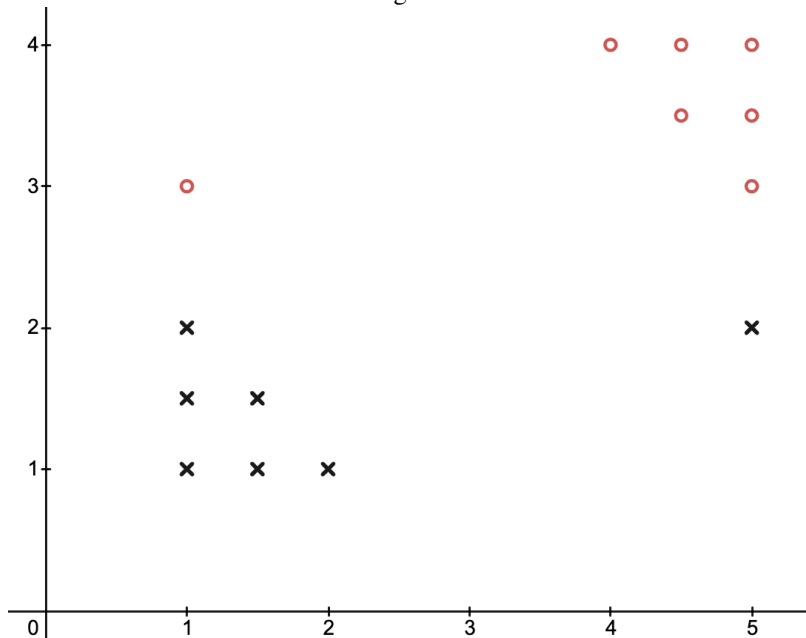
(Because the rank-1 truncated SVD is $\delta_1 u_1 v_1^T$, and it minimizes the Frobenius norm reconstruction error. This is equivalent to Mr. Funk's objective when R has no zeros.)

Q3. [10 pts] Decision Boundaries

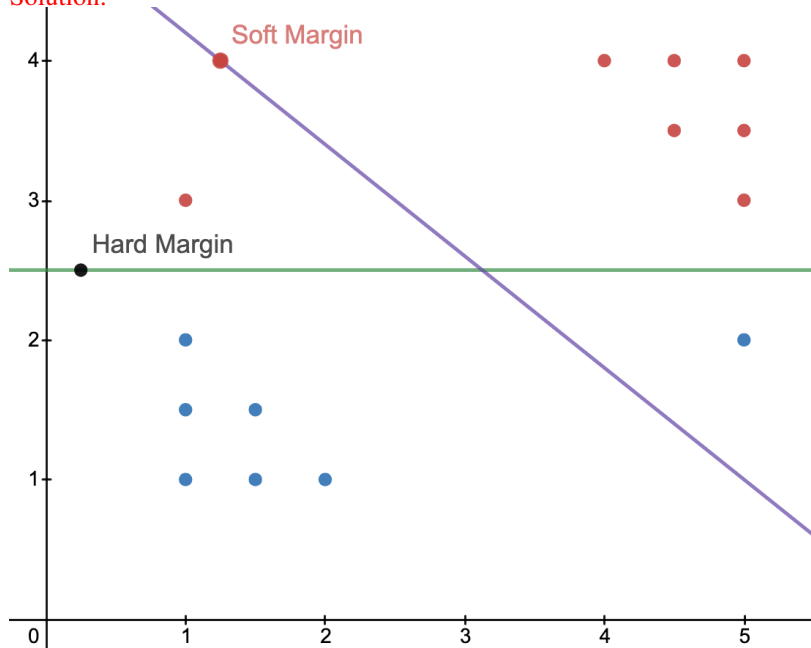
In the question, you will draw the decision boundaries that classifiers would learn.

- (a) [6 pts] Given the sample points below, draw and label **two lines**: the decision boundary learned by a hard-margin SVM and the decision boundary learned by a soft-margin SVM. We are not specifying the hyperparameter C , but don't make C too extreme. (We are looking for a qualitative difference between hard- and soft-margin SVMs.) **Label the two lines clearly.**

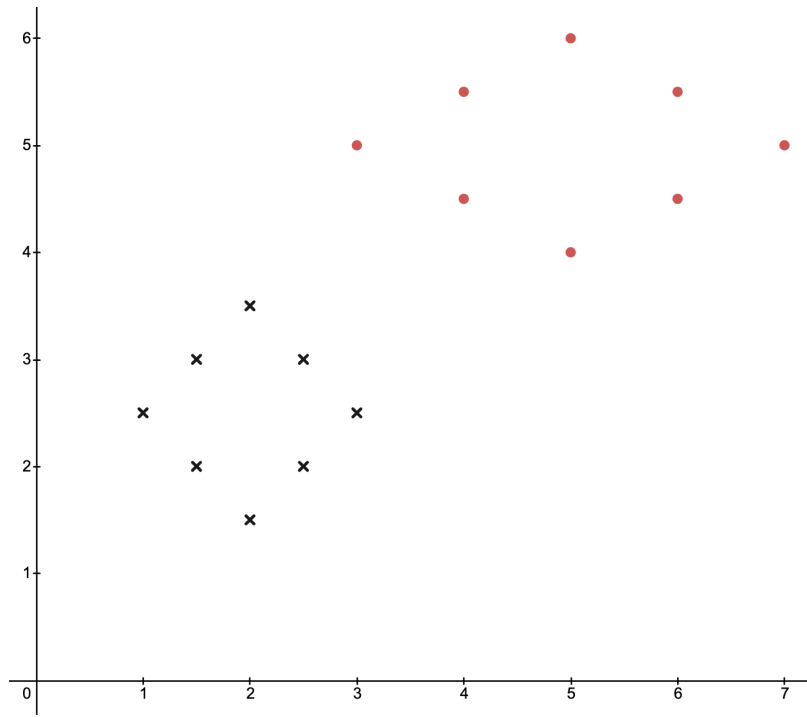
Also draw and label **four dashed lines** to show the margins of both SVMs.



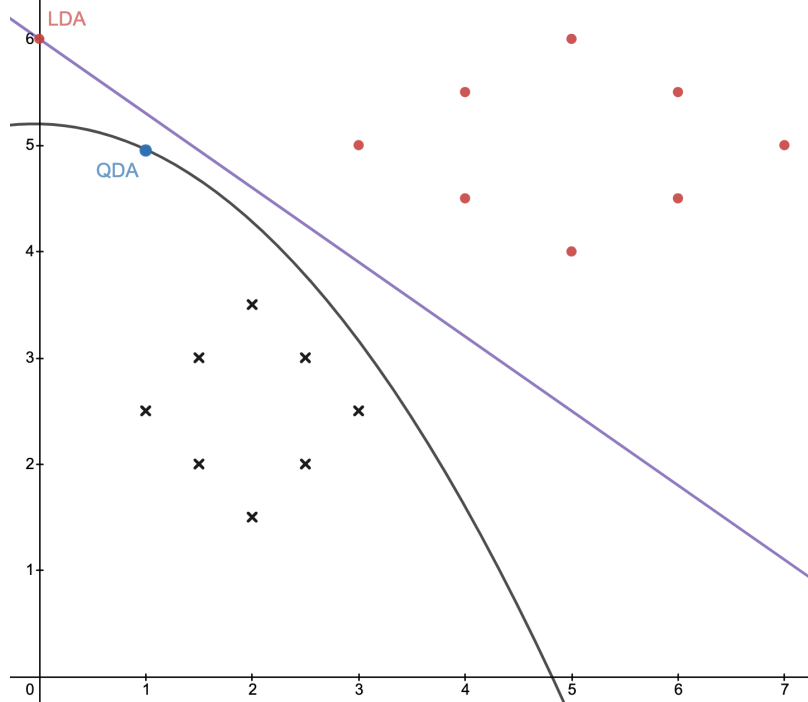
Solution:



- (b) [4 pts] Given the sample points below, draw and label **two curves**: the decision boundary learned by LDA and the decision boundary learned by QDA. Label the two curves clearly.



