CS446: Machine Learning **Fall 2015**

October 27th, 2015

- 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%.
- 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.
- **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		/29
Kernels		/25
Online Learning		/25
Decision Trees		/20
Total		/100

Short Questions [29 points]

(a) **(5 points)** In the following definition of PAC learning we left a few blank fields. Fill in the blanks by choosing, for each empty field, one of the options given below. Note that under each line defining a blank we provided a small set of options for you to choose from.

(a) δ	(b) ϵ	(c) $1/\delta$	(d) $1/\epsilon$	(e) $1-\delta$	(f) $1-\epsilon$
(g) m	(h) n	(i) $n\epsilon/\delta$		(j) size($\mathbf{H}$)	
(k) number of examples		(l) instance size		(m) computation time	
(n) linear		(o) polynomial		(p) exponential	
(q) $\frac{1}{2}-\gamma$	(r) $\frac{1}{2}+\gamma$	(s) $1-\gamma$			

A concept class $\mathbf{C}$ defined over the instance space $\mathbf{X}$ (with instances of length n) is *strongly* PAC learnable by learner $\mathbf{L}$ using a hypothesis space $\mathbf{H}$ if for all concepts $f \in \mathbf{C}$, for all distributions $\mathbf{D}$ on $\mathbf{X}$, and for all fixed $\delta, \epsilon \in [0,1]$, given a sample of m examples sampled independently according to the distribution $\mathbf{D}$, the learner $\mathbf{L}$ produces with a probability ___at least___ {at least | at most | equal to} ___$1-\delta$___ {one of (a) to (f)} a hypothesis $g \in \mathbf{H}$ with error ___at most___ {at least | at most | equal to} ___ϵ___ {one of (a) to (f)} where the ___number of examples___ {one of (k) to (m)} is ___polynomial___ {one of (n) to (p)} in ___n___, ___$1/\delta$___, ___$1/\epsilon$___, and ___size($\mathbf{H}$)___ {four of (a) to (j)}.

(b) **(8 points)** Consider the following algorithm $\mathcal{B}$ for functions in C. $\mathcal{B}$ has the property that given any polynomial size sample of m labeled examples $\{(x_1, c(x_1)), \ldots (x_m, c(x_m))\}$ according to some $c \in C$, $\mathcal{B}$ outputs a hypothesis $h \in H$ that is incorrect on at most one of the m examples.

Then, the concept class C ___is PAC learnable___ {PAC learnable | not PAC learnable}.

Justify your answer.

For the polynomial sized $m > 2$, $\mathcal{B}$ can achieve an error of at most $1/m < 1/2$ given any sample and hence $\mathcal{C}$ is weakly PAC learnable. By applying boosting, we can achieve an error of at most ϵ with probability $1 \geq 1-\delta$ for any $\epsilon,\ \delta > 0$.

(c) **(8 points)**

In this question we consider the Winnow algorithm.

(1) In class we proved that Winnow can learn k-monotone disjunctions over a domain of n variables. State the upper bound on the number of **mistakes** Winnow will make.

Winnow will make $O(k \log n)$ mistakes when learning k-literal monotone disjunction using n variables.

(2) We consider a learning problem of the domain $X = \{1, 2, \ldots, N\}^d$. A d-dimensional hyper-rectangle over this domain X is a subset $c \subseteq X$ defined by $2d$ values: $1 \leq a_i \leq b_i \leq N$, for $i = 1, \ldots, d$.

The subset is

$$c = \{(x_1, \ldots, x_d) \in X : a_i \leq x_i \leq b_i \forall i = 1, \ldots, d\}.$$

Let RECT denote the class of all such d-dimensional hyper-rectangles over X.

i. Show that you can use Winnow to learn the concept class RECT. Justify your answer.

Winnow learns linear functions over its feature space; in order to use Winnow we therefore need to expand the feature space so that the target concept becomes linear in the new feature space. One direct way is to define a new feature space Y of features of the form:

$$y_i^{a,b} \equiv a \leq x_i \leq b \qquad \forall i = 1, \ldots, d, 1 \leq a \leq b \leq N.$$

Clearly, $|Y| = dN^2/2$, and in this space, a hyper-rectangle is a conjunction of d variables. This is a linear function and therefore Winnow can learn it.

ii. What is the mistake bound of your algorithm? Explain.

