• This is a closed book exam. Everything you need in order to solve the problems is supplied in the body of this exam.

• This exam booklet contains four problems. You need to solve all problems to get 100%.

• Please check that the exam booklet contains 14 pages, with the appendix at the end.

• The exam ends at 1:45 PM. You have 75 minutes to earn a total of 100 points.

• Answer each question in the space provided. If you need more room, write on the reverse side of the paper and indicate that you have done so.

• A list of potentially useful functions has been provided in the appendix at the end.

• Besides having the correct answer, being concise and clear is very important. For full credit, you must show your work and explain your answers.

Good Luck!

Name (NetID): (1 Point)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Trees</td>
<td>20</td>
</tr>
<tr>
<td>PAC Learning</td>
<td>29</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>25</td>
</tr>
<tr>
<td>Short Questions</td>
<td>25</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
Decision Trees [20 points]

You work in a weather forecasting company and your job as a machine learning expert is to design a decision tree which would predict whether it is going to rain today (‘WillRain?’ = 1) or not (‘WillRain?’ = 0). You are given a dataset D with the following attributes: \( Is\text{Humid} \in \{0,1\} \), \( Is\text{Cloudy} \in \{0,1\} \), \( Rained\text{Yesterday} \in \{0,1\} \) and \( Temp>20 \in \{0,1\} \).

<table>
<thead>
<tr>
<th>IsHumid</th>
<th>IsCloudy</th>
<th>RainedYesterday</th>
<th>Temp&gt;20</th>
<th>WillRain?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

To simplify your computations please use: \( \log_2(3) \approx \frac{3}{2} \).

(a) (4 points) What is the entropy of the label ‘WillRain?’?

\[
\text{Entropy}(\text{‘WillRain?’}) = -\frac{2}{8}\log_2\left(\frac{2}{8}\right) - \frac{6}{8}\log_2\left(\frac{6}{8}\right) = \frac{7}{8} = 0.875
\]

(b) (4 points) What should the proportion of the examples labeled ‘WillRain?’=1 be, in order to get the maximum entropy value for the label?

Half of the examples should have label 1 and the other half have the label 0.
(c) **(4 points)** Compute the Gain(D, IsCloudy).

\[
\text{Entropy}(D, \text{IsCloudy}=1) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)
\]
⇒ 1

\[
\text{Entropy}(D, \text{IsCloudy}=0) = -\frac{0}{4}\log_2\left(\frac{0}{4}\right) - \frac{4}{4}\log_2\left(\frac{4}{4}\right)
\]
⇒ 0

\[
\text{Gain}(D, \text{IsCloudy}) = 0.875 - \frac{3}{8}\times1 - \frac{1}{8}\times0
\]
⇒ 0.375

(d) **(4 points)** You are given that:

- Gain(D, IsHumid) = 0.25,
- Gain(D, RainedYesterday) = 0.11,
- Gain(D, Temp>20) = 0
- Gain(D, IsCloudy) is as computed in part c.

i. Which node should be the root node?

IsCloudy should be the root node since it gives the highest information gain.

ii. Without any additional computation, draw a decision tree that is consistent with the given dataset and uses the root chosen in (i).

```python
if(IsCloudy):
    if(IsHumid):
        1
    else:
        0
else:
    0
```
(e) (4 points) Express the function ‘WillRain?’ as a simple Boolean function over the features defining the data set $D$. That is, define a Boolean function that returns true if and only if ‘WillRain?’ = 1.

$\text{IsCloudy} \land \text{IsHumid}$
PAC Learning [29 points]

We define a set of functions

\[ T = \{ f(x) = 1[x > a] : a \in \mathbb{R} \}, \]

where \( 1[x > a] \) is the indicator function returning 1 if \( x > a \) and returning 0 otherwise.

For input domain \( \mathcal{X} = \mathbb{R} \), and a fixed positive number \( k \), consider a concept class \( DT_k \) consisting of all decision trees of depth at most \( k \) where the function at each non-leaf node is an element of \( T \). Note that if the tree has only one decision node (the root) and two leaves, then \( k = 1 \).