Solution:



Q4. [16 pts] Kernel Principal Components Analysis

Let X be an $n \times d$ design matrix. Suppose that X has been centered, so the sample points in X have mean zero. In this problem we consider kernel PCA and show that it equates to solving a *generalized Rayleigh quotient problem*.

- (a) [1 pt] Fill in the blank: every principal component direction for X is an eigenvector of _____ . $X^T X$
- (b) [1 pt] Fill in the blank: an optimization problem can be kernelized only if its solution w is always a linear combination of the sample points. In other words, we can write it in the form $w =$ _____ .
 $X^T a$ (for some vector a).
- (c) [4 pts] Show that every principal component direction w with a nonzero eigenvalue is a linear combination of the sample points (even when $n < d$).

As w is an eigenvector of $X^T X$, there is a $\lambda \in \mathbb{R}$ such that $X^T X w = \lambda w$. Setting $a = \frac{1}{\lambda} X w$, we have $X^T a = w$.

- (d) [4 pts] Let $\Phi(z)$ be a feature map that takes a point $z \in \mathbb{R}^d$ and maps it to a point $\Phi(z) \in \mathbb{R}^D$, where D might be extremely large or even infinite. But suppose that we can compute the kernel function $k(x, z) = \Phi(x) \cdot \Phi(z)$ much more quickly than we can compute $\Phi(x)$ directly. Let $\Phi(X)$ be the $n \times D$ matrix in which each sample point is replaced by a featurized point. By our usual convention, row i of X is X_i^T , and row i of $\Phi(X)$ is $\Phi(X_i)^T$.

Remind us: what is the kernel matrix K ? Answer this two ways: explain the relationship between K and the kernel function $k(\cdot, \cdot)$; then write the relationship between K and $\Phi(X)$. Lastly, show that these two definitions are equivalent.

K is the $n \times n$ matrix with components $K_{ij} = k(X_i, X_j)$. Also, $K = \Phi(X)\Phi(X)^T$.

These two characterizations are equivalent because $K_{ij} = k(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j)$ is the inner product of row i of $\Phi(X)$ and column j of $\Phi(X)^T$, which implies that $K = \Phi(X)\Phi(X)^T$.

- (e) [2 pts] Fill in the space: the first principle component direction of the *featurized* design matrix $\Phi(X)$ is any nonzero vector $w \in \mathbb{R}^D$ that maximizes the *Rayleigh quotient*, which is _____ .

$$\frac{w^T \Phi(X)^T \Phi(X) w}{w^T w}$$

- (f) [4 pts] Show that the problem of maximizing this Rayleigh quotient is equivalent to maximizing

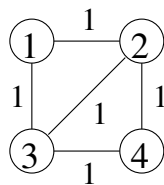
$$\frac{a^T B a}{a^T C a}$$

for some positive semidefinite matrices $B, C \in \mathbb{R}^{n \times n}$, where $a \in \mathbb{R}^n$ is a vector of dual weights. This expression is called a *generalized Rayleigh quotient*. What are the matrices B and C ? For full points, express them in a form that does not require any direct computation of the feature vectors Φ , which could be extremely long.

$$\frac{w^T \Phi(X)^T \Phi(X) w}{w^T w} = \frac{a^T \Phi(X)\Phi(X)^T \Phi(X)\Phi(X)^T a}{a^T \Phi(X)\Phi(X)^T a} = \frac{a^T K^2 a}{a^T K a}. \text{ Hence } B = K^2 \text{ and } C = K.$$

Q5. [12 pts] Spectral Graph Clustering

Let's apply spectral graph clustering to this graph.



- (a) [4 pts] Write the Laplacian matrix L for this graph. All the edges have weight 1.

$$\begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

- (b) [2 pts] Consider the minimum bisection problem, where we find an indicator vector y that minimizes $y^T L y$, subject to the balance constraint $1^T y = 0$ and the strict binary constraint $\forall i, y_i = 1$ or $y_i = -1$. Write an indicator vector y that represents a minimum bisection of this graph.

Any one of $\begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$ or $\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$ or $\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$ or $\begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$ will do.

- (c) [4 pts] Suppose we relax (discard) the binary constraint and replace it with the weaker constraint $y^T y = \text{constant}$, permitting y to have real-valued components. (We keep the balance constraint.) What indicator vector is a solution to the relaxed optimization problem? What is its eigenvalue?

Hint: Look at the symmetries of the graph. Given that the continuous values of the y_i 's permit some of the vertices to be at or near zero, what symmetry do you think would minimize the continuous-valued cut? Guess and then check whether it's an eigenvector.

$\begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$ or $\begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ or any nonzero multiple of these. The eigenvalue is 2.

- (d) [2 pts] If we apply the sweep cut to find a cut with good sparsity, what two clusters do we get? Is it a bisection?

The sweep cut either puts vertex 1 in a subgraph by itself, or vertex 4 in a subgraph by itself. It does not choose a bisection.

Q6. [17 pts] Learning Mixtures of Gaussians with k-Means

Let $X_1, \dots, X_n \in \mathbb{R}^d$ be independent, identically distributed points sampled from a mixture of two normal (Gaussian) distributions. They are drawn independently from the probability distribution function (PDF)

$$p(x) = \theta N_1(x) + (1 - \theta) N_2(x), \quad \text{where } N_1(x) = \frac{1}{(\sqrt{2\pi})^d} e^{-\|x-\mu_1\|^2/2} \text{ and } N_2(x) = \frac{1}{(\sqrt{2\pi})^d} e^{-\|x-\mu_2\|^2/2}$$

are the PDFs for the isotropic multivariate normal distributions $\mathcal{N}(\mu_1, 1)$ and $\mathcal{N}(\mu_2, 1)$, respectively. The parameter $\theta \in (0, 1)$ is called the *mixture proportion*. In essence, we flip a biased coin to decide whether to draw a point from the first Gaussian (with probability θ) or the second (with probability $1 - \theta$).

Each data point is generated as follows. First draw a random Z_i , which has value 1 with probability θ , and has value 2 with probability $1 - \theta$. Then, draw $X_i \sim \mathcal{N}(\mu_{Z_i}, 1)$. Our learning algorithm gets X_i as an input, but does not know Z_i .

Our goal is to find the maximum likelihood estimates of the three unknown distribution parameters $\theta \in (0, 1)$, $\mu_1 \in \mathbb{R}^d$, and $\mu_2 \in \mathbb{R}^d$ from the sample points X_1, \dots, X_n . Unlike MLE for one Gaussian, it is **not** possible to give explicit analytic formulas for these estimates. Instead, we develop a variant of k -means clustering which (often) converges to the correct maximum likelihood estimates of θ , μ_1 , and μ_2 . This variant doesn't assign each point entirely to one cluster; rather, each point is assigned an estimated posterior probability of coming from normal distribution 1.

- (a) [4 pts] Let $\tau_i = P(Z_i = 1|X_i)$. That is, τ_i is the posterior probability that point X_i has $Z_i = 1$. Use Bayes' Theorem to express τ_i in terms of X_i , θ , μ_1 , μ_2 , and the Gaussian PDFs $N_1(x)$ and $N_2(x)$. To help you with part (c), also write down a similar formula for $1 - \tau_i$, which is the posterior probability that $Z_i = 2$.

