CS446: Machine Learning **Fall 2015**

Problem Set 4

Handed Out: October 13^{th}*,* 2015 *Due: October* 22^{th}*,* 2015

- Feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You should, however, write down your solution yourself. Please try to keep the solution brief and clear.
- Please use Piazza first if you have questions about the homework. Also feel free to send us e-mails and come to office hours.
- Please, no handwritten solutions. You will submit your solution manuscript as a single pdf file.
- Please present your algorithms in both pseudocode and English. That is, give a precise formulation of your algorithm as pseudocode and *also* explain in one or two concise paragraphs what your algorithm does. Be aware that pseudocode is much simpler and more abstract than real code.
- The homework is due at 11:59 PM on the due date. We will be using Compass for collecting the homework assignments. Please submit your solution manuscript as a pdf file via Compass (`http://compass2g.illinois.edu`). Please do NOT hand in a hard copy of your write-up. Contact the TAs if you are having technical difficulties in submitting the assignment.
- You cannot utilize the late submission credit hours for this problem set. We will release the solution immediately after it is due so that you can have time to go over it before the mid-term on Oct. 27th.
- No code is needed for any of these problems. You can do the calculations however you please. You need to turn in only the report.

1. **[VC Dimension - 30 points]**

   (a) **[15 points]** Assume that all examples are points in a two-dimensional space, i.e. $\mathbf{x} = \langle x_1, x_2 \rangle \in \mathbb{R}^2$. Consider the concept space of triangles in the plane. Hence, a concept $h \in \mathbf{H}$ is specified by three vertices $\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \langle x_3, y_3 \rangle$ (that are not on the same line). An example $\mathbf{x} \in \mathbb{R}^2$ is labeled as positive by h if and only if $\mathbf{x}$ lies inside or on the border of the triangle. Give the VC dimension of $\mathbf{H}$ and prove that your answer is correct. [Hint: When proving the upper bound of the VC dimension, use the concept of *convex hull* to distinguish among different situations. Provide illustrative pictures if necessary.]

   solution:

   Given: Points are collinear thus for determining the VC dimension d we have to prove that the maximum size of set that can be shattered is d and that no set of size d+1 can be shattered using the concept class that is a triangle in the above scenario. We will consider the general position of points i.e no 3 points can be collinear in the plane and easiest way to implement that is to consider the point in the circle

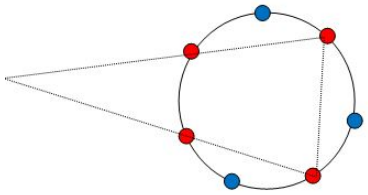

Figure 1: A set of size 7 shattered

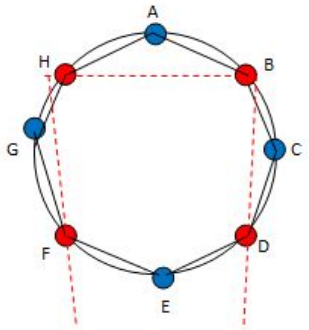

Figure 2: A set of size 8 not shattered

Analysis It can be illustrated using the below figure that any set of size 7 can be shattered using the triangle. Consider the arrangement of 7 points on the circle as shown in figure. The data can be labeled +,-,+,-,+,-,+,. + are represented by red circles and - are represented by blue circles. Now we can compute the convex hull of the positive points and the resultant convex hull encloses all the positive points as we need to show that any one set of size 7 can be shattered then the minimum VC dimension is 7. Moreover, in case of 7 points there can be maximum of 3 contiguous negative points on the circle and one edge of the triangle can be used to separate those three points and enclose rest of the four points.

□

Now, let us consider the case of 8 points on the circle as shown in figure 2. In case where 7 points are positive and 1 is negative, due to convexity of a triangle that is a triangle that encloses 7 points in its convex hull will always enclose the 8^{th} point in its convex hull. Thus, if 7 are positive and 1 is negative it can't be labeled. Similarly if we place the 8 points (labeled A-H) alternately as POS and NEG. Now if we join all the points we will get a convex polygon as show in figure below. Therefore to shatter the set, we require B,D,F,H to be in the same set and rest points outside the triangle. Lets see if that is possible. For the first set of points B and D, C should be outside the hyperplane separating B,D from C thus it will intersect lines BC and CD. Same is true for the points H and F. As shown in figure, remaining point E will always be enclosed in the traingle that is formed. Thus no set of size 8 can be shattered using the triangle and hence *VCdim(C) = 7*.

