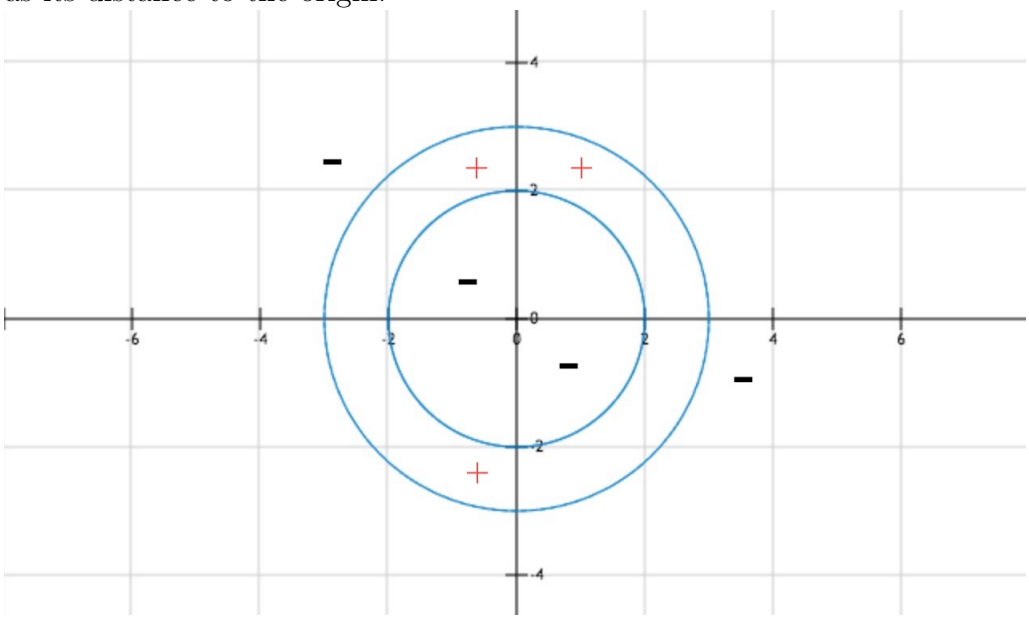
CS446: Machine Learning **Spring 2017**

Problem Set 4

Zhiyu Zhu (zzhu24) *Handed In: March 12, 2017*

1. **[PAC Learning]**

   (a) Instead of doing the two dimensional training, we transfer and record each point as its distance to the origin.



Therefore, we firstly calculate the distances of all the points to the origin. Secondly, we arrange all the point by distance from smallest to the largest and set up a link list that contains all the positive training samples also from smallest distance to largest distance. Thirdly, set rs to the distance of the closest positive training sample and rl to the distance of the furthest positive training sample. However, if the link list is empty meaning that there is no positive sample in the training sample set, then we just fund two point that has largest distance between them and set rs to the distance of the one closer to origin and rl to the distance of the one further from origin.

Here is the pseudo code for the algorithm:

```
def struct point{
    name;
    distance;
    next} point
point * first, end = NULL;
For each positive training sample:
```

```
    if first = end = NULL:
        first = new positive sample;
    end -> next = new positive sample;
    end = new positive sample;
if first = end = NULL:
    (point1, point2) = maximum(difference(distance1, distance2))
    rs = maximum(distance1, distance2);
    rl = minimum(distance1, distance2);
rs = first -> distance;
rl = end -> distance;
```

(b) i. According to the algorithm, we found the tight area bounded by all of the positive sample data. Since the sample set can never have a negative whose distance is between two positive points' distances, all the negative points should all lay outside of the area and therefore means all negative points are correctly labeled. However, it is possible that some positive data will be labeled incorrectly.

ii. For every point, the possibility that a point is correctly labeled and not in area $r_1^* < \|x\|_2 \leq r_1$ or $r_2 < \|x\|_2 \leq r_2^*$ is $(1 - \epsilon$. Since the posibility of correctly labling each point is independent from others, the probability of drawing a sample of m points from D is $(1 - \epsilon)^m$