Bayes' Theorem implies that

$$\tau_i = \frac{\theta N_1(X_i)}{\theta N_1(X_i) + (1 - \theta) N_2(X_i)}, \quad 1 - \tau_i = \frac{(1 - \theta) N_2(X_i)}{\theta N_1(X_i) + (1 - \theta) N_2(X_i)}.$$

- (b) [3 pts] Write down the log-likelihood function, $\ell(\theta, \mu_1, \mu_2; X_1, \dots, X_n) = \ln p(X_1, \dots, X_n)$, as a summation. Note: it doesn't simplify much.

Because the samples are iid, $p(X_1, \dots, X_n) = \prod_{i=1}^n p(X_i)$, so

$$\ell(\theta, \mu_1, \mu_2; X_1, \dots, X_n) = \sum_{i=1}^n \ln(\theta N_1(X_i) + (1 - \theta) N_2(X_i)).$$

- (c) [3 pts] Express $\frac{\partial \ell}{\partial \theta}$ in terms of θ and τ_i , $i \in \{1, \dots, n\}$ and simplify as much as possible. There should be no normal PDFs explicitly in your solution, though the τ_i 's may implicitly use them. Hint: Recall that $(\ln f(x))' = \frac{f'(x)}{f(x)}$.

$$\frac{\partial \ell}{\partial \theta} = \sum_{i=1}^n \left(\frac{\tau_i}{\theta} - \frac{1 - \tau_i}{1 - \theta} \right) = \frac{1}{\theta - \theta^2} \sum_{i=1}^n (\tau_i - \theta) = \frac{(\sum \tau_i) - \theta n}{\theta - \theta^2}.$$

(Any of these expressions is simple enough to receive full marks.)

- (d) [4 pts] Express $\nabla_{\mu_1} \ell$ in terms of μ_1 and $\tau_i, X_i, i \in \{1, \dots, n\}$. Do the same for $\nabla_{\mu_2} \ell$ (but in terms of μ_2 rather than μ_1). Again, there should be no normal PDFs explicitly in your solution, though the τ_i 's may implicitly use them. Hint: It will help (and get you part marks) to first write $\nabla_{\mu_1} N_1(x)$ as a function of $N_1(x), x$, and μ_1 .

$$\begin{aligned} \nabla_{\mu_1} \ell &= \sum_{i=1}^n \frac{\theta \nabla_{\mu_1} N_1(X_i)}{\theta N_1(X_i) + (1 - \theta) N_2(X_i)} \\ &= \sum_{i=1}^n \frac{\theta N_1(X_i)}{\theta N_1(X_i) + (1 - \theta) N_2(X_i)} (X_i - \mu_1) \\ &= \sum_{i=1}^n \tau_i (X_i - \mu_1). \end{aligned}$$

Similarly,

$$\nabla_{\mu_2} \ell = \sum_{i=1}^n (1 - \tau_i) (X_i - \mu_2).$$

- (e) [3 pts] We conclude: if we know μ_1, μ_2 , and θ , we can compute the posteriors τ_i . On the other hand, if we know the τ_i 's, we can estimate μ_1, μ_2 , and θ by using the derivatives in parts (c) and (d) to find the maximum likelihood estimates. This leads to the following k -means-like algorithm.

- Initialize $\tau_1, \tau_2, \dots, \tau_n$ to arbitrary values in the range $[0, 1]$.
- Repeat the following two steps.
 1. Update the Gaussian cluster parameters: for fixed values of $\tau_1, \tau_2, \dots, \tau_n$, update μ_1, μ_2 , and θ .
 2. Update the posterior probabilities: for fixed values of μ_1, μ_2 and θ , update $\tau_1, \tau_2, \dots, \tau_n$.

In part (a), you wrote the update rule for step 2. Using your results from parts (c) and (d), write down the explicit update formulas for step 1.

$$\mu_1 \leftarrow \frac{\sum_{i=1}^n \tau_i X_i}{\sum_{i=1}^n \tau_i}, \quad \mu_2 \leftarrow \frac{\sum_{i=1}^n (1 - \tau_i) X_i}{\sum_{i=1}^n (1 - \tau_i)}, \quad \theta \leftarrow \frac{1}{n} \sum_{i=1}^n \tau_i.$$

- The exam is open book, open notes, and open web. However, you may not consult or communicate with other people (besides your exam proctors).
- You will submit your answers to the multiple-choice questions through Gradescope via the assignment “**Final Exam – Multiple Choice**”; please **do not** submit your multiple-choice answers on paper. 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 Exam – Writeup**.”
- Please write your name at the top of each page of your written answers. (You may do this before the exam.)
- You have 180 minutes to complete the midterm exam (3:00–6:00 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 (6:00 PM), **stop writing**. You must submit your multiple-choice answers before 6:00 PM sharp. Late multiple-choice submissions will be penalized at a rate of 5 points per minute after 6:00 PM. (The multiple-choice questions are worth 60 points total.)
- From 6:00 PM, you have 15 minutes to scan the written portion of your exam and turn it into Gradescope via the assignment “Final Exam – Writeup.” Most of you will use your cellphone 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 5 points per minute after 6:15 PM.
- Mark your answers to multiple-choice questions directly into Gradescope. Write your answers to written questions on blank paper. **Clearly label all written questions and all subparts of each written question. Show your work in written questions.**
- 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).
- The total number of points is 150. There are 16 multiple choice questions worth 4 points each, and six written questions worth 86 points total.
- 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	

Q1. [64 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.

(1) [4 pts] Which of the following are true for the k -nearest neighbor (k -NN) algorithm?

- A: k -NN can be used for both classification and regression.
- C: The decision boundary looks smoother with smaller values of k .
- B: As k increases, the bias usually increases.
- D: As k increases, the variance usually increases.

(2) [4 pts] Let X be a matrix with **singular value decomposition** $X = U\Sigma V^T$. Which of the following are true for all X ?

- A: $\text{rank}(X) = \text{rank}(\Sigma)$.
- C: The first column of V is an eigenvector of $X^T X$.
- B: If all the singular values are unique, then the SVD is unique.
- D: The singular values and the eigenvalues of $X^T X$ are the same.

A is correct because the number of non-zero singular values is equal to the rank. B is incorrect because you could change both $U \rightarrow -U, V \rightarrow -V$. C is correct because the SVD and eigendecomposition of $X^T X$ is $V\Sigma^2 V^T$. D is correct as $X^T X$ is positive semidefinite, so the eigenvalues can't be negative.

(3) [4 pts] Lasso (with a fictitious dimension), random forests, and principal component analysis (PCA) **all** ...

- A: can be used for dimensionality reduction or feature subset selection
- B: compute linear transformations of the input features
- C: are supervised learning techniques
- D: are *translation invariant*: changing the origin of the coordinate system (i.e., translating all the training and test data together) does not change the predictions or the principal component directions

Option B is incorrect because random forests don't compute linear transformations. Option C is incorrect because PCA is unsupervised.

(4) [4 pts] Suppose your training set for two-class classification in one dimension ($d = 1$; $x_i \in \mathbb{R}$) contains three sample points: point $x_1 = 3$ with label $y_1 = 1$, point $x_2 = 1$ with label $y_2 = 1$, and point $x_3 = -1$ with label $y_3 = -1$. What are the values of w and b given by a **hard-margin SVM**?

- A: $w = 1, b = 1$
- C: $w = 1, b = 0$
- B: $w = 0, b = 1$
- D: $w = \infty, b = 0$

(5) [4 pts] Use the same training set as part (d). What is the value of w and b given by **logistic regression** (with no regularization)?