(a) (4 points) We want to learn a function in \( DT_k \). Define

i. The Instance Space \( X \)
\( X = \mathbb{R} \)

ii. The Label Space \( Y \)
\( Y = \{0, 1\} \)

iii. Give an example of \( f \in DT_2 \).

There are many possible answers for this. Here is one, where we assume that the right branch is taken if the node is satisfied and the left node is taken otherwise.

```
  1[x > 0]
  /   \
/     \          \[x > -1]\ [x > 1]
1  0  0  1
```

iv. Give 3 examples that are consistent with your function \( f \) and one that is not consistent with it.
For the previous tree, here are three consistent examples:

\[ x = 10, y = 1 \]
\[ x = \frac{1}{2}, y = 0 \]
\[ x = -50000, y = 1 \]

And here is an inconsistent example: \((x = 10, y = 0)\), since the label of \(x = 10\) should be 1.
(b) (7 points) Determine the VC dimension of $DT_k$, and prove that your answer is correct.

First, note that the root node of the tree partitions the input space into two intervals. Each child node then recursively divides the corresponding interval into two more intervals. Hence, a full decision tree of depth $k$ divides the input space into at most $2^k$ intervals, each of which can be assigned a label of 0 or 1. We will show that the VC dimension of $DT_k$ is $2^k$.

Proof that $\text{VCDim}(DT_k) \geq 2^k$: given $2^k$ points, construct a decision tree such that each point lies in a separate interval. Then, one can assign labels to the leaves of the tree corresponding to any possible labeling of the points.

Proof that $\text{VCDim}(DT_k) < 2^k + 1$: recall that a function from $DT_k$ can divide the input space into at most $2^k$ intervals. Consequentially, the pigeonhole principle tells us that, given $2^k + 1$ points, at least one interval must contain two points no matter which function we are considering. This implies that no function in $DT_k$ can shatter $2^k + 1$ points, since every function in the class will contain an interval with more than one point and thus cannot assign different labels to those points.
(c) (5 points) Now consider a concept class $DT_\infty$ consisting of all decision trees of unbounded depth where the function at each node is an element of $T$. Give the VC dimension of $DT_\infty$, and prove that your answer is correct.
We will show that $VCDim(DT_\infty) = \infty$.

Proof: For all positive integers $m$, given $m$ points, we can construct a tree of height $\lceil \log_2(m) \rceil$ that places each point in a separate interval, thus allowing the set of points to be shattered. Since there is no limit to how deep the tree can be constructed, we can therefore shatter any number of points.
(d) (7 points) Assume that you are given a set \( S \) of \( m \) examples that are consistent with a concept in \( DT_k \). Give an efficient learning algorithm that produces a hypothesis \( h \) that is consistent with \( S \).

Note: The hypothesis you learn, \( h \), does not need to be in \( DT_k \). You can represent it any way you want.

Let \( S = \{(x_1, y_1), \ldots, (x_m, y_m)\} \) be the set of the \( m \) examples after we sorted it based on the values of the \( x \)s. That is, \( x_1 \leq x_2 \leq \ldots x_m \).

We create a list of interval boundaries in the following way: we place a boundary between two points with a different label. We represent the set of intervals determined by the set \( S \) as a set of pairs \((b_i, y_i)\), where \( b_i \) is the lower end of the interval and \( y_i \) is the label for that interval. Formally:

- Initialize list of interval boundaries \( I = [(-\infty, y_1)] \)
- for \( i = 1 \ldots m - 1: \)
  - if \( y_i \neq y_{i+1} \):
    - add \((\frac{x_i + x_{i+1}}{2}, y_{i+1})\) to \( I \)

The set \( I \) is the hypothesis learned by our algorithm.

To classify a \( x \), find the boundary with the largest \( b_i \) such that \( b_i \leq x \). and assign \( x \) the label \( y_i \). Note that this way it is clear that the hypothesis \( I \) is consistent with the training set \( S \).

Since the set \( S \) of examples given is known to be consistent with a function in \( DT_k \), at most \( 2^k \) intervals will be constructed. Building the hypothesis can thus be done in \( O(m \log m) \) (assuming an efficient sorting algorithm is used). Classifying new points can be accomplished in \( O(k) \) time if binary search is used.

(e) (6 points) Is the concept class \( DT_k \) PAC learnable? Explain your answer.

