CS446: Machine Learning **Fall 2012**

Mid-term Exam Solutions

October 30^{th}, 2012

- This is a closed book exam. Everything you need in order to solve the problems is supplied in the body of this exam. Note that there is an appendix with possibly useful formulae and computational shortcuts at the end.
- This exam booklet contains **five** problems, out of which you are expected to answer **four** problems of your choice.
- The exam ends at 10:45 AM. You have 75 minutes to earn a total of 100 points. You can earn 25 additional (bonus) points if you successfully attempt all five problems.
- If you choose to attempt all five problems, the four problems with the highest points will be considered for your mid-term score and the lowest will be considered as bonus.
- 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.
- **Besides having the correct answer, being concise and clear is very important. For full credit, you must show your work and explain your answers.**

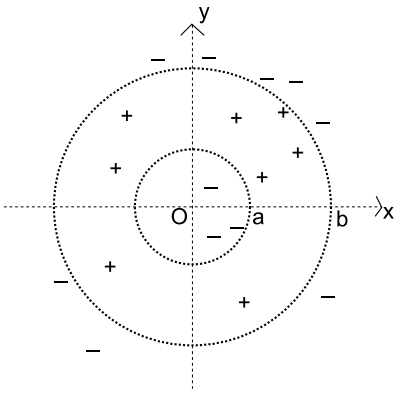
Good Luck!

Name (NetID): (1 Point)

Short Questions		/24
Decision Trees		/25
Online Learning		/25
Perceptrons		/25
Kernels		/25
Total		/100
Bonus		/25

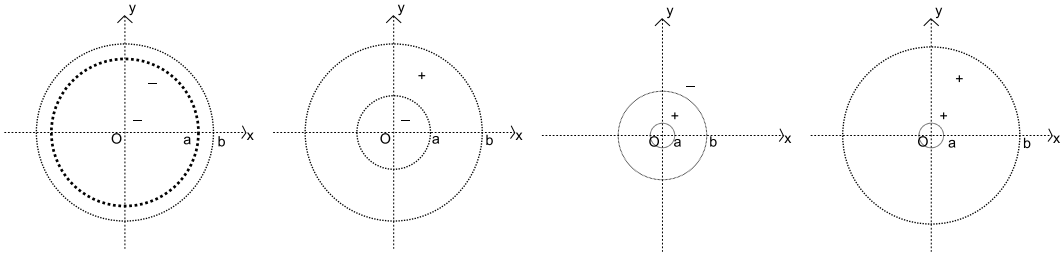
Short Questions [24 points]

(a) [6 points] Consider a concept space **H** of two nested circles centered on the origin (see figure below). Formally, a concept $h \in \mathbf{H}$ is defined by 2 non-negative real parameters $a, b \in \mathbb{R}^+$ such that $a < b$. An example $(x, y) \in \mathbb{R}^2$ is labeled +1 if and only if $a^2 < x^2 + y^2 < b^2$, i.e. (x, y) is within the band of the two nested circles of radius a and b respectively.



State the VC-dimension of **H**. Prove that your answer is correct.

There exists a set of 2 points in $\mathbb{R}^2$ that can be shattered by **H**.



So VC(**H**) ≥ 2.

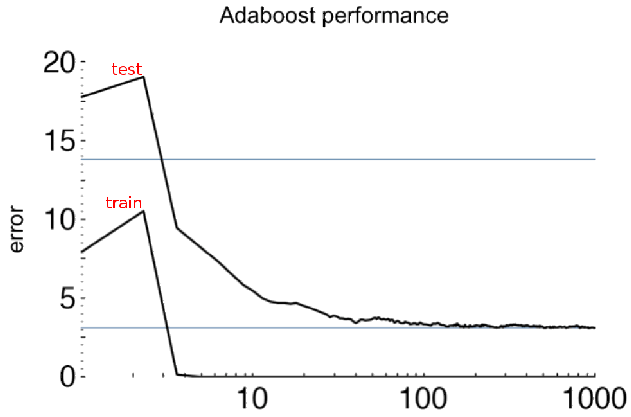
Prove that no 3 points in $\mathbb{R}^2$ may be shattered by **H**.