First note that the mistake bound was proved for k-monotone disjunctions. However, using De Morgan law it is easy to see that it holds also for k-monotone conjunctions.

The dimensionality of the new feature space is $n = dN^2/2$. The number of relevant features is $k = d$. Consequently, the number of mistakes is $O(d \log dN^2/2) = O(d \log N)$.

(d) **(8 points)** We define a set of concepts

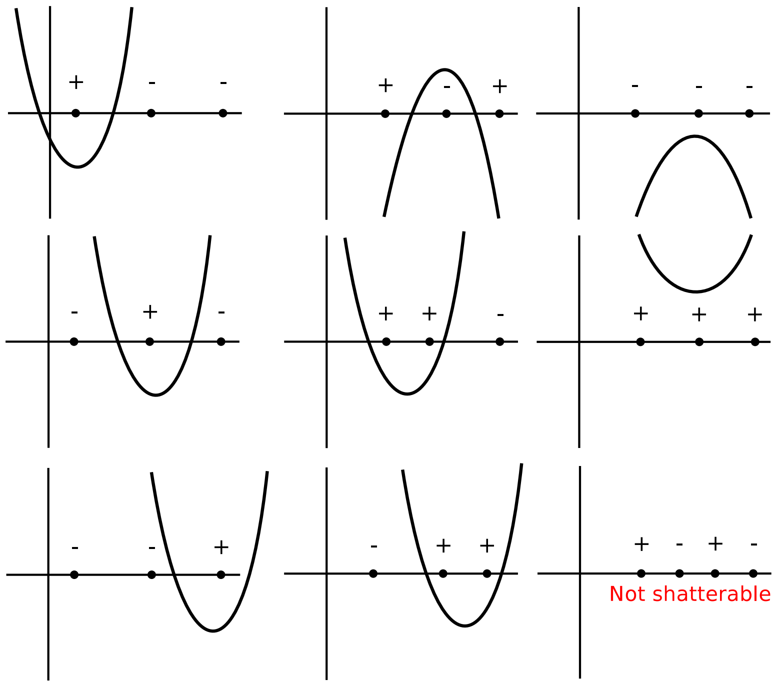
$$H = \{sgn(ax^2 + bx + c); a, b, c, \in R\},$$

where $sgn(z)$ is 1 when z is positive, and 0 otherwise. What is the VC dimension of H? Prove your claim.

The VC dimension is 3. To prove this, we need to show that there is one configuration of three points such that all its labelings can be shattered, and that no set of 4 points can be shattered. Note that, from the definition of H we are only dealing with points on the x axis (although the VC dimension is still 3 in two dimensions).

The case of 3 can easily be verified by checking the 8 possible labelings.

And, any alternating labeling of four points will result in a configuration that cannot be shattered because quadratic functions can change signs at most twice.



Kernels [25 points]

In this question we will define kernels, study some of their properties and develop one specific kerenl.

(a) **(1 point)** Choose one of the options below:
A function $K(x,z)$ is a valid kernel if it corresponds to an $\underbrace{\text{inner(dot) product}}_{\{\text{"inner(dot) product" | "sum"}\}}$ in some feature space, between feature representations that correspond to x and z.

(b) **(12 points)** In the next few questions we guide you to prove the following property of kernels:

Linear Combination Property: if $\forall i$, $k_i(x,x')$ are valid kernels, and $c_i > 0$ are constants, then $k(x,x') = \sum_i c_i k_i(x,x')$ is a valid kernel.

i. Given a valid kernel $k_1(x,x')$ and a constant $c > 0$, use the definition in(a) to show that $k(x,x') = ck_1(x,x')$ is also a valid kernel.

Since $k_1(x,x')$ is a vaild kernel, it can be written as a dot product $k_1(x,x') = \phi(x).\phi(x')$, for some $\phi(x)$ that is a feature mapping function that maps x to a feature vector in some (possibly larger dimensionality) space.
We define a new feature mapping function $\phi'(x) = \sqrt{c}\phi(x)$. We get that:

$$k(x,x') = ck_1(x,x') = \phi'(x).\phi'(x'),$$

so $k(x,x')$ can also be written as a dot product and it therefore a vaild kernel.

ii. Given valid kernels $k_1(x,x')$ and $k_2(x,x')$, use the definition in (a) to show that $k(x,x') = k_1(x,x') + k_2(x,x')$ is also a valid kernel.