(b) **[15 points]** Consider the concept space of the union of k disjoint intervals in a real line. Hence, a concept $h \in \mathbf{H}$ is represented by $2k$ parameters $a_1 < b_1 < a_2 < b_2 < \ldots < a_k < b_k$. An example $x \in \mathbb{R}$ is labeled as positive by h if and only if x lies in one of the intervals $[a_p, b_p], p \in \{1, 2, \ldots, k\}$. Give the VC dimension of $\mathbf{H}$ and prove that your answer is correct. [Hint: Proving the lower bound of the VC dimension becomes easier if you divide it into simpler sub-problems.]

Solution:

Given: Concept space comprises of union of k disjoint sets of interval on real line. We can generalise the concept by looking at the set of two intervals and then generalising it to k intervals.
Let us take the value of $k = 1$ that is there is just single interval α_1 and α_2 where instance is positive if it lies withing the range that is α_1 <x< α_2 and negative if it lies outside the range. As we saw in the class that one interval can shatter 2 points and can not any three points +,-,+ can not be shattered using a single interval. Thus $VCdim = 2$ for a single interval i.e for $k = 1$ $VCdim = 2k = 2$. Now lets us introduce one more interval i.e. $k = 2$ taking the alternate labelling of POS and NEG +,-,+,-. Similar to aboe example we can shatter these instances by keeping the value of first positive within the interval $a_1 < x_1 < b_1$ and the third instance in interval $a_1 < x_3 < b_1$. Instances x_2 and x_4 are outside the interval and hence are negative. Thus minimum $VCdim = 4 = 2k$ for the above scenario. Now let us introduce another instance +,-,+,-,+, similar to above argument the set of size 5 can not be shattered using 2 intervals.
Intuitively, set of $2k + 1$ points labeled alternatively as POS and NEG, starting with a POS label, cannot be shattered using union of k intervals. Thus $VCdim = 2k$ for union of k intervals on the real line.

Grading note: You will not get any points without proper justification of your answer.

2. [**Decision Lists - 40 points**]

In this problem, we are going to learn the class of k-decision lists. A decision list is an ordered sequence of if-then-else statements. The sequence of if-then-else conditions are

tested in order, and the answer associated with the first satisfied condition is returned. The class of k-decision lists is a subset of the class of all decision lists, where the statements in each rule are of a bounded size. Formally, for a fixed k, we define:

Definition 1 *A* **k-decision list** *over the variables $x_1, \ldots, x_n$ is an ordered sequence $L = (c_1, b_1), \ldots, (c_\ell, b_\ell)$ and a bit b, in which each c_i is a conjunction of at most k literals over $x_1, \ldots, x_n$. The bit b_i is referred to as the bit* associated *with condition c_i, and b is called the* default *value. For any input $\mathbf{x} \in \{0,1\}^n$, $L(\mathbf{x})$ is defined to take the value b_j, where $j \in \{1, \ell\}$ is the smallest index satisfying $c_j(\mathbf{x}) = 1$; if no such index exists, then $L(\mathbf{x}) = b$.*

We denote by **k-DL** *the class of concepts that can be represented by a k-decision list.*

Figure **??** shows an example of a 2-decision list over six variables, $x_1, \ldots, x_6$. For this decision list, $L(\mathbf{x}_1 = \langle 0, 1, 1, 0, 0, 1\rangle) = 1$ and $L(\mathbf{x}_2 = \langle 1, 0, 0, 1, 0, 0\rangle) = 0$, where in the case of $\mathbf{x}_1$ no condition is satisfied, so the labels is determined by the default bit, and for $\mathbf{x}_2$, the first condition is satisfied.

(a) **[5 points]** Show that if a concept c can be represented as a k-decision list so can its complement, $\neg c$. You can show this by providing a k-decision list that represents $\neg c$, given $c = \langle (c_1, b_1), \ldots, (c_\ell, b_\ell), b\rangle$.

Solution:

Given: There is a concept class of K-DL such that $c = \langle (c_1, b_1), \ldots, (c_\ell, b_\ell), b\rangle$. Moreover, we know that its complement will be the complement of the vector b, which is $b = \langle b_0, b_1, b_2, \ldots, b_\ell\rangle$. Thus, we can define the $\neg c$ as $\langle (c_1, \neg b_1), \ldots, (c_\ell, \neg b_\ell), \neg b\rangle$.That is we can take the complement of b vector and add it to the values of corresponding function ϵ C. Effectively we just took the component wise complement of b vector and the output is another K-DL.