- Let $R_2(p) = (p.x^2 + p.y^2)$. Pick and order p_1, p_2, p_3 such that $R_2(p_1) \leq R_2(p_2) \leq R_2(p_3)$ but for any $i \neq j$, $p_i \neq p_j$.
- If for any two points $i \neq j$, $R_2(p_i) = R_2(p_j)$, cannot shatter if we pick different labels for p_i and p_j.
- If $R_2(p_1) < R_2(p_2) < R_2(p_3)$, then $p_1 = +, p_2 = -, p_3 = +$ cannot be shattered.

This covers all cases so no 3 points in $\mathbb{R}^2$ may be shattered by **H**. So VC(**H**) < 3.

Since there exists a set of cardinality 2 that can be shattered and no set of cardinality 3 can be, VC-dimension of **H** is 2.

(b) [6 points] The following graph illustrating the performance of Adaboost appeared in Schapire et al ICML 1997:



These results were obtained by boosting Quinlan's C4.5 decision tree learning algorithm on a UCI data set, varying the number of boosting rounds and recording the error of Adaboost's hypothesis on both the training set and a separate testing set.

1. [2 points] Which curve on the graph corresponds to training error and which to testing error? Put the labels "train" and "test" on the graph as your response.
2. [2 points] What did learning theorists find unusual or interesting about these results?
   The testing error continues to decrease even after the training error is 0.
3. [2 points] How can this interesting phenomenon be explained? Use evidence from the Adaboost learning algorithm and give a concise answer.
   - Adjustments made by Adaboost during each round of boosting are not a function of final, combined hypothesis' error.
   - They are a function of the weak learner's error, which continues to be non-zero after the training error of the combined hypothesis reaches zero.
   - Thus, the distribution kept by the algorithm continues to be modified, and useful features continue to be added to the combined hypothesis.

(c) [6 points] Dr. Robert Moose has shown you his latest invention: *the Moose classifier*. Each classifier divides the instance space X into positive and negative examples. Dr. Moose also has an efficient algorithm which, for any finite set S of labeled examples given as input, will return a classifier that correctly labels at least $\frac{2}{3}$ of the examples in S.

Does this imply that the class of Moose classifier is PAC-learnable? Justify your answer.

Yes.

Dr. Moose has a weak PAC learning algorithm since he can do better than a chance on any sample presented to him; therefore, this algorithm can be boosted to a strong PAC learning algorithm.

(d) [6 points] Let P be a specific probability distribution over the instance space X (P could be the uniform distribution over X, a normal distribution with known parameters, etc.). Define a concept class C to be in *PPAC-learnable* if and only if there is an algorithm that, when given a collection of labeled examples sampled randomly and independently according to the distribution P, with probability at least $(1 - \delta)$ it learns a hypothesis that makes at most ϵ error on data sampled according to P.

Denote by PAC the set of all *concept classes* that are PAC-learnable and by PPAC all *concept classes* that are PPAC-learnable. *Determine* which of the following four possibilities is true and briefly *justify* your answer.

A. PPAC $\subseteq$ PAC but not the opposite.

B. PAC $\subseteq$ PPAC but not the opposite.

C. PPAC = PAC.

D. None of the above.

B. holds.

PAC set requires each concept class to be learnable for all distributions, while PPAC set requires each concept class to be learnable only for a specific distribution P. Comparatively, PAC has a stronger requirement than PPAC, and thus the set of concept classes is a subset of PPAC.

Decision Trees [25 points]

You are given a collection of data points S describing the conditions and yield for 8 agricultural farms. Each farm has three attributes, and each has either a high yield or a low yield. The label is **Yield**. You may refer to the appendix at the end for some useful formulae and computational shortcuts.