(c) Since $(1 - \epsilon)^m < \delta$ and according to the hint that $1 - x \leq e^{-x}$, we can conduct that:

$$1 - \epsilon < e^{-\epsilon}$$

$$\text{if } e^{-\epsilon m} < \delta, \text{ then } (1 - \epsilon)^m < \delta$$

$$-\epsilon m < log\delta$$

$$m > \frac{log\delta}{-\epsilon}$$

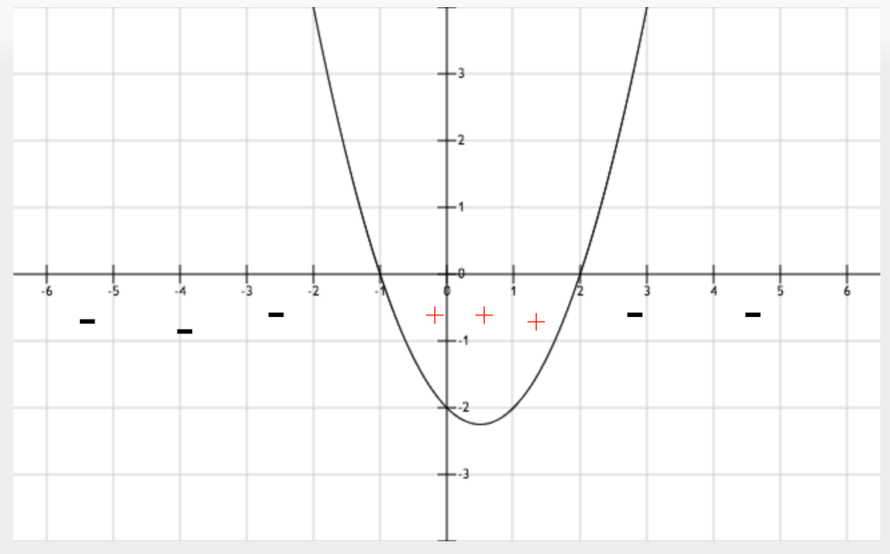
$$m > \frac{1}{\epsilon} log \frac{1}{\delta}$$

(d) Since for three points, there is no function that can guarantee correctly shattering the points, so the VC dimension of H_{2cc} is 2.
According to the bound of the training data set for the infinite hypothesis space:

$$m > \frac{1}{2\epsilon^2}(log|H_{2cc}| + log\frac{1}{\delta})$$

Since in VC, the sample size needed is a little bit larger than that previous approach, previous approach is tighter than VC dimension.

2. [**VC Dimension**] The VC dimension for the hypothesis space is three in this problem. Since the maximum section that can be separated by the function $ax^2 + bx + c$ is three. So if the sample is "+, -, +, -" or "-, +, -, +", then function will not be able to shatter the sample set. However, if the sample is like "+, -, +" or "-, +, -", then the two intersections of the function with the axis should be between positive and negative sample, therefore can be shattered.



3. [**Kernels**]

   (a) Assume w is an initial weight vector for perceptron and x^i be samples:
   the weight vector would be a linear function: $\sum_{1,m} \gamma\alpha_i y_i x^i$ where α is the number of mistakes made on x^i
   $f(x) = sgn(w, x)$
   $= sgn(\sum_{1,m} \gamma\alpha_i y_i (x^i \cdot x))$
   Then rewrite this formula into kernel based:
   $f(x) = sgn(\sum_{z \in M} S(z)K(M, z))$
   M is the set of x^i that the algorithm made mistakes on and need to be tracked.
   $S(z)$ results in $z = \pm\gamma$ depends on the mistakes
   $K(M, z)$ is the dot product for $z \cdot x$

   (b) Since the rules that applying polynomial to a valid kernel, multiplying a valid kernel with a constant c, adding a constant c to a valid kernel and adding two valid kernel all can generate a new valid kernel:

$$K(\vec{\mathbf{x}}, \vec{\mathbf{z}}) = (\vec{\mathbf{x}}^T\vec{\mathbf{z}})^3 + 49(\vec{\mathbf{x}}^T\vec{\mathbf{z}} + 4)^2 + 64\vec{\mathbf{x}}^T\vec{\mathbf{z}}$$
$$K(\vec{\mathbf{x}}, \vec{\mathbf{z}}) = (\vec{\mathbf{x}}^T\vec{\mathbf{z}})^3 + 49(\vec{\mathbf{x}}^T\vec{\mathbf{z}})^2 + 392(\vec{\mathbf{x}}^T\vec{\mathbf{z}}) + 784 + 64\vec{\mathbf{x}}^T\vec{\mathbf{z}}$$
$$K(\vec{\mathbf{x}}, \vec{\mathbf{z}}) = (\vec{\mathbf{x}}^T\vec{\mathbf{z}})^3 + 49(\vec{\mathbf{x}}^T\vec{\mathbf{z}})^2 + 456\vec{\mathbf{x}}^T\vec{\mathbf{z}} + 784$$

applying all the rules, this is a valid kernel.

(c) According to the problem: $K(\mathbf{x}, \mathbf{z}) = \sum_{c \in C} c(\mathbf{x})c(\mathbf{z}) = C_{k,m}$
if the total number of sample that both in the original and the monotone conjunction is smaller than k, then the $K(\mathbf{x}, \mathbf{z})$ would be zero.
Else, we need to find all the positive features in the k-conjunction that are positive in both x and z, the searching time should be O(n) and the multiplication time is also O(n). So $K(\mathbf{x}, \mathbf{z})$ computation should be linear time.