- A: $w = 1, b = 1$
- C: $w = 1, b = 0$
- B: $w = 0, b = 1$
- D: $w = \infty, b = 0$

(6) [4 pts] Below are some choices you might make while training a neural network. Select all of the options that will generally make it **more difficult** for your network to achieve high accuracy on the test data.

A: Initializing the weights to all zeros

C: Using momentum

B: Normalizing the training data but leaving the test data unchanged

D: Reshuffling the training data at the beginning of each epoch

A) Initializing weights with zeros makes it impossible to learn. B) Mean and standard deviation should be computed on the training set and then used to standardize the validation and test sets, so that the distributions are matched for each set. C) This describes momentum and will generally help training. D) This is best practice.

(7) [4 pts] To the left of each graph below is a number. Select the choices for which the number is the multiplicity of the eigenvalue zero in the **Laplacian matrix** of the graph.

<input type="radio"/> A: 1		<input type="radio"/> C: 2	
<input checked="" type="radio"/> B: 1		<input type="radio"/> D: 4	

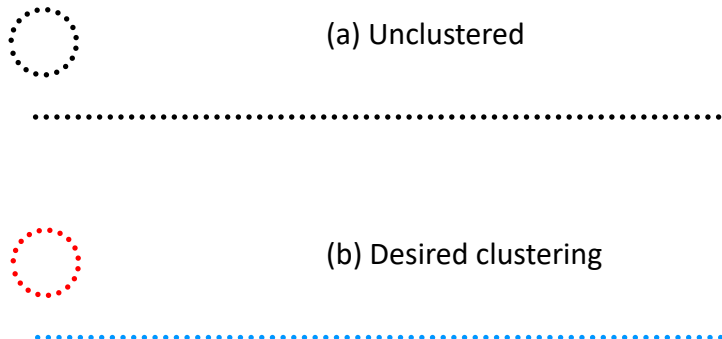
The multiplicity is equal to the number of connected components in the graph.

(8) [4 pts] Given the spectral graph clustering optimization problem
 Find y that minimizes $y^T Ly$
 subject to $y^T y = n$
 and $\mathbf{1}^T y = 0$,

which of the following optimization problems produce a vector y that leads to **the same sweep cut** as the optimization problem above? M is a diagonal mass matrix with different masses on the diagonal.

<input checked="" type="radio"/> A: Minimize $y^T Ly$ subject to $y^T y = 1$ and $\mathbf{1}^T y = 0$	<input checked="" type="radio"/> C: Minimize $y^T Ly / (y^T y)$ subject to $\mathbf{1}^T y = 0$
<input type="radio"/> B: Minimize $y^T Ly$ subject to $\forall i, y_i = 1 \text{ or } y_i = -1$ and $\mathbf{1}^T y = 0$	<input type="radio"/> D: Minimize $y^T Ly$ subject to $y^T My = 1$ and $\mathbf{1}^T My = 0$

(9) [4 pts] Which of the following methods will cluster the data in panel (a) of the figure below into the two clusters (red circle and blue horizontal line) shown in panel (b)? Every dot in the circle and the line is a data point. In all the options that involve hierarchical clustering, the algorithm is run until we obtain two clusters.



- | | |
|---|--|
| <input type="radio"/> A: Hierarchical agglomerative clustering with Euclidean distance and complete linkage | <input checked="" type="radio"/> B: Hierarchical agglomerative clustering with Euclidean distance and single linkage |
|---|--|

C: Hierarchical agglomerative clustering with Euclidean distance and centroid linkage

D: k -means clustering with $k = 2$

Single linkage uses the minimum distance between two clusters as a metric for merging clusters. Since the two clusters are densely packed with points and the minimum distance between the two clusters is greater than the within-cluster distances between points, single linkage doesn't link the circle to the line until the very end.

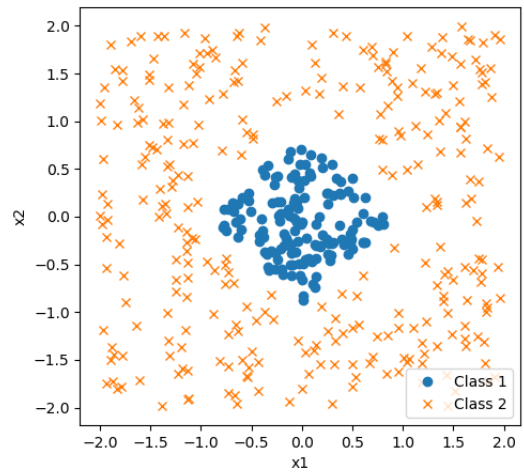
The other three methods will all join some of the points at the left end of the line with the circle, before they are joined with the right end of the line.

(10) [4 pts] Which of the following statement(s) about **kernels** are true?

- A: The dimension of the lifted feature vectors $\Phi(\cdot)$, whose inner products the kernel function computes, can be infinite.
- B: For any desired lifting $\Phi(x)$, we can design a kernel function $k(x, z)$ that will evaluate $\Phi(x)^\top \Phi(z)$ more quickly than explicitly computing $\Phi(x)$ and $\Phi(z)$.
- C: The kernel trick, when it is applicable, speeds up a learning algorithm if the number of sample points is substantially less than the dimension of the (lifted) feature space.
- D: If the raw feature vectors x, y are of dimension 2, then $k(x, y) = x_1^2 y_1^2 + x_2^2 y_2^2$ is a valid kernel.

A is correct; consider the Gaussian kernel from lecture. B is wrong; most liftings don't lead to super-fast kernels. Just some special ones do. C is correct, straight from lecture. Though in this case, the dual algorithm is faster than the primal whether you use a fancy kernel or not. D is correct because $k(x, y)$ is inner product of $\Phi(x) = [x_1^2 \ x_2^2]^\top$ and $\Phi(y) = [y_1^2 \ y_2^2]^\top$.

(11) [4 pts] We want to use a **decision tree** to classify the training points depicted. Which of the following decision tree classifiers is capable of giving 100% accuracy on the training data with **four splits or fewer**?



- A: A standard decision tree with axis-aligned splits
- B: Using PCA to reduce the training data to one dimension, then applying a standard decision tree
- C: A decision tree with multivariate linear splits
- D: Appending a new feature $|x_1| + |x_2|$ to each sample point x , then applying a standard decision tree

A standard decision tree will need (substantially) more than four splits. PCA to 1D will make it even harder. However, four non-axis-aligned multivariate linear splits suffice to cut the diamond out of the center. Finally, adding the L_1 norm feature lets us perfectly classify the data with a single split that cuts off the top of the pyramid.

(12) [4 pts] Which of the following are true about **principal components analysis (PCA)**?

- A: The principal components are eigenvectors of the centered data matrix.
- B: The principal components are right singular vectors of the centered data matrix.
- C: The principal components are eigenvectors of the sample covariance matrix.
- D: The principal components are right singular vectors of the sample covariance matrix.

The first three follow directly from definitions. The last is because the covariance matrix is symmetric, so the singular vectors are the eigenvectors.

(13) [4 pts] Suppose we are doing **ordinary least-squares linear regression** with a fictitious dimension. Which of the following changes can **never** make the cost function's value on the **training data** smaller?

- A: Discard the fictitious dimension (i.e., don't append a 1 to every sample point).
- B: Append quadratic features to each sample point.
- C: Project the sample points onto a lower-dimensional subspace with PCA (without changing the labels) and perform regression on the projected points.
- D: Center the design matrix (so each feature has mean zero).

A: Correct. Discarding the fictitious dimension forces the linear regression function to be zero at the origin, which may increase the cost function but can never decrease it.

B: Incorrect. Added quadratic features often help to fit the data better.