#	IsIrrigated	IsFertilized	UsesPesticide	Yield
1	Not Enough	Yes	No	High
2	Too Much	Yes	No	High
3	None	Yes	No	Low
4	Not Enough	Yes	No	High
5	None	No	Yes	Low
6	Too Much	No	Yes	Low
7	Not Enough	No	No	Low
8	Not Enough	Yes	No	High

(a) [5 points] What is the entropy of the data (i.e. of label Yield)? What is the entropy of the attribute (not the split-based information gain) IsIrrigated? The entropy of the data is

$$-\frac{1}{2}\log(\frac{1}{2}) - -\frac{1}{2}\log(\frac{1}{2}) = 1.$$

The entropy of the attribute IsIrrigated is

$$-\frac{1}{4}\log\frac{1}{4} - -\frac{1}{4}\log\frac{1}{4} - -\frac{1}{2}\log\frac{1}{2} = \frac{3}{2}.$$

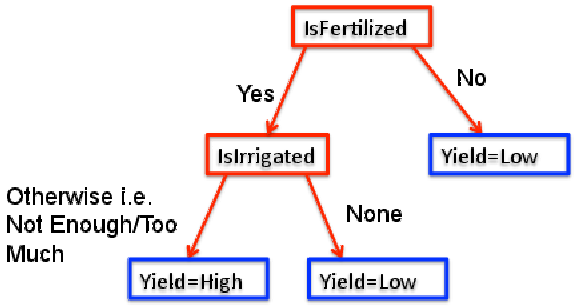
(b) [5 points] Compute $Gain(S,$ UsesPesticide$)$. (If needed, see the back page for formulas.)

$$Gain(S, UsesPesticide) = Entropy(S) - Entropy(Split(S, UsesPesticide))$$

$$= 1 - \frac{1}{4}\cdot 0 - \frac{3}{4}\left(-\frac{1}{3}\log\frac{1}{3} - \frac{2}{3}\log\frac{2}{3}\right) = \frac{3}{8} = 0.375$$

(c) [10 points] Suppose, in addition to your calculation in part (b), we tell you that $Gain(S,$ IsIrrigated$) = 0.3$ and $Gain(S,$ IsFertilized$) = 0.5$.
Use the information you have to choose the appropriate root node. Using this root node, provide a minimal (i.e. with the smallest number of nodes) decision tree that is **consistent** with the given data. Please note that beyond the root node, you do not need to use any specific algorithm; just make sure your tree is consistent with the data. Draw the tree below.

Using part (b), clearly the attribute IsFertilized has the highest information gain. Thus we choose that as the root node and construct a consistent tree as given below. Notably the above tree does not use the attribute UsePesticide. Some



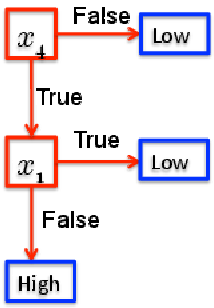
students got the root node correct but used the 'UsePesticide' attribute in the tree (which really serves no purpose) – they lost 5 points as it contradicts with the minimality requirement of the question. Also, some other students chose a different node due to mis-calculation in part (c). However, if their selection of root node is consistent with their result in part (c) and the tree is consistent with the data, they didn't lose points in this part.

(d) [5 points] Let us define five boolean features $(x_1, \ldots, x_5)$ as follows.

- $x_1 \equiv$ (IsIrrigated == None) is a boolean feature which is 1 if the feature IsIrrigated is 'None' and otherwise 0. Similarly,
- $x_2 \equiv$ (IsIrrigated == Not Enough),
- $x_3 \equiv$ (IsIrrigated == Too Much),
- $x_4 \equiv$ (IsFertilized == Yes), and
- $x_5 \equiv$ (UsesPesticide == Yes).

Express the decision tree you obtained in part (c) above as a linear threshold function over $x_1, \ldots, x_5$.

We can use the following 1-DL to express our decision tree.