4. **[SVM]**

(a) 1. Define $\mathbf{w} =$ $[-1, 0]$
Define $\theta =$ -0.3

2. Define $\mathbf{w} =$ $[-0.5, -0.5]$
Define $\theta =$ 0

3. In order to optimize into the hard SVM formation, we need to make the margin as large as possible to reduce further mistakes. So, we need to find the SVM formation that is furthest to the two closet positive point and negative point. To find the SVM formation, just compare the perpendicular distance. From all of those bisection, find the SVM formation that results in the largest margin to finish the optimization.

(b) 1. Define $I =$ $[0, 2]$, $[-2, 0]$

2. Define $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_{|I|}\} =$ $0.25, -0.25$

3. *Objective function value* = $\frac{1}{2}\|w\|^2 = 0.25$

(c) For $C = 0$, the solution to optimization problem gives the hyperplane that you have found in (a)-2.
Since we know that C contribute for hinge loss and margin:
When $C = \infty$, the optimization gives the tightest SVM formation and tries to both minimize the hinge loss and the margin. In this way, the formation is most closely bounded by the training data.
When $C = 1$, the optimization balances the hinge loss and margin.
When $C = 0$, the optimization only maximize the margin of the SVM formation. Since we can choose any slack variable, the objective function is meaningless.

5. **[Boosting]**

i	Label	Hypothesis 1				Hypothesis 2			
		D_0	$f_1 \equiv [x > 2]$	$f_2 \equiv [y > 6]$	$h_1 \equiv [x > 2]$	D_1	$f_1 \equiv [x > 9]$	$f_2 \equiv [y > 6]$	$h_2 \equiv [y > 6]$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
1	$-$	$\frac{1}{10}$	-	$+$	-	$\frac{1}{16}$	-	$+$	$+$
2	$-$	$\frac{1}{10}$	-	-	-	$\frac{1}{16}$	-	-	-
3	$+$	$\frac{1}{10}$	$+$	$+$	$+$	$\frac{1}{16}$	-	$+$	$+$
4	$-$	$\frac{1}{10}$	-	-	-	$\frac{1}{16}$	-	-	-
5	$-$	$\frac{1}{10}$	-	$+$	-	$\frac{1}{16}$	-	$+$	$+$
6	$+$	$\frac{1}{10}$	$+$	$+$	$+$	$\frac{1}{4}$	-	$+$	$+$
7	$+$	$\frac{1}{10}$	$+$	$+$	$+$	$\frac{1}{16}$	$+$	$+$	$+$
8	$-$	$\frac{1}{10}$	-	-	-	$\frac{1}{16}$	-	-	-
9	$+$	$\frac{1}{10}$	-	$+$	-	$\frac{1}{4}$	-	$+$	$+$
10	$-$	$\frac{1}{10}$	$+$	$+$	$+$	$\frac{1}{16}$	-	$+$	$+$

Table 1: Table for Boosting results

(a) Answer filled in above

(b) The best function for D_0 is $h_1 \equiv x > 2$

(c) The best function for D_1 is $h_2 \equiv y > 6$

The mistakes made buy hypothesis 1 is 3, so $\epsilon_0 = \frac{2}{10}$
$\alpha_0 = \frac{1}{2} log(\frac{1-\epsilon_0}{\epsilon_0}) = log(2)$
Since the two mistake get demoted and eight correct ones get promoted:
$z_0 = \sum_i D_0^i e^{-\alpha_0 y_i h_0(x^i)}$ and $D_0 = \frac{1}{10}$
$e^{+\alpha_0} = \frac{1}{2}$
$e^{-\alpha_0} = 2$
$z_0 = 2(2 \cdot \frac{1}{10}) + 8(\frac{1}{2} \cdot \frac{1}{10})$
$z_0 = 0.8$

For the second hypothesis h_2:
$D_1^+ = \frac{D_0 \cdot e^{+\alpha_0}}{z_0} = \frac{1}{16}$
$D_1^- = \frac{D_0 \cdot e^{-\alpha_0}}{z_0} = \frac{1}{4}$
Since the second weak learner made 4 mistakes:
$\epsilon_1 = \frac{4}{16}$
$\alpha_0 = \frac{1}{2} log(\frac{1-\epsilon_0}{\epsilon_0}) = \frac{1}{2} log(3)$

(d) According to the choice of weak learners and the calculation of alpha:
$Hypothesis_{final} = sgn(log(2) \cdot sgn(x-2) + \frac{1}{2} log(3) \cdot sgn(y-6))$