Yes. Any hypothesis class with a finite VC dimension is PAC Learnable.
Neural Networks [25 points]

Consider the following set $S$ of examples over the feature space $X = \{X_1, X_2\}$. These examples were labeled based on the XNOR (NOT XOR) function.

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$y^*$ (Label)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) (4 points) The set of 4 examples given above is not linearly separable in the $X = \{X_1, X_2\}$ space. Explain this statement in one sentence.

It means that there exists no triple of real numbers $(w_1, w_2, b)$ such that for all labeled examples $(X_1, X_2, y^*)$ given, we have that: $y^*(w_1X_1 + w_2X_2 + b) > 0$.

(b) (6 points) Propose a new set of features $Z = \{Z_1, \ldots, Z_k\}$ such that in the $Z$ space, this set of examples is linearly separable.

i. Define each $Z_i$ as a function of the $X_i$s.

There are many such mappings, and any reasonable mapping that results in linearly separable data is acceptable. One such mapping is:

$Z_1 = \neg X_1 \land \neg X_2, \quad Z_2 = X_1 \land X_2$

ii. Write down the set of 4 examples given above in the new $Z$ space.

Following the same order of examples as in the table above, we get:

<table>
<thead>
<tr>
<th>$Z_1$</th>
<th>$Z_2$</th>
<th>$y^*$ (Label)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
iii. Show that the data set is linearly separable. (Show, don’t just say that it is separable.)

Given the definition of linear separability in (a), we need to provide a triple \((w_1, w_2, b)\) that linearly separates the points in the \(Z\) space. We note that in the \(Z\) space the target function is a disjunction \(Z_1 \lor Z_2\) and therefore one such triple is \((w_1 = 1, w_2 = 1, b = -0.5)\).
(c) (5 points) Now consider running the set $S$ of examples presented above in the space $X$ through a neural network with a single hidden layer, as shown in the figure below.

Note that numbers on the edges correspond to weights and the arrows into the units indicate the bias term. Recall that the output of a node (denoted by the terms inside the nodes in the graph e.g. $a_1, a_2, y$) in the neural network is given by $f(w^Tx + b)$, where $x$ is the input to the unit, $w$ are the weights on the input, $b$ is the bias in the unit, and $f$ is the activation function.

For the sake of simplicity, assume that the function $sgn(x)$ ($sgn(x) = 1$ if $x \geq 0$, 0 otherwise) is used as the activation function at all the nodes of the network.

**Which of the following sets of weights** guarantees that the neural network above is consistent with all the examples in $S$? (That is, the 0-1 loss is 0).

The correct set of weights is \( (3) \)

\[ \{ \text{option (1)} \mid \text{option (2)} \mid \text{option (3)} \} \]

**Options:**

<table>
<thead>
<tr>
<th>Options</th>
<th>$w_{11}$</th>
<th>$w_{21}$</th>
<th>$b_1$</th>
<th>$w_{12}$</th>
<th>$w_{22}$</th>
<th>$b_2$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-0.5</td>
<td>0</td>
<td>1</td>
<td>-0.5</td>
<td>-1</td>
<td>-1</td>
<td>0.9</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>-1</td>
<td>-1</td>
<td>2.5</td>
<td>0</td>
<td>-1</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>-0.5</td>
<td>-1</td>
<td>-1</td>
<td>1.5</td>
<td>-1</td>
<td>-1</td>
<td>1.5</td>
</tr>
</tbody>
</table>

To show that the neural network is consistent with that dataset, one needs to run each example through the neural network, and check that the output of the neural network matches with the true label. We show this for option (3) below.
<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$y$</th>
<th>$y^*$ (Label)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
(d) **(10 points)** We now want to use the data set $S$ to *learn* the neural network depicted earlier.

