CS446: Machine Learning	**Spring 2017**

Problem Set 4

Handed Out: Solution	*Due: N/A*

1. (a) A simple algorithm is as follows: let $S_+ = \{(x_i, y_i) | y_i = 1, i \in 1, 2, \ldots, m\}$ be the set of all positive examples. Set $r_1 = min_{x \in S_+} \|x\|$ and $r_2 = max_{s \in S_+} \|x\|$ for h_{r_1, r_2}.

   (b) i. Since the examples are generated according to $h^*_{r_1^*, r_2^*} \in \mathcal{H}_{2cc}$ and we are setting the radii of the hypothesis at positive examples, we have that $r_1^* \leq r_1$ and $r_2 \leq r_2^*$. Thus h_{r_1, r_2} will only make mistakes on unseen positive examples.

   ii. The probability of drawing a point outside the specified region is $1 - \epsilon$. Since the m data points are I.I.D. samples from D, the probability that none of them lie within the specified region is $(1 - \epsilon)^m$.

   (c) We fix ϵ to be an error value that we can tolerate. We know that the error is the probability of incorrectly classifying a positive example. The question we now ask is the following: how large should m be to make sure that, with high confidence (i.e. probability $1 - \delta$), our error will be less than this ϵ?

   In the previous part, we derived that the probability that the m examples used in the learning process were all outside the area of size ϵ is $(1 - \epsilon)^m$. We want this quantity to be *less* than δ (in other words, we want a small chance for this "bad event" to happen).

   We know that $(1 - \epsilon)^m < e^{-\epsilon m}$, and therefore we will find m so that $e^{-\epsilon m} < \delta$, which will also guarantee that $(1 - \epsilon)^m < \delta$.

   By taking the natural logarithm of both sides and rearranging, we thus get that

$$m > -\frac{1}{\epsilon} \ln(\delta) = \frac{1}{\epsilon} \ln \frac{1}{\delta}$$

   That is, if you use at least these many examples, with probability at least $1 - \delta$ the error of the learned hypothesis will be less than ϵ.

   One thing worth noting is that, in general, following a proof process like this one results in a *loose* bound of the number of examples strictly required to achieve the PAC learning conditions. In other words, it is possible that the actual number of samples we need to see in order to guarantee PAC learning is smaller than the value we derived above. For given values of δ and ϵ, the smallest number of samples that are needed for a learning algorithm to ensure PAC learning is called the *sample complexity* [1]. What we have done in this problem is find an upper

[1] That is, if you see more than this number of examples, you are guaranteed to have PAC learnability; if you see fewer than this number, there is a distribution and a hypothesis class such that there is **no algorithm** that, with confidence greater than 1-δ, will produce a hypothesis with error $< \epsilon$.

bound to the sample complexity, i.e. if we let $m(\delta, \epsilon)$ be the sample complexity for our algorithm, we have proven that

$$m(\delta, \epsilon) < \frac{1}{\epsilon} \ln \frac{1}{\delta}$$

(d) One can easily verify that the VC dimension of $\mathcal{H}_{cc} = 2$ (this is left as an exercise for the reader). There is a sample complexity bound for hypothesis classes with finite VC dimensions presented in the notes; substituting our value of VC Dimension, we get the following bound:
$m > \frac{1}{\epsilon}\left(16 \log \frac{13}{\epsilon} + 4 \log \frac{2}{\delta}\right)$

2. $VC(H) = 3$. We provide a geometric proof. See figure 1. The first 8 diagrams show that all possible dichotomies of three points can be shattered by the hypothesis. The eigth shows that 4 points cannot. Note: there's an inconsistency in the drawings. The polynomials that are shattering the dichotomies besides $+++$ and $---$ should be reflected about the x-axis. We'll accept either.

3. (a) In the dual representation, the weight vector can be presented as a collection of important examples M on which the Perceptron makes mistakes. We predict the label for a new example $\vec{\mathbf{x}}$ using the following equation:

DualPrediction(M, $\vec{\mathbf{x}}$)
return $Th_\theta\left(\sum_{(\vec{\mathbf{x}}_m, y_m)} y_m \vec{\mathbf{x}}^\top \vec{\mathbf{x}}_m\right)$

The training algorithm is as follows:

$M \leftarrow \emptyset$
for $(\vec{\mathbf{x}}, y) \in S$ **do**
  if $y \neq$ DualPrediction(M, $\vec{\mathbf{x}}$) **then**
    $M \leftarrow M \cup (\vec{\mathbf{x}}, y)$
  end if
end for

Note: we will accept simply writing the closed form of the dual perceptron.