C: Correct. Regular OLS is at least as expressive. Projecting the points may increase the cost function but can never decrease it. Centering features doesn't matter so WLOG assume X has centered features. If the full SVD is $X = U\Sigma V^T$, then projecting onto a k -dimensional subspace gives $X_k = U\Sigma_k V^T$. If w_k is a solution for the PCA-projected OLS, we can take $w = Vz$ where z is the first k elements of $V^T w_k$ with the rest zero, and get $X_k w_k = Xw$.

D: Correct. Since we're using a fictitious dimension, translating the points does not affect the cost of the optimal regression function (which translates with the points).

(14) [4 pts] Which of the following are true about **principal components analysis** (PCA)? Assume that no two eigenvectors of the sample covariance matrix have the same eigenvalue.

- A: Appending a 1 to the end of every sample point doesn't change the results of performing PCA (except that the useful principal component vectors have an extra 0 at the end, and there's one extra useless component with eigenvalue zero).
- B: If you use PCA to project d -dimensional points down to j principal coordinates, and then you run PCA again to project those j -dimensional coordinates down to k principal coordinates, with $d > j > k$, you always get the same result as if you had just used PCA to project the d -dimensional points directly down to k principal coordinates.
- C: If you perform an arbitrary rigid rotation of the sample points as a group in feature space before performing PCA, the principal component directions do not change.
- D: If you perform an arbitrary rigid rotation of the sample points as a group in feature space before performing PCA, the largest eigenvalue of the sample covariance matrix does not change.

Appending an extra dimension with the same values introduces no variance in the extra dimension, so PCA will ignore that dimension. PCA discards the eigenvector directions associated with the largest eigenvalues; as the eigenvectors are mutually orthogonal, this does not affect the variance in the surviving dimensions, so your results depend solely on how many directions you discard. Rotating the sample points rotates the principal components, but it doesn't change the variance along each of those (rotated) component directions.

(15) [4 pts] Consider running a single iteration of **AdaBoost** on three sample points, starting with uniform weights on the sample points. All the ground truth labels and predictions are either +1 or -1. In the table below, some values have been omitted. Which of the following statements can we say **with certainty**?

	True Label	Classifier Prediction	Initial Weight	Updated Weight
X_1	-1	-1	1/3	?
X_2	?	+1	1/3	$\sqrt{2}/3$
X_3	?	?	1/3	$\sqrt{2}/6$

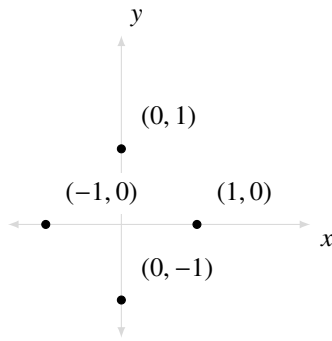
- A: X_1 's updated weight is $\sqrt{2}/6$
- C: X_2 is misclassified
- B: X_3 's classifier prediction is -1
- D: X_3 is misclassified

In the AdaBoost algorithm, all correctly classified points have their weights changed by the same multiplicative factor. Since we observe two different updated weights, we know one of x_2 or x_3 is correctly classified, and the other is misclassified. Since x_1 is correctly classified, the error rate is $\text{err} = 1/3$. As the error rate is less than 1/2, the weights of correctly classified points will decrease and the weights of misclassified points will increase. Hence, X_2 is misclassified and X_3 is correctly classified. As X_1 is correctly classified, it has the same updated weight as X_3 . But we can't tell what X_3 's classifier prediction is; only that it is correctly classified.

As an aside, we can confirm the multipliers used for reweighting of misclassified and correctly classified points (in that order):

$$\sqrt{\frac{\text{err}}{1 - \text{err}}} = \sqrt{\frac{2/3}{1/3}} = \sqrt{2} \qquad \sqrt{\frac{1 - \text{err}}{\text{err}}} = \sqrt{\frac{1/3}{2/3}} = \frac{\sqrt{2}}{2}$$

(16) [4 pts] Consider running the **hierarchical agglomerative clustering** algorithm on the following set of four points in \mathbb{R}^2 , breaking ties arbitrarily. If we stop when only two clusters remain, which of the following linkage methods *ensures* the resulting clusters are balanced (each have two sample points)? Select all that apply.



A: Complete linkage

C: Centroid linkage

B: Single linkage

D: Average linkage

Under each of these linkage methods, we know that two adjacent points along a side of the rhombus will first be fused together. Without loss of generality, assume these are the points $(0, 1)$ and $(1, 0)$. Treating these as a cluster, the average, maximum and centroid distances to each of the two remaining points are all larger than the distance between the two points themselves. Therefore, complete linkage, single linkage and average linkage all result in balanced clusters. On the other hand, single linkage does not, since, for example, $(1, 0)$ and $(0, -1)$ have the same distance as $(-1, 0)$ and $(0, -1)$.

Q2. [14 pts] Principal Components Analysis

Consider the following design matrix, representing four sample points $X_i \in \mathbb{R}^2$.

$$X = \begin{bmatrix} 4 & 1 \\ 2 & 3 \\ 5 & 4 \\ 1 & 0 \end{bmatrix}.$$

We want to represent the data in only one dimension, so we turn to principal components analysis (PCA).

- (1) [5 pts] Compute the **unit-length principal component directions** of X , and **state which one the PCA algorithm would choose** if you request just one principal component. Please provide an exact answer, without approximation. (You will need to use the square root symbol.) **Show your work!**

We center X , yielding

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

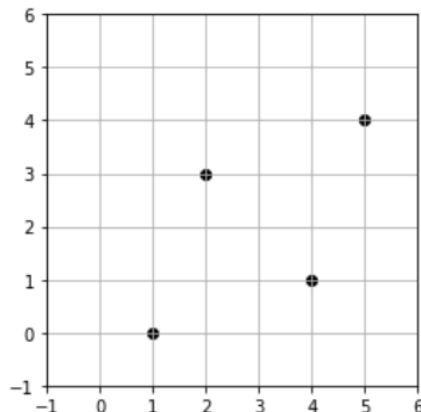
Then $\dot{X}^T \dot{X} = \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix}$. (Divide by 4 if you want the sample covariance matrix. But we don't care about the magnitude.)

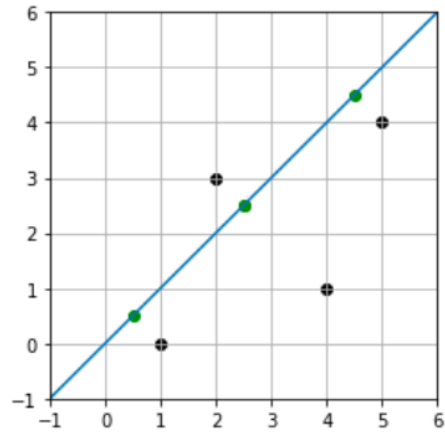
Its eigenvectors are $[1/\sqrt{2} \ 1/\sqrt{2}]^T$ with eigenvalue 16 and $[1/\sqrt{2} \ -1/\sqrt{2}]^T$ with eigenvalue 4. The former eigenvector is chosen.

(Negated versions of these vectors also get full points.)

- (2) [5 pts] The plot below depicts the sample points from X . We want a one-dimensional representation of the data, so **draw the principal component direction (as a line) and the projections of all four sample points onto the principal direction**.

