Name: ________________________________

UID: ________________________________

RIGHT NOW: Write your name and UID in the top right corner of every page!!! This is worth 10 points!

Test Information

This test has two parts: true/false and short answer.

There is no partial credit for the true/false questions, so don’t bother explaining your answers.

There are TWO short answer questions, of which you should complete ONE. If you complete more than one, we will grade only grade the first. If you write in one of the questions and then change your mind and decide not to do it, cross out all your writing completely so there is no confusion.

You are only allowed to use the one page of notes that you brought, and definitely not allowed to use any communicating device (i.e., no internet browsing, chatting, emails, texting, friends, telepathy, etc.). You must sign below to indicate that you understand this and as confirmation that you have not and will not break this rule. Not signing will result in a ZERO exam score.

Grading (do not write in this section)

<table>
<thead>
<tr>
<th></th>
<th>/ 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name+UID</td>
<td>_____</td>
</tr>
<tr>
<td>1: True/False</td>
<td>_____</td>
</tr>
<tr>
<td>2: DTs++</td>
<td>_____</td>
</tr>
<tr>
<td>3: Multiclass</td>
<td>_____</td>
</tr>
<tr>
<td>Total</td>
<td>_____</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
</tbody>
</table>
1 True/False (30%)

Please write (T or F) in the parentheses to the left. No partial credit. Each is worth 3 points. Not necessarily balanced.

( ) Gradient descent will always find the minimum of a convex function.

( ) Inductive bias is what a learning algorithm will do when the data is insufficient to make a decision.

( ) As a perceptron makes multiple passes over the training data, we expect it to initially overfit and eventually underfit, so early stopping can find a good break-even point.

( ) As the depth of a decision tree grows, training error will go down and test error will probably go down for a while and then go up.

( ) The K-means algorithm is guaranteed to converge.

( ) Even on linearly inseparable data, the perceptron might still converge.

( ) Because OVA only requires training $K$ classifiers and AVA requires $\binom{K}{2}$, OVA is always faster to train.

( ) In a linear model that optimizes $\text{loss} + \lambda R(w)$, where $R$ is a $p$-norm regularizer, then small $\lambda$ leads to overfitting and large $\lambda$ leads to underfitting.

( ) In KNN, having a large $K$ is likely to lead to underfitting.

( ) The inductive bias of decision trees make them good on linearly separable data.
2 Decision Trees++ (60%)

One disadvantage to decision trees is that they are only “good at” making axis-aligned decisions. This is because they can only ask questions about one feature at a time. For instance, they would have a very hard time with the data shown (separating crosses from circles). Note also that linear models would have a hard time with this data, since it represents an XOR-like problem.

Suppose that instead of asking questions about a single feature at each node in a tree, you could ask questions about any linear combination of features.

(a) [15%] Specify a depth two decision tree (i.e., one that asks at most two questions) that achieves perfect classification on this data set (at the leaves, return either x or o).

(b) [15%] Draw its decision boundary on top of the data and label the regions with their assigned label.
(c) [20%] Write (in psuedo-code) a training algorithm for this decision tree; you may assume that you have access to a subroutine for training a linear classifier.

(d) [10%] If you ran your training algorithm from (c), would it find the decision boundary you came up with in (a) and (b)? If so, why; if not, why not?
3 Multiclass Perceptron (60%)

Suppose that instead of predicting +1 or −1, we need to predict a label from a set of \( L \) classes, 1, 2, \ldots, \( L \). Below is an algorithm for training a multiclass extension of a perceptron. Instead of having one weight vector \( \mathbf{w} \in \mathbb{R}^D \), we keep around \( L \)-many weight vectors, called \( \mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(L)} \). Intuitively, the weight vector \( \mathbf{w}^{(l)} \) corresponds to recognizing when \( k \) is the correct label.

The decision boundary and weight vectors for an example where \( L = 3 \) is shown below. The region in which \( \mathbf{w}^{(1)} \cdot \mathbf{x} \) is greatest is the \( y = 1 \) region, and similarly for \( l = 2, 3 \).

![Decision boundary and weight vectors](image)

1. For each training example \((\mathbf{x}, y)\), where \( y \in \{1, \ldots, L\} \)
   
   (a) Compute the activation for each weight vector:
   
   For \( l \in \{1, \ldots, L\} \):
   
   i. Compute \( a_l = \mathbf{w}^{(l)} \cdot \mathbf{x} \)

   (b) Predict the label that has the highest activation:

   Compute \( \hat{y} = \text{arg max}_l a_l \)

   (c) Check for error and update only if error:

   If \( \hat{y} \neq y \):

   i. Update weights for correct label toward \( \mathbf{x} \):

   \[
   \mathbf{w}^{(y)} \leftarrow \mathbf{w}^{(y)} + \mathbf{x}
   \]

   ii. Update weights for predicted label away from \( \mathbf{x} \):

   \[
   \mathbf{w}^{(\hat{y})} \leftarrow \mathbf{w}^{(\hat{y})} - \mathbf{x}
   \]

2. Return all weights \( \mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(L)} \)

(a) [15%] Suppose that you this algorithm observed an example \((\mathbf{x}, y)\), made and error, and then got the same example again. Show that the second time around the algorithm has a better chance of being correct.
(b) [15\%] Write down a formal (mathematical) definition of what it means to be linearly separable in this setting, such that the data from the above figure is deemed linearly separable.

(c) [15\%] If the data is linearly separable according to your definition in (b), and we turn it into $L$-many binary classification problems in an one-versus-rest (OVA), is each of these guaranteed to be linearly separable? If so, say why. If not, give a counter example.

(d) [15\%] For an appropriate definition of multiclass margin ($\gamma$), one can show that this algorithm converges after $L/\gamma^2$ updates, analogous to the binary perceptron. Suppose that, instead of training a multiclass perceptron, we trained $\binom{L}{2}$ binary perceptrons in an all-pairs (AVA) style. Assume that a margin of $\gamma$ on the multiclass problem means that we get at least the same margin on each of the binary problems. Give an upper bound on the total number of updates made in the binary case (summed across all binary problems).