(b) As mentioned in the notes, the sum of multiple kernels is a valid kernel; furthermore, multiplying a kernel by a constant c also produces a valid kernel. Thus, all that remains is to show that $(\vec{\mathbf{x}}^\top \vec{\mathbf{z}})^3$, $(\vec{\mathbf{x}}^\top \vec{\mathbf{z}})^2$, and $\vec{\mathbf{x}}^\top \vec{\mathbf{z}}$ are valid kernel functions. The last one is trivial. To show that $(\vec{\mathbf{x}}^\top \vec{\mathbf{z}})^3$ is a valid kernel, we can show that it is a dot product.

$$\begin{aligned}(\vec{\mathbf{x}}^\top \vec{\mathbf{z}})^3 &= x_1^3 z_1^3 + 3x_1^2 x_2 z_1^2 z_2 + 3x_1 x_2^2 z_1 z_2^2 + x_1^3 z_1^3 \\ &= \Phi(\vec{\mathbf{x}})^\top \Phi(\vec{\mathbf{z}})\end{aligned}$$

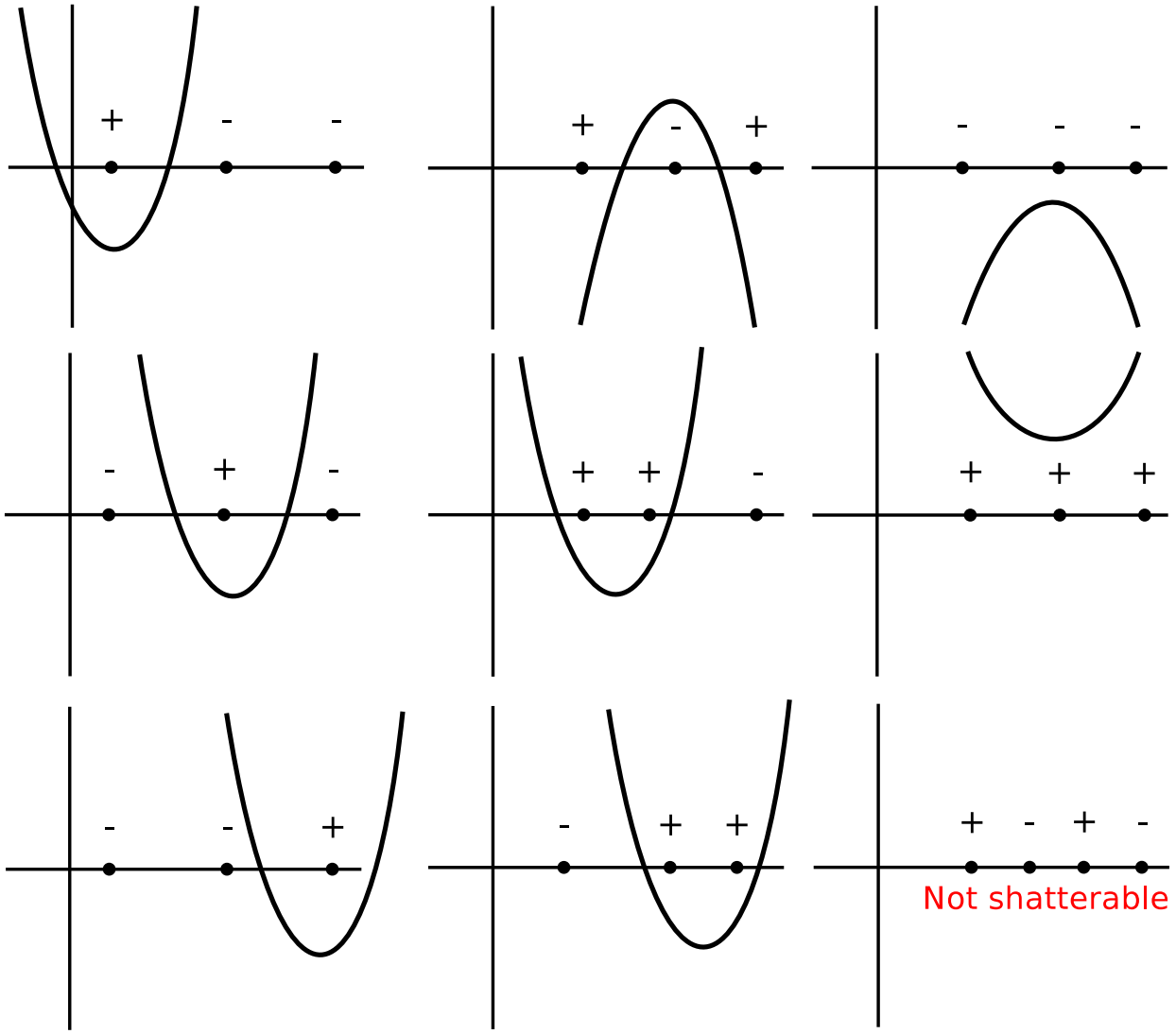

Figure 1: Geometric proof that $VC(H) = 3$.