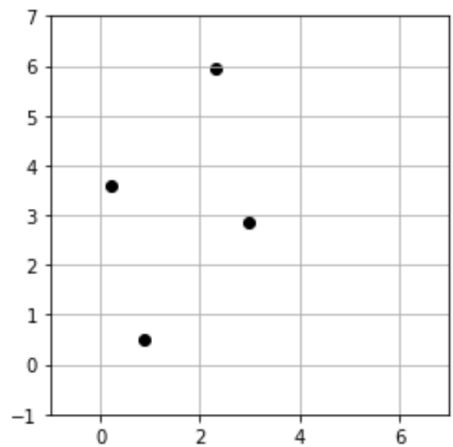
Label each projected point with its principal coordinate value (where the origin's principal coordinate is zero). Give the principal coordinate values exactly.

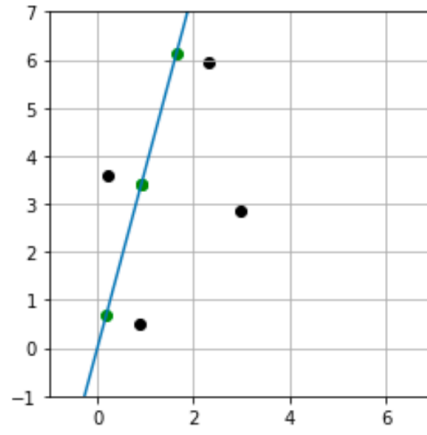




The principal coordinates are $\frac{1}{\sqrt{2}}$, $\frac{5}{\sqrt{2}}$, $\frac{5}{\sqrt{2}}$, and $\frac{9}{\sqrt{2}}$. (Alternatively, all of these could be negative, but they all have to have the same sign.)

- (3) [4 pts] The plot below depicts the sample points from X rotated 30 degrees counterclockwise about the origin. As in part (b), **identify the principal component direction that the PCA algorithm would choose and draw it (as a line) on the plot.** Also **draw the projections of the rotated points onto the principal direction.** **Label each projected point with the exact value of its principal coordinate.**





The line passes through the origin and is parallel to the two sample points that are farthest apart, so it's easy to draw. Rotation has not changed the principal coordinates: $\frac{1}{\sqrt{2}}$, $\frac{5}{\sqrt{2}}$, $\frac{5}{\sqrt{2}}$, and $\frac{9}{\sqrt{2}}$. (Again, these could all be negative.)

Q3. [14 pts] A Decision Tree

In this question we investigate whether students will pass or fail CS 189 based on whether or not they studied, cheated, and slept well before the exam. You are given the following data for five students. There are three features, “Studied,” “Slept,” and “Cheated.” The column “Result” shows the label we want to predict.

	Studied	Slept	Cheated	Result
Student 1	Yes	No	No	Passed
Student 2	Yes	No	Yes	Failed
Student 3	No	Yes	No	Failed
Student 4	Yes	Yes	Yes	Failed
Student 5	Yes	Yes	No	Passed

- (1) [4 pts] What is the **entropy** $H(\text{Result})$ at the root node? (There is no need to compute the exact number; you may write it as an arithmetic expression.)

$$H(\text{Result}) = -\left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5}\right).$$

- (2) [5 pts] **Draw the decision tree** where every split maximizes the information gain. (An actual drawing, please; a written description does not suffice.) Do not perform a split on a pure leaf or if the split will produce an empty child; otherwise, split. **Explain (with numbers)** why you chose the splits you chose.

A tree that first splits on “Cheated” and then “Studied.”

- (3) [2 pts] Did the tree you built implicitly perform feature subset selection? **Explain.**

Yes, because it does not use the feature “Slept.”

- (4) [3 pts] Suppose you have a sample of n students for some large n , with the same three features. Assuming that we use a reasonably efficient algorithm to build the tree (as discussed in class), what is the **worst-case running time to build** the decision tree? (Write your answer in the simplest asymptotic form possible.) **Why?**

We have 3 binary features, so the tree’s depth cannot exceed 3 and each sample point participates in at most four treenodes. Hence, it cannot take more than $\Theta(n)$ time to build the tree.

Q4. [20 pts] Spectral Graph Clustering

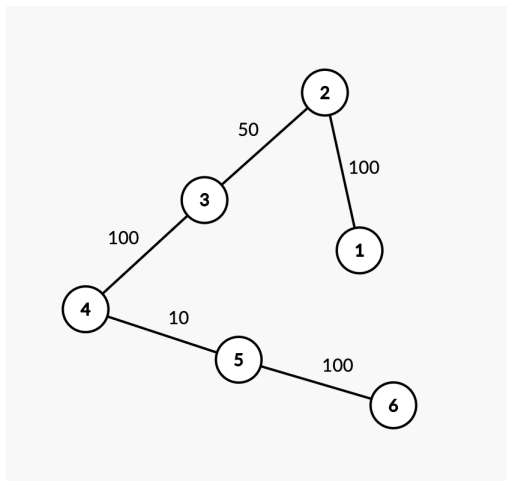


Figure 1: An undirected, weighted graph in which all vertices have mass 1. The numbers inside the vertices are their indices (not masses).

In this problem, we approximate the sparsest cut of the graph above with the spectral graph clustering algorithm. Recall the spectral graph clustering optimization objective is to

$$\begin{aligned} &\text{find } y \text{ that minimizes } y^\top L y \\ &\text{subject to } y^\top y = 6 \\ &\text{and } \mathbf{1}^\top y = 0. \end{aligned}$$

(1) [4 pts] Write out the **Laplacian matrix** L .

$$\begin{bmatrix} 100 & -100 & 0 & 0 & 0 & 0 \\ -100 & 150 & -50 & 0 & 0 & 0 \\ 0 & -50 & 150 & -100 & 0 & 0 \\ 0 & 0 & -100 & 110 & -10 & 0 \\ 0 & 0 & 0 & -10 & 110 & -100 \\ 0 & 0 & 0 & 0 & -100 & 100 \end{bmatrix}$$

(2) [3 pts] What is the rank of L ? **Explain your answer.**

5, because the Laplacian matrix of a connected graph has one and only one eigenvector with eigenvalue zero.

(3) [4 pts] L has the following six unit eigenvectors, listed in random order. **Write down the Fiedler vector. Then explain how you can tell which one is the Fiedler vector without doing a full eigendecomposition computation.** (There are several ways to see this; describe one.)

- $[0.36, -0.58, 0.61, -0.4, 0.04, -0.03]$
 $[-0.56, -0.32, 0.43, 0.62, -0.06, -0.1]$
- $[0.3, -0.33, -0.23, 0.3, -0.6, 0.55]$
 $[0.36, 0.34, 0.25, 0.19, -0.55, -0.6]$
- $[-0.41, 0.41, 0.41, -0.41, -0.41, 0.41]$
 $[0.41, 0.41, 0.41, 0.41, 0.41, 0.41]$

The Fiedler vector is $v_2 = [0.36, 0.34, 0.25, 0.19, -0.55, -0.6]$. Since we know these are all eigenvectors of the Laplacian matrix L , one way to identify it is to explicitly multiply the first row of L by an eigenvector, and divide by the first component of the eigenvector to reveal its eigenvalue. (The Fiedler vector has eigenvalue $\lambda_2 = 6.5$; all the others except v_1 have larger

eigenvalues.) An easier way is to notice that only the Fiedler vector is monotonic in one direction (decreasing in this example).

- (4) [5 pts] **How does the sweep cut decide** how to cut this graph into two clusters? (Explain in clear English sentences.) For every cut considered by the algorithm, **write down the “score”** it is assigned by the sweep cut algorithm. (You may use fractions; decimal numbers aren't required.) **Identify the chosen cut** by writing down two sets of vertex indices.

The sweep cut sorts the values in the Fiedler vector, then decides which pair of consecutive vertices to cut between by explicitly computing the sparsity of each of the five possible cuts. The cut with the lowest sparsity wins.