Since $k_1(x,x')$ is a vaild kernel, we can write $k_1(x,x') = \phi_1(x)\phi_1(x')$, where $\phi_1(x)$ is a feature mapping function:

$$\phi_1(x) =< f_1(x), f_2(x), \ldots, f_n(x) >$$

Similarily, we can write $k_2(x,x') = \phi_2(x)\phi_2(x')$ where

$$\phi_2(x) =< g_1(x), g_2(x), \ldots, g_m(x) >$$

Now, we define a new feature mapping function with a range of dimensionality $n+m$:

$$\phi_3(x) =< f_1(x), f_2(x), \ldots, f_n(x), g_1(x), g_2(x), \ldots, g_m(x) > .$$

We get that

$$\phi_3(x)\phi_3(x') = \phi_1(x)\phi_1(x') + \phi_2(x)\phi_2(x') = k_1(x,x') + k_2(x,x') = k(x,x').$$

So, $k(x,x')$ is a vaild kernel.

iii. Conclude that the Linear Combination Property holds.

From i, we know that

$$\forall c_i > 0, \;\; c_i k_i(x,x') \text{ is a valid kernel.}$$

From ii, by induction, we know that (N^+ denotes $\{1,2,3,\cdots\}$)

$$\forall t \in N^+, \;\; \sum_{i=1}^{t} c_i k_i(x,x') \text{ is a valid kernel.}$$

To see that, observer that (initial conditions):

$$c_1 k_1(x,x') \text{ is a valid kernel}$$

$$c_1 k_1(x,x') + c_2 k_2(x,x') \text{ is a valid kernel}$$

Then suppose we have valid kernels

$$\sum_{i=1}^{t-1} c_i k_i(x,x') \text{ and } c_t k_t(x,x')$$

We know from ii that

$$\sum_{i=1}^{t} c_i k_i(x,x') \text{ is also a valid kernel}$$

for $\forall t \in N^+$.

This concludes that $k(x,x') = \sum_i c_i k_i(x,x')$ is a valid kernel.

(c) **(12 points)** In order to learn functions over sets, we represent sets as feature vectors in an n dimensional space, and define a kernel in this space. Our instances are all subsets of a set S, of size $|S| = m$. The n dimensional feature representation of $A \subseteq S$ is:

$$\varphi(A) = (\phi_{U_1}(A), \phi_{U_2}(A), \ldots, \phi_{U_n}(A)),$$

where $U_1, U_2, \ldots, U_n$ are all the subsets of S, and the coordinates are defined using the following feature mapping function:

$$\phi_U(A) = \begin{cases} 1, & \text{if } U \subseteq A \\ 0, & \text{otherwise.} \end{cases}$$

That is, $\varphi(A) \in \{0,1\}^n$, where $n = 2^m$.
Let A, B be subsets of S. We define the kernel

$$K(A,B) = \varphi(A)^\top \varphi(B).$$

Show that $K(A,B)$ can be computed in time that is polynomial in m.

$$\varphi(A) = (\phi_{U_1}(A), \phi_{U_2}(A), \ldots, \phi_{U_n}(A))$$
$$\varphi(B) = (\phi_{U_1}(B), \phi_{U_2}(B), \ldots, \phi_{U_n}(B))$$

So, we have

$$K(A,B) = \varphi(A)\varphi(B) = \sum_{i=1}^{2^m} \mathbf{1}\{\phi_{U_i}(A) = 1, \phi_{U_i}(B) = 1\},$$

that is, we add 1 to the sum iff the corrspoding coordinates in the dot product are both 1. When $\phi_{U_i}(A) = 1$ and $\phi_{U_i}(B) = 1$, we know that

$$U_i \subseteq A \text{ and } U_i \subseteq B$$

Thus, we get

$$U_i \subseteq A \cap B$$

This leads to

$$K(A,B) = \sum_{U_i} \mathbf{1}\{U_i \subseteq A \cap B\} = 2^{|A \cap B|}$$

It take $O(m)$ (linear) time to compute $|A \cap B|$. Then it takes constant time to futher compute $K(A,B)$ as $2^{|A \cap B|}$ (or $\log(m)$ time depending on how we compute the exponential function).
This concludes that $K(A,B)$ can be computed in time that is polynomial in m.