where $\Phi(\vec{\mathbf{x}})$ is defined as

$$\Phi(\vec{\mathbf{x}}) = \begin{bmatrix} x_1^3 \\ \sqrt{3}x_1^2 x_2 \\ \sqrt{3}x_1 x_2^2 \\ x_2^3 \end{bmatrix}$$

A similar argument can be made for the other terms.

Alternatively, we can expand $K(x, z)$ and factor the terms of the expanded form to see that it is a dot product of

$$\Phi(\vec{\mathbf{x}}) \quad = \quad \langle x_1^3, x_2^3, \sqrt{3}x_1^2 x_2, \sqrt{3}x_1 x_2^2, 7x_1^2, 7\sqrt{2}x_1 x_2, 7x_2^2, , (14\sqrt{2} \ + \ 8)x_1, (14\sqrt{2} \ +$$

8)$x_2, 28\rangle$

(c) Consider each of the monotone conjunctions c containing exactly k different variables. $c(\mathbf{x})c(\mathbf{z}) = 1$ can only be achieved when both $c(\mathbf{x})$ and $c(\mathbf{z})$ are true (i.e. it returns 1). This means that c only involves variables that are true for both $\mathbf{x}$ and $\mathbf{z}$. We then count the number of these conjunctions c in the family of monotone conjunctions C containing exactly k different variables. This gives us,

$$K(\mathbf{x}, \mathbf{z}) = \binom{same^1(\mathbf{x}, \mathbf{z})}{k}$$

where $same^1(\mathbf{x}, \mathbf{z})$ is the number of common *active* features (features that are 1) of $\mathbf{x}$ and $\mathbf{z}$. Computing $same^1(\mathbf{x}, \mathbf{z})$ takes time linear in the number of features.

4. (a) i. $\mathbf{w} = \langle -1, -1\rangle$, $\theta = 0$ will work. Since we have no constraint on $\|w\|$ there are infinitely many separating hyperplanes.

ii. $\mathbf{w} = \langle -\frac{1}{2}, -\frac{1}{2}\rangle, \theta = 0$

iii. Assume $\mathbf{w}^* = (w_1, w_2)$, θ^* is the solution. We can directly find it as the largest margin hyperplane by geometry. By staring at the data long enough, one may suspect that the closest positive and negative examples to such a hyperplane are examples 1 and 6. Because we think only two points are closest, we can then find the largest margin hyperplane by finding their perpendicular bisector. The midpoint of the two is $(-1, 1)$, and the slope of the segment connecting the two support vectors is $(2-0)/(0-(-2)) = 1$, so the slope of the perpendicular bisector is -1, i.e. $w_1 = w_2$. We know that $(-1.0, 1.0)$ lies on the bisector; we can use this information to determine $\theta = 0$.

One can then verify that all of the other data points are farther away from the hyperplane than our chosen support vectors. The final step is to determine the exact weight values that optimize the objective; this can be done by solving the following system of equations, and then finding the values of w with minimal l2-norm satisfying the derived inequalities:

$$\begin{cases} -1(0 \times w_1 + 2 \times w_2) \geq 1 \\ +1(-2 \times w_1 + 0 \times w_2) \geq 1 \end{cases}$$

(b) 1. $I = \{1, 6\}$

2. $\alpha = \{\frac{1}{4}, \frac{1}{4}\}$ We can get this easily from $w^* = \alpha_1 y^{(1)} x^{(1)} + \alpha_2 y^{(6)} x^{(6)}$.

3. Objective function value: $\frac{1}{2}\|w^*\|^2 = \frac{1}{4}$.

(c) C determines the tradeoff between the model complexity (here, the norm of $\mathbf{w}$) and the training loss (here, the hinge loss from all examples). When $C = \infty$, we minimize the training loss, and obtain the max-margin hyperplane. In this case, typically, a small number of examples will be support vectors. When C is very small, we minimize the model complexity, and obtain an hyperplane that has a very small norm. In this case, most examples will be support vectors because