From left to right in the Fiedler vector, the sparsity of each cut is 20 , $\frac{50}{8} = 6.25$, $\frac{100}{9} \doteq 11.11$, $\frac{10}{8} = 1.25$, and 20 .

The clusters are $\{5, 6\}$ and $\{4, 3, 1, 2\}$.

(Cut between the values 0.19 and -0.55 .)

- (5) [4 pts] Suppose we have computed the four eigenvectors v_1, v_2, v_3, v_4 corresponding to the four largest eigenvalues. (For simplicity, assume no two eigenvectors have the same eigenvalue.) We want to **write a constrained optimization problem that identifies the eigenvector corresponding to the fifth-largest eigenvalue**. Explain how to modify the optimization problem at the beginning of this question so that the vector y it finds is the desired eigenvector. (You may change the objective function and/or add constraints, but they must be mathematical, and you cannot write things like “subject to y being the eigenvector corresponding to the fifth-largest eigenvalue.”)

Add the following constraints: $v_2^\top y = 0$, $v_3^\top y = 0$, $v_4^\top y = 0$.

Q5. [12 pts] Hierarchical Spectral Graph Multi-Clustering

In this problem, we shall consider the same graph as in the previous question, but we use multiple eigenvectors to perform 3-cluster clustering with the algorithm of Ng, Jordan, and Weiss (as opposed to the 2-cluster clustering we performed in the last question).

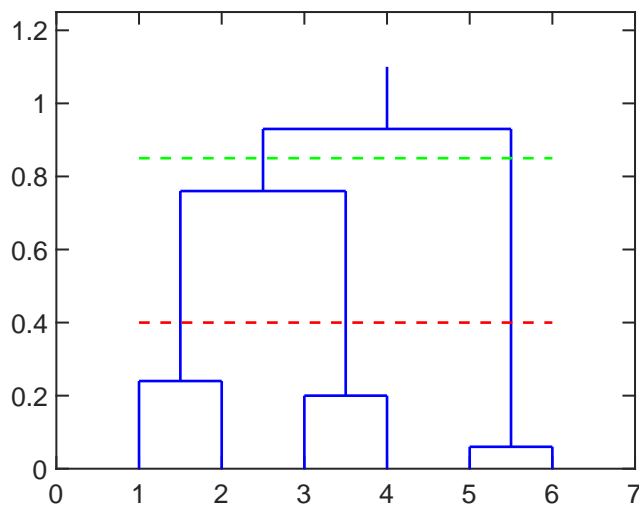
- (1) [4 pts] Based on the six eigenvectors of L given in the previous question, **write down the spectral vector** (as defined in the lecture notes) **for each vertex $1, \dots, 6$ in that order.**

The matrix of spectral vectors is

$$\begin{bmatrix} 0.41 & 0.36 & -0.56 \\ 0.41 & 0.34 & -0.32 \\ 0.41 & 0.25 & 0.43 \\ 0.41 & 0.19 & 0.62 \\ 0.41 & -0.55 & -0.06 \\ 0.41 & -0.6 & -0.1 \end{bmatrix}.$$

- (2) [8 pts] We shall now cluster the six raw, unnormalized spectral vectors obtained above using **hierarchical agglomerative clustering with the Euclidean distance metric and single linkage**. In contrast to what was discussed in class, we are not normalizing the six spectral vectors (because we don't want you to work that hard). Draw the **complete single linkage dendrogram** on paper. The six integer points on the x -axis, $1, \dots, 6$, should represent the vertices of the graph in the order of their indices. The y -axis should indicate the linkage distances, as is standard for dendrograms. **The numerical distance at which each fusion happens should be clearly marked on your figure.**

Points 5 and 6 merge first; the Euclidean distance between them is 0.06. Points 3 and 4 merge next; the Euclidean distance between them is 0.2. Points 1 and 2 merge next; the Euclidean distance between them is 0.24. The single linkage distance between $\{1, 2\}$ and $\{3, 4\}$ is 0.76. The single linkage distance between $\{1, 2\}$ and $\{5, 6\}$ is 0.93. That between $\{3, 4\}$ and $\{5, 6\}$ is 0.94. Therefore, the fourth merger is between $\{1, 2\}$ and $\{3, 4\}$. The clusters after the fourth merger are $\{1, 2, 3, 4\}$ and $\{5, 6\}$, matching what we obtained in the previous question. $\{1, 2, 3, 4\}$ and $\{5, 6\}$ merge at 0.93. The green dotted line indicates the 2-cluster clustering. The red dotted line indicates the 3-cluster clustering. (Note that students aren't asked to specify those lines.)

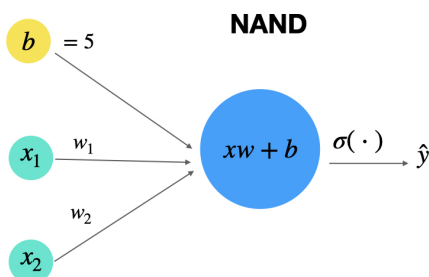


Q6. [10 pts] A Miscellany

- (1) [4 pts] Consider a single unit in a neural network that receives two binary inputs $x_1, x_2 \in \{0, 1\}^2$ and computes a linear combination followed by a threshold activation function, namely,

$$\sigma(z) = \begin{cases} 1, & z \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

The unit is illustrated below. We have chosen a bias term of $b = 5$. **Provide values for the two weights w_1 and w_2** that allow you to compute the NAND function (which is 0 if and only if both inputs are 1).



$w_1 = -3, w_2 = -3$ will work.

- (2) [6 pts] We are drawing sample points from a distribution with the probability density function (PDF) $f(x) = \frac{1}{2} e^{-|x-\mu|}$, but we do not know the mean $\mu \in \mathbb{R}$. We decide to estimate μ with maximum likelihood estimation (MLE). Unfortunately, we have only two sample points $X_1, X_2 \in \mathbb{R}$.

Derive the likelihood and the log-likelihood for this problem. Then **show that every value of μ between X_1 and X_2 is a maximum likelihood estimate**.

The likelihood is

$$\mathcal{L}(\mu; X_1, X_2) = \frac{1}{4} e^{-|X_1-\mu|} e^{-|X_2-\mu|},$$

and the log-likelihood is

$$\ell(\mu; X_1, X_2) = -|X_1 - \mu| - |X_2 - \mu| - \ln 4.$$

For any μ between X_1 and X_2 (inclusive), the log-likelihood is $-|X_1 - X_2| - \ln 4$. For any μ outside that range, it is lesser. (For example if μ is less than $\min\{X_1, X_2\}$, then the log-likelihood is $-|X_1 - X_2| - 2|\mu - \min\{X_1, X_2\}| - \ln 4$).

Q7. [16 pts] Dual Ridge Regression & Leave-One-Out Error

[This question has four independent parts. If you get stuck on one, try the others. Each part depends on the statements made in the previous parts, but not on your answer to the previous parts. Please show your work!]

Let X be an $n \times d$ design matrix representing n sample points with d features. (The last column of X may or may not be all 1's, representing a fictitious dimension; it won't affect this question.) Let $\mathbf{y} \in \mathbb{R}^n$ be a vector of labels. As usual, X_i denotes the i th sample point expressed as a column vector (X_i^\top is row i of X) and y_i denotes the i th scalar component of \mathbf{y} . Recall that **ridge regression** finds the weight vector \mathbf{w}^* minimizing the cost function

$$J(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2$$

where $\lambda > 0$ is the regularization hyperparameter. Because $\lambda > 0$, every regression problem we will consider here has exactly one unique minimizer. For X and \mathbf{y} , the unique minimizer of J is denoted by \mathbf{w}^* , giving a unique linear hypothesis $h(\mathbf{z}) = \mathbf{w}^* \cdot \mathbf{z}$.