(b) **[10 points]** Show that

$$k\text{-DNF} \cup k\text{-CNF} \subseteq k\text{-DL}$$

Here, k-DNF is the disjunctive normal form with each conjunctive clause containing at most k literals. Similarly, k-CNF is the conjunctive normal form with each disjunctive clause containing at most k literals. [Hint: Use the relationship between k-DNF and k-CNF to help your proof.] **Solution:**

Given: A k-DNF is disjunction of conjunctions and can be effectively written as $\langle c_1 \wedge c_2 \wedge \ldots \wedge c_n \rangle$ where each term c is conjunction of k literals.
Similarly, A k-CNF is conjunction of disjunctions and can be effectively written as $\langle d_1 \wedge d_2 \wedge \ldots \wedge d_n \rangle$ where each term d is disjunction of k literals.
Proof: By De Morgan's law we know that:

i. The negation of a conjunction is the disjunction of the negations

ii. The negation of a disjunction is the conjunction of the negations

That is both k-DNF and k-CNF are duals of each other. Moreover we also know from the first part of this question that compliment of k-DL is also a k-DL
Effectively if we can prove that k-DNF $\subseteq$ k-DL then we can also prove that k-DNF $\cup$ k-CNF $\subseteq$ k-DL, one thing to be noted is that neither k-DNF nor k-CNF is the subset of other.
We will first prove that k-DNF $\subseteq$ k-DL. We know that for DNF if any one of the term d_i is on in the given set of k-DNF the concept c will be one. So we can design an appropriate k-DL which is on when anyone of the concept clauses are on in the list and off only when all the concept clauses evaluate to false, which is equivalent to saying $f = \langle (c_1, 1), (c_2, 1), \ldots, (c_\ell, 1), 0 \rangle$ therefore the concept represented by k-DNF $c = f$. Thus k-DNF $\subseteq$ k-DL.
Similarly, as per De Morgan's law we can represent a k-CNF as k-DNF by taking the compliment which is: $\neg d = \langle \neg d_1 \vee \neg d_2 \vee \ldots \vee \neg d_n \rangle$ where $\neg d_i$ is CNF of k terms. As proved above k-DNF $\subseteq$ k-DL which implies that $\neg d \subseteq k$-DL and as stated above complement of a function in k-DL is also a k-DL. Therefore, $d \subseteq k$-DL.
Combining the above two we get

$$k\text{-DNF} \cup k\text{-CNF} \subseteq k\text{-DL}$$

.

(c) **[15 points]** Let S be a sample data set that is *consistent* with some k-decision list. Provide an algorithm to construct a k-decision list that is consistent with S. Prove the correctness of your algorithm. [Hint: Consider the set of all possible conjunctions. When proving the correctness, argue that your algorithm will terminate, and when it does, it indeed produces a k-decision list, which is consistent with the sample S.]

Solution:

Given: sample data set that is *consistent* with some k-decision list, in order to be consistent the resultant decision list should comprise of at most k size conjunctions which satisfy all the instances, that is $\forall x \in S$ there exists a $c_i \in C_k$, where C_k is the concept class of all the conjunction of literals of at most size k (includes empty sets as well), such that c_i is consistent or true for the example and for every c_i there will be a label $\hat{y}$ where $\hat{y}$ is label and $\in \{0,1\}$. Thus, for a concept to be added in the final hypothesis it should meet two conditions

i. There exists set of example $E_i(S) = \{x \in S | c_i(x) = 1\}$

ii. $\forall x \in E_i(S)$ corresponding label $y \in S = \hat{y}$

Intuitively, the algorithm will traverse through the concept space and finds the set of instances $E_i(S)$ which are consistent with the concept c_i. If both the above mentioned conditions are met corresponding concept is added to the hypothesis and all the instances for which the conditions were true were removed from the instance space and keeps on repeating it till all the examples are coveres or removed from the instance space S. Proposed algorithm is as following:

k-decision list, algorithm

–h $\leftarrow \emptyset$

–for $c_i \in C_k$

—-for $\hat{y} = 0, 1$

——FLAG = True

——$E_i \leftarrow \emptyset$

———for (x,y)$\in S$

————if $c_i == 1$

—————-if $\hat{y} == y$

——————$E_i \leftarrow E_i \cup (x, y)$

—————-else

——————FLAG = False

——————Break;

—————-end of if

————end of if

———end of for

—-if (FLAG == True)