i	Label	Hypothesis 1				Hypothesis 2			
		D_0	$f_1 \equiv [x > 2]$	$f_2 \equiv [y > 5]$	$h_1 \equiv [f_1]$	D_1	$f_1 \equiv [x > 10]$	$f_2 \equiv [y > 11]$	$h_2 \equiv [f_2]$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
1	$-$	0.1	$-$	$+$	$-$	0.0625	$-$	$-$	$-$
2	$-$	0.1	$-$	$-$	$-$	0.0625	$-$	$-$	$-$
3	$+$	0.1	$+$	$+$	$+$	0.0625	$-$	$-$	$-$
4	$-$	0.1	$-$	$-$	$-$	0.0625	$-$	$-$	$-$
5	$-$	0.1	$-$	$+$	$-$	0.0625	$-$	$+$	$+$
6	$-$	0.1	$+$	$+$	$+$	0.25	$-$	$-$	$-$
7	$+$	0.1	$+$	$+$	$+$	0.0625	$+$	$-$	$-$
8	$-$	0.1	$-$	$-$	$-$	0.0625	$-$	$-$	$-$
9	$+$	0.1	$-$	$+$	$-$	0.25	$-$	$+$	$+$
10	$+$	0.1	$+$	$+$	$+$	0.0625	$-$	$-$	$-$

Table 1: Table for Boosting results

their hinge loss will be positive. When $C = 0$, we will get $\mathbf{w} = \mathbf{0}$. When C has a moderate value ($C = 1$), we penalize both the model complexity and the training loss. In this case, typically, the number of support vectors will be more than the number of support vectors in the max-margin hyperplane. This is desirable in practice because we want to avoid overfitting.

5. (a) We note that $D_0(i) = 0.1$ for all ten examples. The best learner aligned to the y-axis is $f_1 \equiv [x > \{2\}]$ and the best learner aligned to the x-axis is $f_2 \equiv [y > \{5\}]$. We choose the former as h_1 since

$$\epsilon_{x_1} = [\text{weighted sum of mistakes if } h = x_1] = 2/10$$

$$\epsilon_{x_2} = [\text{weighted sum of mistakes if } h = x_2] = 3/10$$

If base 2 is used, we get the following:

$$\alpha_0 = \frac{1}{2} \log_2 \frac{1-\epsilon}{\epsilon} = \frac{1}{2} \log_2 \frac{0.8}{0.2} = 1$$

If base e is used, we instead get:

$$\alpha_0 = \frac{1}{2} \ln \frac{1-\epsilon}{\epsilon} = \frac{1}{2} \ln \frac{0.8}{0.2} = \ln 2 \approx 0.693$$

(b) Using α_0 (base 2) to compute the new distribution, we get:

$$D_{t+1}(i) = \begin{cases} \frac{1}{Z_0} D_0(i) 2^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ \frac{1}{Z_0} D_0(i) 2^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

$$D_1(i) = \begin{cases} \frac{1}{20 Z_0} & \text{if } h_t(x_i) = y_i \\ \frac{1}{5 Z_0} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

To calculate Z_0,

$$\frac{8}{20Z_0} + \frac{2}{5Z_0} = 1 \Rightarrow Z_0 = 4/5$$

The new distribution D_1 is thus

$$D_1(i) = \begin{cases} 0.0625 & \text{if } h_1(x_i) = y_i \\ 0.25 & \text{if } h_1(x_i) \neq y_i \end{cases}$$

(Note that we get the same final result if base e is used instead)

To choose the next candidate functions we simply observe the data and choose $f_1 \equiv [x > \{10\}]$ and the best learner aligned to the x-axis is $f_2 \equiv [y > \{11\}]$.

$$\epsilon_{f_1} = [\text{weighted sum of mistakes if } h = f_1] = 1 \times 0.25 + 2 \times 0.0625 = 0.385$$
$$\epsilon_{f_2} = [\text{weighted sum of mistakes if } h = f_2] = 0 \times 0.25 + 4 \times 0.0625 = 0.25$$

Clearly f_2 is the winner. Then we have:

$$\alpha_1 = \frac{1}{2} \log_2 \frac{1 - \epsilon_{f_2}}{\epsilon_{f_2}} = \frac{1}{2} \log_2 \frac{0.75}{0.25} = 0.79$$

or, using base e:

$$\alpha_1 = \frac{1}{2} \ln \frac{1 - \epsilon_{f_2}}{\epsilon_{f_2}} = \frac{1}{2} \ln \frac{0.75}{0.25} \approx 0.54$$

(c) See the answer to (b).

(d)

$$H(x) = sgn\left(1 \times [x > 2] + 0.79 \times [y > 11]\right)$$

If we use natural logarithms, the final hypothesis is just a scaled equivalent:

$$H(x) = sgn\left(0.69 \times [x > 2] + 0.54 \times [y > 11]\right)$$

Note: a correct solution to this problem should use the same base for all of the calculations in this problem.