We convert it into a linear threshold function similar to part (e) of problem 3 in homework 4. Quite a few students followed this procedure. We can express this 1-DL as a linear threshold function: $x_4 - 1 - \frac{1}{2}x_1 + \frac{1}{4} < 0 \leftrightarrow Yield = low$. Thus our (one of the acceptable) linear threshold function is $x_4 - \frac{x_1}{2} - \frac{3}{4}$.

Online Learning in Infinite Attribute Space [25 points]

In this problem we consider learning in a very large, possibly infinite, attribute space.

In the standard learning model we represent examples as bit vectors $\mathbf{x} \in \{0,1\}^n$, with the interpretation that the ith bit of the vector is 1 if the element described by $\mathbf{x}$ has the i-th attribute. **In the new model we will represent each example as a *list* of all attributes the example *has*, leaving unmentioned all the attributed it does not have.**

Recall the Elimination Algorithm for **Monotone Disjunctions**. Your job is to develop a version of this algorithm in the infinite attribute space. Formally, you need to learn a function f that is a monotone disjunction over a finite set of attributes selected from an infinite attribute space $X = \{x_1, x_2, x_3, \ldots\}$. You are given that **there is** a monotone disjunction which is consistent with the data.

You will use the on-line learning model. In this model, learning proceeds in a sequence of stages. In each stage you are given an example $\mathbf{x}$, asked to predict the value of $f(\mathbf{x})$, and then told whether or not your prediction was correct. You will quantify the performance of your algorithm in terms of the number of mistakes the algorithm makes.

(a) [15 points] Develop a learning algorithm for the class of monotone disjunctions over the infinite attribute domain X. Your description of the algorithm should include:

1. [3 points] How to initialize the hypothesis h? Justify.
2. [6 points] How do you update h when you make a mistake on a positive example? Justify.
3. [6 points] How do you update h when you make a mistake on a negative example? Justify.

- Let the initial hypothesis be $h \equiv 0$ which always says "negative".
- On the first positive example: let h be the disjunction of all attributes present in it.
- When h makes a mistake on a positive example: add all attributes in x to the disjunction h. Note that at least one of these attributes must be in the target disjunction.
- When h makes a mistake on a negative example: eliminate all attributes in x from the disjunction h. Note that none of these can be present in the target disjunction.

(b) [10 points] Let n be the largest number of attributes you see active in any given example, and r the number of attributes in the target disjunction we are trying to learn.

Bound the number of mistakes your algorithm will make as a function of n and r. Justify your bound clearly and concisely.

[Suggestion: in order to derive the bound, consider separately the number of mistakes your algorithm makes on positive examples and on negative examples.]

The algorithm makes at most r mistakes on positive examples. The reason is that on each such mistake at least one attribute from the target disjunction is added to h, and no attribute that is part of the target disjunction is ever eliminated.

The algorithm makes $(n-1)r$ mistakes on negative examples. To see that notice that each mistake on a positive example may add up to $n-1$ irrelevant attributes to h. That is, $r(n-1)$ might be added. In the worst case, they will be eliminated one by one, when h makes mistakes on negative examples.

The total number of mistakes in $r + (n-1)r = nr$.

Perceptrons [25 points] Answer deliberately withheld.

In this question, we will be asking you about Perceptrons and their variants.

Let $D = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$, where the j-th example $\mathbf{x}^{(j)}$ is associated with the label $y^{(j)} \in \{-1, +1\}$. Each example $\mathbf{x}^{(j)}$ is a bit-vector of length n, i.e. $\mathbf{x}^{(j)} \in \{0,1\}^n$, with the interpretation that the i-th bit of the vector $(x_i^{(j)})$ is 1 if the element described by $\mathbf{x}^{(j)}$ has the i-th attribute on.

(a) [7 points] Let us first consider a Perceptron where the positive example $\mathbf{x}$ satisfies $\mathbf{w} \cdot \mathbf{x} \geq \theta$, where $\mathbf{w} \in \mathbb{R}^n$, $\theta \in \mathbb{R}$ and $\mathbf{x}$ is some example $\mathbf{x}^{(j)}$ from D.