- (1) [4 pts] Regression doesn't usually have zero training error; we would like to check the value $h(X_i) = \mathbf{w}^* \cdot X_i$ to see how close it is to y_i . **Recall the dual form of ridge regression and use it to show that $\mathbf{w}^* \cdot X_i = \mathbf{y}^\top (K + \lambda I)^{-1} K_i$** , where K is the kernel matrix and K_i is column i of K . Show your work. (*Note: we are not lifting the sample points to another feature space; we are just doing dual ridge regression with kernel matrix $K = XX^\top$.*)

In dual ridge regression, we set $\mathbf{w} = X^\top \mathbf{a}$ where $\mathbf{a} \in \mathbb{R}^n$ is a vector of dual weights, and the optimal dual solution is $\mathbf{a}^* = (K + \lambda I)^{-1} \mathbf{y}$, so $\mathbf{w}^* = X^\top (K + \lambda I)^{-1} \mathbf{y}$. Thus $\mathbf{w}^* \cdot X_i = \mathbf{y}^\top (K + \lambda I)^{-1} X X_i$. Column i of $K = X X^\top$ is $X X_i$, so $\mathbf{w}^* \cdot X_i = \mathbf{y}^\top (K + \lambda I)^{-1} K_i$.

The **Leave-One-Out (LOO)** error of a regression algorithm is the expected loss on a randomly chosen training point when you train on the other $n - 1$ points, leaving the chosen point out of training. Let X^i denote the $(n - 1) \times d$ design matrix obtained by removing the sample point X_i (the i th row of X) from X , and let $\mathbf{y}^i \in \mathbb{R}^{n-1}$ denote the vector obtained by removing y_i from \mathbf{y} . Let J^i be the cost function of ridge regression on X^i and \mathbf{y}^i , and let \mathbf{w}^i be the optimal weight vector that minimizes $J^i(\mathbf{w})$.

- (2) [4 pts] Suppose that after we perform ridge regression on X^i and \mathbf{y}^i , we discover that our linear hypothesis function just happens to fit the left-out sample point perfectly; that is, $\mathbf{w}^i \cdot X_i = y_i$. **Prove that $\mathbf{w}^* = \mathbf{w}^i$** . That is, removing the sample point X_i did not change the weights or the linear hypothesis. (*Hint: find the difference between $J(\mathbf{w})$ and $J^i(\mathbf{w})$ (for an arbitrary \mathbf{w}), then reason about the relationships between $J(\mathbf{w})$, $J^i(\mathbf{w})$, $J^i(\mathbf{w}^i)$, and $J(\mathbf{w}^i)$.)*

$$J(\mathbf{w}) = \sum_{j=1}^n (X_j \cdot \mathbf{w} - y_j)^2 + \lambda\|\mathbf{w}\|^2 \quad \text{and} \quad J^i(\mathbf{w}) = \sum_{j \neq i} (X_j \cdot \mathbf{w} - y_j)^2 + \lambda\|\mathbf{w}\|^2.$$

Hence $J(\mathbf{w}) - J^i(\mathbf{w}) = (X_i \cdot \mathbf{w} - y_i)^2$. Therefore, $J(\mathbf{w}) \geq J^i(\mathbf{w})$ for all \mathbf{w} . By assumption, $\mathbf{w}^i \cdot X_i = y_i$, so $J(\mathbf{w}^i) = J^i(\mathbf{w}^i)$. Recall that \mathbf{w}^i is the weight vector that minimizes J^i .

It follows that for every $\mathbf{w} \in \mathbb{R}^d$, $J(\mathbf{w}) \geq J^i(\mathbf{w}) \geq J^i(\mathbf{w}^i) = J(\mathbf{w}^i)$. Hence \mathbf{w}^i minimizes J . J has only one unique minimizer, which we call \mathbf{w}^* , so $\mathbf{w}^* = \mathbf{w}^i$.

Suppose we are not so lucky, and it turns out that $\mathbf{w}^i \cdot X_i \neq y_i$. Let $\mathbf{y}^{(i)} \in \mathbb{R}^n$ denote the vector obtained by taking \mathbf{y} and changing the i th component, replacing y_i with $\mathbf{w}^i \cdot X_i$. Let $\mathbf{w}^{(i)}$ be the optimal weight vector that minimizes the ridge regression cost function on the inputs X and $\mathbf{y}^{(i)}$. Our result from part (b) shows that $\mathbf{w}^{(i)} = \mathbf{w}^i$.

- (3) [4 pts] From part (a), **show that $\mathbf{w}^{(i)} \cdot X_i - \mathbf{w}^* \cdot X_i = (\mathbf{w}^{(i)} \cdot X_i - y_i) (K + \lambda I)_i^{-1} K_i$** , where $(K + \lambda I)_i^{-1}$ denotes row i of $(K + \lambda I)^{-1}$. (*Hint: The result from part (a) implies that $\mathbf{w}^{(i)} \cdot X_i = \mathbf{y}^{(i)\top} (K + \lambda I)^{-1} K_i$. What does $\mathbf{y}^{(i)} - \mathbf{y}$ look like?*)

$\mathbf{y}^{(i)} - \mathbf{y} = [0 \quad \dots \quad 0 \quad \mathbf{w}^i \cdot X_i - y_i \quad 0 \quad \dots \quad 0]^\top$, a vector of all zeros except in component i , so by part (a),

$$\mathbf{w}^{(i)} \cdot X_i - \mathbf{w}^* \cdot X_i = (\mathbf{y}^{(i)} - \mathbf{y})^\top (K + \lambda I)^{-1} K_i = (\mathbf{w}^i \cdot X_i - y_i) (K + \lambda I)_i^{-1} K_i = (\mathbf{w}^{(i)} \cdot X_i - y_i) (K + \lambda I)_i^{-1} K_i.$$

The Leave-One-Out error is defined to be

$$R_{\text{LOO}} = \frac{1}{n} \sum_{i=1}^n (\mathbf{w}^i \cdot X_i - y_i)^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{w}^{(i)} \cdot X_i - y_i)^2.$$

The LOO error is often an excellent estimator for the regression loss on unseen data. In general, the computation of LOO error can be very costly because it requires training the algorithm n times. But for dual ridge regression, remarkably, the LOO error can be computed by training the algorithm only once! Let's see how to compute the terms in the summation quickly.

(4) [4 pts] **Show that** $\mathbf{w}^{(i)} \cdot X_i - y_i = \frac{\mathbf{w}^* \cdot X_i - y_i}{1 - (K + \lambda I)_i^{-1} K_i}.$

From part (c), we have

$$\begin{aligned} (\mathbf{w}^{(i)} \cdot X_i - y_i) - (\mathbf{w}^* \cdot X_i - y_i) &= (\mathbf{w}^{(i)} \cdot X_i - y_i)(K + \lambda I)_i^{-1} K_i \\ (\mathbf{w}^{(i)} \cdot X_i - y_i)(1 - (K + \lambda I)_i^{-1} K_i) &= \mathbf{w}^* \cdot X_i - y_i \\ \mathbf{w}^{(i)} \cdot X_i - y_i &= \frac{\mathbf{w}^* \cdot X_i - y_i}{1 - (K + \lambda I)_i^{-1} K_i}. \end{aligned}$$

Postscript: We can add the kernel trick to this method if we want; it adds no difficulties, though for speed we usually want to use the kernel function to compute K and each $\mathbf{w}^* \cdot X_i$. The technique also requires us to compute the diagonal of $(K + \lambda I)^{-1} K$, which is probably best done by a Cholesky factorization of $(K + \lambda I)^{-1}$ and backsubstitution.