We will use the *sigmoid* function, $\text{sigmoid}(x) = (1 + e^{x})^{-1}$, as the activation function in the hidden layer, and *no activation function in the output layer* (i.e. it’s just a linear unit). As the loss function we will use the Hinge Loss:

$$\text{Hinge loss}(w, x, b, y^*) = \begin{cases} 1 - y^*(w^T x + b), & \text{if } y^*(w^T x + b) > 1 \\ 0, & \text{otherwise} \end{cases}$$

**Write down** the BackPropagation update rules for the weights in the output layer ($\Delta v_i$), and the hidden layer ($\Delta w_{ij}$). By definition, we have that:

$$v_{i}^{t+1} = v_{i}^{t} + \Delta v_{i}$$

and

$$w_{ij}^{t+1} = w_{ij}^{t} + \Delta w_{ij}$$

where the updates are computed by:

$$\Delta v_{i} = -\eta \nabla v_{i}$$

and

$$\Delta w_{ij} = -\eta \nabla w_{ij}$$

We now need to compute the derivatives. First, assuming no activation function on the output layer, we get that:

$$\nabla v_{i} = \begin{cases} -y^* a_{i}, & \text{if } y^* y > 1 \\ 0, & \text{otherwise} \end{cases}$$

and:

$$\delta_{i} = \begin{cases} -y^* v_{i}, & \text{if } y^* y > 1 \\ 0, & \text{otherwise} \end{cases}$$

And, assuming the sigmoid function as the activation function in the hidden layer, we get:

$$\nabla w_{ij} = \delta_{j} a_{j} (1 - a_{j}) x_{i}$$
Short Questions [25 points]

(a) (10 points) In this part of the problem we consider Adaboost. Let \( D_t \) be the probability distribution in the \( t \)th round of Adaboost, \( h_t \) be the weak learning hypothesis learned in the \( t \)th round, and \( \epsilon_t \) its error.

i. Denote by \( D_t(i) \) the weight of the \( i \)th example under the distribution \( D_t \). Use it to write an expression for the error \( \epsilon_t \) of the AdaBoost weak learner in the \( t \)th round.

Let \( S \) be the set of all examples. We denote by \([a]\) the characteristic function that returns 1 if \( a \) is true and 0 otherwise. Then, the error is defined as the total weight, under \( D_t \), of the examples \( h_t \) misclassifies:

\[
\epsilon_t = Error_{D_t}(h_t) = \sum_{i \in S} D_t(i)[h_t(i)\neg = y(i)]
\]

ii. Consider the following four statements with respect to the hypothesis at time \( t \), \( h_t \). Circle the one that is true, and provide a short explanation.

A. \( \forall t, Error_{D_t}(h_t) = Error_{D_{t+1}}(h_t) \)
B. \( \forall t, Error_{D_t}(h_t) > Error_{D_{t+1}}(h_t) \)
C. \( \forall t, Error_{D_t}(h_t) < Error_{D_{t+1}}(h_t) \)
D. The relation between \( Error_{D_t}(h_t) \) and \( Error_{D_{t+1}}(h_t) \) cannot be determined in general.

Explanation: The right option is C. Consider the set of examples that are misclassified by \( h_t \). On the left hand side we have an expression that gives the total weight of these examples under \( D_t \). On the right hand side we have an expression that gives the total weight of the same set of examples under \( D_{t+1} \). However, Adaboost reweighs each example \( e \) that is misclassified by \( h_t \) so that \( D_{t+1}(e) > D_t(e) \), resulting in \( C \) being correct.

Note also that we know that the value of the left hand side is less than 1/2, by the weak learning assumption, and it can be shown (but not needed as part of the explanation) that the value of the right hand side is exactly 1/2.
(b) (10 points) We consider Boolean functions in the class $L_{10,20,100}$. This is the class of 10 out of 20 out of 100, defined over $\{x_1, x_2, \ldots x_{100}\}$.

Recall that a function in the class $L_{10,20,100}$ is defined by a set of 20 relevant variables. An example $x \in \{0, 1\}^{100}$ is positive if and only if at least 10 out these 20 are on.

In the following discussion, for the sake of simplicity, whenever we consider a member in $L_{10,20,100}$, we will consider the function $f$ in which the first 20 coordinates are the relevant coordinates.

i. Show that the perceptron algorithm can be used to learn functions in the class $L_{10,20,100}$. In order to do so,

A. Show a linear threshold function $h$ that behaves just like $f \in L_{10,20,100}$ on $\{0, 1\}^{100}$.

$f$ is determined by a bias terms $b = -10$, and a weight vectors $w \in \mathbb{R}^{100}$ so that $w_i = 1$, for $i = 1, 2, \ldots 20$, and $w_i = 0$ otherwise.