On-Line Learning [25 points]

In this problem we will consider a few variations of the Perceptron Learning algorithm and their properties. Recall the perceptron algorithm,

```
1: for $(x, y) \in \{x_i, y_i\}_{i=1}^T$ do
2:     if $y(w^\intercal x) \leq 0$ then
3:         $w' \leftarrow w + \eta y x$
4:         $w \leftarrow w'$
5:     end if
6: end for
7: return $w$
```

(a) **(10 points)** Suggest a variation of the preceptron update rule (line 3) which has the following property:

If the algorithm sees two consecutive occurrences of the same example, it will never make a mistake on the second occurrence.

(Hint: determine an appropriate learning rate that guarantees this property). Your work will result in one of the options below.

Which one? d
{one of (a) to (d)}

(a) $\eta \geq \frac{1-y(w^\intercal x)}{\|x\|^2+1}$ (b) $\eta \leq \frac{-y(w^\intercal x)}{\|x\|^2+1}$ (c) $\eta \leq \frac{-y(w^\intercal x)}{\|x\|^2}$ (d) $\eta \geq \frac{-y(w^\intercal x)}{\|x\|^2}$

Prove that your proposed variation satisfies the property stated above (that is, show how you derive the learning rate you chose).

Let w_i be the weight before making the mistake and w_{i+1} be the updated weight. We want $yw_i^T x < 0$ but $yw_{i+1}^T x \geq 0$

$$
\begin{aligned}
w_{i+1} &= w_i + \eta y x \\
y w_{i+1}^T x &= y w_i^T x + \eta y^2 \|x\|^2 \geq 0 && \text{(from prev. equality, and the requirement)} \\
y w_i^T x + \eta y^2 \|x\|^2 &\geq 0 && \text{(as y=+1 or -1)} \\
y w_i^T x + \eta \|x\|^2 &\geq 0 \\
\eta &\geq \frac{-y w_i^T x}{\|x\|^2}
\end{aligned}
$$

(b) **(5 points)** Consider the following variant of perceptron: Instead of *returning* w in line 7, return $\frac{w}{\|w\|}$.

Choose one of the following options and justify your answer: The decision boundary of the variant and the standard perceptron are $\underset{\{\text{same} \mid \text{different}\}}{\underline{\quad\text{same}\quad}}$ because:

The decision boundary are the same because,

$$sgn(w^T x) = sgn(\frac{w^T x}{\|w\|})$$

since $\|w\| > 0$. Thus making the final prediction with a rescaled version of the original w makes no difference.

(c) **(5 points)** Consider the following variant of perceptron: *Following each mistake*, instead of the usual update in line 3, perform the following update,

$$w' \leftarrow \frac{w + \eta y x}{\|w + \eta y x\|}.$$

Choose one of the following options and justify your answer: The decision boundary of the variant and the standard perceptron are $\underset{\{\text{same} \mid \text{different}\}}{\underline{\quad\text{different}\quad}}$ because:

You can get intuition for answering this question but thinking about (a) above. There, rescaling the weight vector made the difference between predicting correctly on example x or making a mistake on it. To make this concrete, it's best to consider a specific scenario.

Consider 2 training examples $\{(0,5), -1\}$ and $\{(5,0), +1\}$. Without loss of generality, we can take $\eta = 1$.

We start with $w_0 = (0,0)$,

With usual updates, we get

$$w_1 = w_0 - (0,5) = (0,-5)$$
$$w_2 = w_1 + (5,0) = (5,-5)$$

With the modified updates, we get

$$w_1 = \frac{w_0 - (0,5)}{\|w_0 - (0,5)\|} = (0,-1)$$
$$w_2 = \frac{w_1 + (5,0)}{\|w_1 + (5,0)\|} = \frac{(5,-1)}{\sqrt{26}}$$

The resulting ws are not the same and there is at least one example x (find it) on which they will make a different prediction. Hence the decision boundary are not same.