——$h \leftarrow h \cup E_i$

——$S \leftarrow S - E_i$

—-end of if

—-end of for

—if (S == $\emptyset$)

—-Break;
—end of if
–end of for

Correctness: The algorithm terminates when there are no more instances left in S. Output of the algorithm is set of D = $\langle c_i, y \rangle$ which is nothing but a decision list. The conditions mentioned in the previous section ensures that concepts which are consistent with that of a set of examples and their respective labels are added to the final hypothesis. Moreover, we are removing the set of examples which are consistent and added to the hypotheis which ensures that only the first concept c_i which is consistent with the data is only added to the hypothesis as there can be another c_j which is consistent, but only the first one is added.
Delving further,FLAG variable ensures that no concept for which any two instances, which are consistent with it, but with different labels is added to the final hypothesis. Which ensures correct labelling. As, concept space is very large and we are iteratively calling the same function, algorithm will halt and is certain to cover all the instances.

(d) **[10 points]** Use Occam's Razor to show:
For any constant $k \geq 1$, the class of k-decision lists is efficiently PAC-learnable.
[Hint: Consider the number of different k-DL functions.]

Solution:

There are two limitations to PAC learn-ability if both of these are satisfied for any learner than that learner is PAC learn-able. We will evaluate k-DL function for both the criterion and see if its PAC PAC learn-able or not:

i. **Polynomial Sample Complexity:** Let —C_k^n— be the number of concepts of size at most K conjunctions drawn from the set of all literals L_n where n is the dimension of the instance space.Same results can be proved for —D_k^n— where D represents the set of disjunctions of size at most k

$$|C_k^n| = |D_k^n| = \sum_{i=0}^{k} \binom{2n}{i} = \mathcal{O}(n^k)$$

Now each of these concepts in a K-DL can be absent, paired with 0 or paired with 1. Therefore there are 3 options for each element in —C_k^n—. Moreover, as the sequence of arrangement of clauses in the k-DL also matters which gives the $(|C_k^n|)!$ combinations. As we are trying to find the upper bound on the number of concept clauses we are considering the K-DL to comprise of all feasible 0 to k sized conjunction and hence the math. These leads us to defining the size of K-DL which is as below:

$$|H| = |k - DL| = \mathcal{O}(2^{|C_k^n|}(|C_k^n|)!)$$

The above equation is nothing but the size of hypothesis space. Now using the Occams razor bound, we can see that the number of examples we require in S is $log|H| = n^k log2 + kn^k logn$ (Using Stirlings formula for approximate log of factorial). As it can be seen the complexity is polynomial in n

ii. **Polynomial time Complexity:** The algorithm is polynomial in time can be easily seen by the fact that $|C_k^n| = \mathcal{O}(n^k)$ and thus the running time will also be a polynomial in n.

Thus K-DL is polynomial in both space and time and hence it is PAC learnable.

3. [**Constructing Kernels - 30 points**]

For this problem, we wish to learn a Boolean function represented as a **monotone** DNF (DNF without negated variables) using kernel Perceptron. For this problem, assume that the size of each term in the DNF is bounded by k, i.e., the number of literals in each term of the DNF is between 1 to k. In order to complete this task, we will first define a kernel that maps an example $\mathbf{x} \in \{0,1\}^n$ into a new space of conjunctions of **upto** k different variables from the n-dimensional space. Then, we will use the kernel Perceptron to perform our learning task.

(a) [**5 points**] Define a kernel $K(\mathbf{x}, \mathbf{z}) = \sum_{c \in C} c(\mathbf{x})c(\mathbf{z})$, where C is a family of monotone conjunctions containing **upto** k different variables, and $c(\mathbf{x}), c(\mathbf{z}) \in$

$\{0, 1\}$ is the value of c when evaluated on example $\mathbf{x}$ and $\mathbf{z}$ separately. Show that $K(\mathbf{x}, \mathbf{z})$ can be computed in time that is linear in n. [Hint: It would be useful to think about the case where each term of the DNF is *exactly* of k literals and then generalize.] **Solution:**

Given: A kernel function $K(\mathbf{x}, \mathbf{z}) = \sum_{c \in C} c(\mathbf{x})c(\mathbf{z})$ where C is a family of monotone conjunctions containing **upto** k different variable. Similar to what we saw in the lectures $\mathbf{z}$ is the union of number Promotions and Demotions $P \cup D$ while $\mathbf{x}$ is the instance. Now consider the case of K-DNF, now if we consider the inner product of $c(\mathbf{x})c(\mathbf{z})$ only the monomials which are on in the product will contribute to the sum, now we can determine the monomials which are on by looking at the number of common bits in both $\mathbf{z}$ and $\mathbf{x}$.So,any disjunction which has any bit that is on in both $\mathbf{x}$ and $\mathbf{z}$ will cause that particular term to be on in both of them. Thus effectively contributing to the sum.