It is easy to see that $w \cdot x + b > 0$ iff $f(x) = 1$

B. Write $h$ as a weight vector that goes through the origin and has size (as measured by the $L_2$ norm) equal to 1.

To represent $h$ as a weight vector that goes through the origin we represent the example now as $x' = (x, 1) \in \{0, 1\}^{101}$ and the weight vector as $w' = (w, b) \in \mathbb{R}^{101}$. We note that $w \cdot x + b = w' \cdot x'$. To make sure that $h$ has $L_2$ norm that is equal to 1, we normalize it by dividing $(w, b)$ by $\| (w, b) \|_2 = \sqrt{20 + 100}$. 


ii. Let $R$ be the set of 20 variables defining the target function. We consider the following two data sets, both of which have examples with 50 on bits.

$D_1$: In all the negative examples exactly 9 of the variables in $R$ are on; in all the positive examples exactly 11 of the variables in $R$ are on.

$D_2$: In all the negative examples exactly 5 of the variables in $R$ are on; in all the positive examples exactly 15 of the variables in $R$ are on.

Consider running perceptron on $D_1$ and on $D_2$. On which of these data sets do you expect Perceptron to make less mistakes?

Perceptron will make less mistakes on the data set

$\{D_1 \mid D_2\}$

Perceptron will make less mistakes on $D_2$ since the margin of this data set is larger than that of $D_1$.

iii. Define the margin of a data set $D$ with respect to weight vector $w$.

$$\gamma = \min_{||w||_2=1} |yw \cdot x|$$

Explain your answer to (ii) using the notion of the margin.

Formally we can appeal to Novikoff’s bound, assuming that, w.l.o.g., $R$ will be the same, since the mistake bound of perceptron is lower bounded by $R^2/\gamma^2$. 

17
(c) (5 points) Let $f$ be a concept that is defined on examples drawn from a distribution $D$. The “true” error of the hypothesis $h$ is defined as

$$Error_D(h) = Pr_{x \in D} (h(x) \neq f(x)).$$

In the class, we saw that the true error of a classifier is bounded above by two terms that relate to the training data and the hypothesis space. That is

$$Error_D(h) < A + B$$

What are $A$ and $B$? (If you do not remember the exact functional forms of these terms, it is sufficient to briefly describe what they mean.)

$A$ is the training error of $h$.

$B$ is a term that bounds how much will the true error of $h$ deviate from the observed (training) error of $h$. This term scales proportionally with the VC dimension of the hypothesis space $H$, and is inversely proportional to the number of training examples.
Appendix

(a) $\text{Entropy}(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$

(b) $\text{Gain}(S, a) = \text{Entropy}(S) - \sum_{v \in \text{values}(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$

(c) $\text{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$

(d) $\text{sigmoid}(x) = \frac{1}{1 + \exp^{-x}}$

(e) $\frac{\partial}{\partial x} \text{sigmoid}(x) = \text{sigmoid}(x) \left(1 - \text{sigmoid}(x)\right)$

(f) $\text{ReLU}(x) = \max(0, x)$

(g) $\frac{\partial}{\partial x} \text{ReLU}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$

(h) $\text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

(i) $\frac{\partial}{\partial x} \text{tanh}(x) = 1 - \text{tanh}^2(x)$

(j) $\text{Zero-One loss}(y, y^*) = \begin{cases} 1, & \text{if } y \neq y^* \\ 0, & \text{if } y = y^* \end{cases}$

(k) $\text{Hinge loss}(w, x, b, y^*) = \begin{cases} 1 - y^*(w^T x + b), & \text{if } y^*(w^T x + b) > 1 \\ 0, & \text{otherwise} \end{cases}$

(l) $\frac{\partial}{\partial w} \text{Hinge loss}(w, x, b, y^*) = \begin{cases} -y^*(x), & \text{if } y^*(w^T x + b) > 1 \\ 0, & \text{otherwise} \end{cases}$

(m) $\text{Squared loss}(w, x, y^*) = \frac{1}{2}(w^T x - y^*)^2$

(n) $\frac{\partial}{\partial w} \text{Squared loss}(w, x, y^*) = x(w^T x - y^*)$