(d) **(5 points)** Derive the SGD update for the SVM algorithm, which has the following loss:

$$\|w\|^2 + \sum_{i=1}^{M} \max(0, 1 - y_i(w^\intercal x_i))$$

(disregard the single point where the objective function is not differentiable.)

Denote the gradient of the loss function wrt w for a single example (x_i, y_i) by $g_i(w)$

$$g_i(w) = 2w + \begin{cases} -y_i x_i & \text{if } 1 - y_i(w^\intercal x_i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

The SGD update is

$$w_{t+1} \leftarrow w_t - \eta g_i(w_t)$$

$$w_{t+1} \leftarrow w_t - 2\eta w_t + \begin{cases} \eta y_i x_i & \text{if } 1 - y_i(w_t^\intercal x_i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Decision Trees [20 points]

(a) **(10 points)** Let x be a vector of n Boolean variables and let $k \leq n$ be an integer. We define the class P_k of k-parity functions over the n variables. A k-parity function f_S is defined as follows: a set $S \subseteq \{x_1, \ldots, x_n\}$ is chosen such that $|S| = k$. Let $x = (x_1, x_2, \ldots x_n) \in \{0,1\}^n$. Then $f_S(x) = 1$ iff an odd number of variables in S are set to 1 in x.

Example: Let $n = 3$ and consider functions in P_2. Let $S = \{x_1, x_2\}$. Then $f_S(100) = f_S(101) = f_S(010) = f_S(011) = 1$, and $f_S(000) = f_S(001) = f_S(110) = f_S(111) = 0$.

i. Fix $S \subseteq \{x_1, \ldots, x_n\}$, and assume that you are using ID3 to learn a decision tree from data that is consistent with f_S. Consider two variables, x_1, x_2 such that $x_1 \in S$ and $x_2 \notin S$. Which of these variables is more likely to be the root of the decision tree for f_S? $\underset{\{x_1\ ;\ x_2\}}{\underline{x_1}}$

Justify your answer:

x_1 is more likely to be the root of the decision tree. Consider the following two cases: (i) $S = \{x_1\}$, and (ii) $S = \{x_1, ..., x_s\}$. Now, for the first case, x_1 effectively separates the instances according to the target classification, and therefore is the best candidate for root. For the second case, eventhough x_2 is not part of the "ideal" DT, all variables, in fact, lead to maximum entropy, i.e., there's no likely winner. For instance, consider the example given in the problem description.

ii. Consider $f_S \in P_k$. State the *depth* of the smallest possible consistent decision tree for f_S in terms of n and k. Describe the shape of the decision tree for f_S.

Justify your answer.

The depth of the smallest possible consistent decision tree is k. This is because, any consistent decision tree will have to examine all variables in S to classify an instance. Note that, the relative order of these variables doesn't affect the classification/labellings.

(b) **(10 points)** Assume that you are using an implementation of ID3 that takes an upper bound on the depth of the output decision tree as a parameter. You will use this implementation of ID3 to learn from a dataset D_{train}, compute the empirical error, and then evaluate the learned tree on a test set D_{test}. You will learn two trees, T_k, learned with bound k on the depth, and T_m, learned with a depth bound m.

Assume **k** $<$ **m**. (But note that we say nothing about the size of k; it could be a very small number or a very large number).

i. Which tree, T_k or T_m, is likely to have *larger* empirical error (that is, error on D_{train})? $\underline{T_k}$

{T_k ; T_m ; impossible to tell}

Justify your answer:

T_k is likely to have larger empirical error on D_{train}. Since $k < m$, T_m will fit the training data better than T_k (or, no worse). T_k will therefore have larger error.

ii. Which tree, T_k or T_m, is likely to have larger error when tested on D_{test}? <u>impossible to tell</u>

{T_k ; T_m ; impossible to tell}

Justify your answer:

This is impossible to tell. On one end, if both heypotheses fit the training data euqalty well, T_k is likely to generalize better. On the other hand, if $k << m$ and T_k does not fit the training data well relative to T_m's performance on training, it is likely that it will do quite poorly on the test data too.

This page was intentionally left blank.