$$K(\mathbf{x}, \mathbf{z}) = \sum_{c \in C} c(\mathbf{x})c(\mathbf{z}) = \sum_{i=0}^{k} \binom{same(x, z)}{i} = \mathcal{O}(n^k)$$

Thus $K(\mathbf{x}, \mathbf{z})$ is polynomial in time.

(b) **[10 points]** Write explicitly the kernel Perceptron algorithm that uses your kernel. [Hint: Think about the kernel based method.] **Solution:**

Kernel perceptron algorith is implemented similar to that of the perceptron with a altered classifier function $f(x) = sgn((\sum_{x_m \in M} y_m k(x, x_m)) + \theta)$ where M is the set of examples on which algorithm made mistakes before. And T is the set of all training examples in the blown up feature space i.e. T $\in \{X \times Y\}^n$

Algorithm:

Initialise $M \leftarrow \emptyset$

-for example $\langle x_i, y_i \rangle \in T$ do

—$\hat{y} = sgn((\sum_{x_m, Y_m \in M} y_m k(x, x_m)) + \theta)$

—if $y_i \neq \hat{y}$

—-M$\leftarrow M \cup \langle x_i, y_i \rangle$

—- $\theta \leftarrow \theta + y$

–end if

-end for

(c) **[15 points]** State a bound on the number of mistakes the kernel Perceptron algorithm will make on a sample of examples consistent with a function in this class. [Hint: In class, we proved a theorem by Novikoff; use this bound and estimate the two constants in it for this specific case. When doing it, observe that you are no longer learning in the original n dimensional space, but rather in a blown-up space that has a different dimensionality.]

Solution:

Given: A kernel perceptron in which a set of examples $S = \langle (x_1, y_1), (x_2, y_2), \ldots (x_t, y_t) \rangle \in (x, \{\pm 1\})$ and $K : x \times x \leftarrow \Re$ a kernel function which moved from weight vector space to just the example space.
For determining the mistake bound we need to first compute the value of R which is the size of vector $c(x)_i$ in blown up space. Thus $R = \sqrt{K(x^t, x^t)}$ where $t \in |T|$. As data is linearly seperable there exists a $\gamma^{'} > 0$ then for any u $\in$ x, $y_t(K(u, x_t) \geq \gamma^{'} \forall t \in |T|$ If the dimension of the weight vector in the original space was n then representing the same weight vector in terms of at k-DNF as explained in first part of the question, using the binomial theorem dimension of vector in blown up space will be of order = $\mathcal{O}(n^k)$. Now R is ℓ_2 norm of vector $c(x) \Rightarrow R = \sqrt{K(x^t, x^t} = \Theta(n^{k/2})$ so the quantity R is exponantially large.
We will now consider the specific case where the value of **k is fixed** and that is for a single function in a class the value of R^2 in that case will be of order $\mathcal{O}(n)$ .Where n is the number of features in blown space. Assuming that $c(x)_i$ in this case have one active feature corresponding to threshold θ as well.
While Novikoff's theorem doesn't require the γ to be largest margin, but using the largest margin data will always give the tighter upper bounds on the number of mistakes that perceptron makes. In the worst case scenario γ can be exponentially small in n. And if the data is well separated, γ can be large as compared to 1/n. This is called the large margin case. Now, large margins are generally found using SVM but in the given scenario an approximate maximum margin can be computed by altering the update rule. Such that we update the α whenever $sgn((\sum_{x_m, y_m \in M} y_m k(x, x_m)) + \theta)$ doesn't match the given label i.e only on the mistakes, if we update the hypothesis on mistakes as well as in cases where hypothesis was correct by a margin less than $(1 - \epsilon)\gamma$. This algorithm is also guaranteed to halt in polynomial in $1/\epsilon\gamma$.
Thus the same bound that holds for perceptron also holds for the kernel perceptron and the bound is $||u||R^2/\gamma^2$. The respective values of R and γ are as shown in the analysis, where $||u|| \leq 1$ and under the assumption that data is linearly seperable, otherwise we have to consider the slack variable for computing the same.

Submitted by: Shivam Upadhyay(upadhyy3)