1. [3 points] Suggest an equivalent representation of this Perceptron in the form of $\underline{\mathbf{w}' \cdot \mathbf{x}' \geq 0}$ given an example $\mathbf{x}^{(j)}$, where $\mathbf{x}' \in \{0,1\}^{n'}$ for some suitable integer n'.

   Define $n' =$ _____

   Define $\mathbf{w}' =$ _____

   Define $\mathbf{x}' =$ _____

2. [4 points] In the following table, we describe a specific data set S. Using an initialization of $\mathbf{w}' = \mathbf{0}$, i.e. the zero vector, and a learning rate of $R = 1$, complete the columns under **(a)** of the table using the Perceptron learning algorithm.

	S			(a)		(b)	
j	$\mathbf{x}_1^{(j)}$	$\mathbf{x}_2^{(j)}$	$y^{(j)}$	Mistake? Y/N	Updated $\mathbf{w}'$	Mistake? Y/N	Updated $\mathbf{w}'$
Initialization				——	$\mathbf{0}$	——	$\mathbf{0}$
1	1	1	+1				
2	1	0	−1				
3	0	1	+1				

(b) [7 points] Using the same data set used above, we now consider a Perceptron with margin $\gamma > 0$. We can also represent this with $\mathbf{w}' \cdot \mathbf{x}' \geq 0$ like in Perceptron but using a different update rule for the weights.

1. [3 points] Let the margin $\gamma > 0$ and learning rate $R > 0$. For a given $(\mathbf{x}^{(j)}, y^{(j)})$, write down the update rule for the Perceptron with margin.

   If _____ $\leq$ ____ then $\mathbf{w}' =$ _____

   otherwise $\mathbf{w}' =$ _____

2. [4 points] We described a specific data set S in a table earlier. Using an initialization of $\mathbf{w}' = \mathbf{0}$, that is, the zero vector, a learning rate of $R = 1$ and margin $\gamma = 1.5$, complete the columns under **(b)** of the table using the *Perceptron with margin* learning algorithm.

(c) [11 points] Suppose we have the same data set S and now we would like to learn a linear separator of the form $\mathbf{w}' \cdot \mathbf{x}' \geq 0$, the canonical representation for any separating hyperplane. This time however, we would like to learn the weights $\mathbf{w}'$ by *minimizing* the error made by the linear separator over S.

We define the error made by $\mathbf{w}'$ over S using the *hinge loss* function, defined as $L(y^{(j)}, \mathbf{x}^{(j)}, \mathbf{w}') = \max(0, 1 - y^{(j)}\mathbf{w}' \cdot \mathbf{x}^{(j)'})$, where $\mathbf{x}^{(j)'}$ is the representation of example $\mathbf{x}^{(j)}$ in the form of $\mathbf{x}'$ in the canonical representation.

Thus the goal of learning is to minimize the following error:

$$\text{Error}(\mathbf{w}', D) = \sum_{j=1}^{m} L(y^{(j)}, \mathbf{x}^{(j)}, \mathbf{w}') = \sum_{j=1}^{m} \max(0, 1 - y^{(j)}\mathbf{w}' \cdot \mathbf{x}^{(j)'})$$

One way to do this is to make use of Stochastic Gradient Descent.

1. [9 points] Write the pseudocode for Stochastic Gradient Descent using this hinge loss function with a fixed learning rate of $R > 0$.

2. [2 points] Suggest a condition on the problem definition that will make the Stochastic Gradient Descent algorithm identical to the Perceptron with Margin algorithm.

Kernels [25 points]

In this question we will develop a learning algorithm that will take as input a URL and classify it according to whether it is relevant to the topic "Machine Learning" or not. The classifier will only depend on the string of the URL, and not on the web page itself.

In the following we will develop a kernel that will be used to learning how to map a URL string to "relevant" and "irrelevant".

We are given a collection of m URLs $u_1, u_2, \ldots, u_m$. Each URL consists of characters taken from a vocabulary V of n characters $c_1, ..., c_n$. We can assume that V includes *all* ASCII characters.

The *basic* feature vector for each URL u is $F(u)$. $F(u)$ is a binary vector, $F(u) \in \{0,1\}^n$, where the jth component in $F(u)$ indicates whether the character c_j appears in URL u ($F(u)[j] = 1$) or not ($F(u)[j] = 0$). For example, for the URL u = *www.cnn.com*, the set of active features in $F(u)$ is $A = \{\boldsymbol{w}, \boldsymbol{c}, \boldsymbol{n}, \boldsymbol{.}, \boldsymbol{o}, \boldsymbol{m}\}$, i.e. components in $F(u)$ that correspond to the indices of the characters of A will be 1, all others will be 0.

Each u is also labeled as relevant ($l = 1$) or irrelevant ($l = 0$).

(a) [17 points] The presence of some *set* of characters can be indicative of "machine learning", e.g. *ml, sv.* So in addition to the basic features in $F(u)$, we want to include features that indicate if a *different* pair of characters c_i and c_j appear anywhere in the URL u. Let us call this new feature space $\phi(u)$.

1. [3 points] What is the total number of features in the expanded feature space of $\phi(u)$ (as a function of the vocabulary size n)?
   $n + \binom{n}{2}$
2. [3 points] Assume a URL u = *www.a.sg*
   Write down the active features in the expanded feature vector $\phi(u)$.
   $A = \{w, ., a, s, g, w., wa, ws, wg, .a, .s, .g, as, ag, sg\}$
3. [3 points] For a URL u of length s, where all the s characters are different, what is the number of active features in $\phi(u)$ (as a function of s)?
   $s + \binom{s}{2}$

4. [8 points] We want to use the Kernel Perceptron algorithm to learn the function above. Design a kernel $K(u_1, u_2)$ that allows us to compute the value of the dot product $\phi(u_1)\phi(u_2)$ in time linear in n by directly computing it in the $F(u)$ space without expanding to the $\phi(u)$ space (assume that length of u_1 and u_2 is at most n). Give the formula for $K(u_1, u_2)$ and explain why it is true.

$K(u_1, u_2) = same(u_1, u_2) + \binom{same(u_1,u_2)}{2}$ where $same(u_1, u_2)$ is the number of characters that appear both in u_1 and u_2

(b) [8 points] Now we are going to define a new feature space $\psi(u)$. $\psi(u)$ will include all features from $F(u)$ and also include features that indicate whether a pair of characters appear consecutively in URL u. For example, for the URL $u =$ *www.cnn.com*, the set of active features in $\psi(u)$ will be $A = \{$***w***, ***c***, ***n***, ***.***, ***o***, ***m***, ***ww***, ***w.***, ***.c***, ***cn***, ***nn***, ***n.*** , ***co***, ***om***$\}$.

1. [2 points] What is the size of the expanded feature space $\psi(u)$ (as a function of the vocabulary size n)?
   $n + n^2$
2. [3 points] For a URL u of length s where all the s characters are different, what is the number of active features of $\psi(u)$ (as a function of s)?
   $s + s - 1$
3. [3 points] Write a kernel $K(u_1, u_2)$ that can compute the dot product $\psi(u_1)\psi(u_2)$ in time linear in n (assume that length of u_1 and u_2 is at most n).
   Since the length of $\psi(u_1)$ and $\psi(u_2)$ are linear in n, computing the dot product directly will be time linear in n.$K(u_1, u_2) = \psi(u_1).\psi(u_2)$.

Some formulae you may need

- $P(A, B) = P(A|B)P(B)$
- $Entropy(S) = -p^+ \log(p^+) - p^- \log(p^-) = -\sum_{i=1}^{k} p_i \log(p_i)$, where k is number of values
- $Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$
- $\log\left(\frac{a}{b}\right) = \log(a) - \log(b)$
- $\log_2(3) \approx \frac{3}{2}$