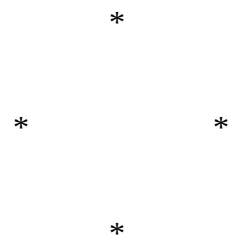
Machine Learning: 10701 and 15781, 2003
Assignment 4

1. VC Dimension (30 Points)

Consider the space of instance X corresponding to all points in the 2D x, y plane. Give the VC dimension of the following hypothesis spaces. No explanation required.

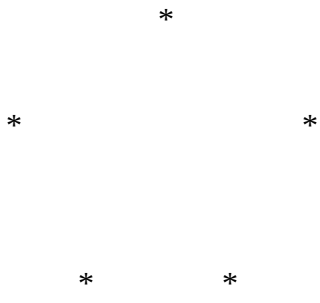
(a) H_r = the set of all axes-parallel rectangles in the x, y plane. That is, $H_r = \{((a < x < b) \wedge (c < y < d)) \mid a, b, c, d \in \mathbf{R}\}$. Points inside the rectangle are classified as positive.

Answer: 4

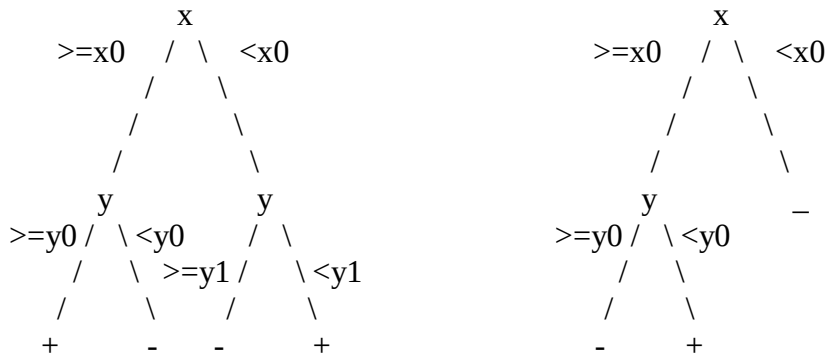


(b) H_a = like (a), but including all rectangles (not just the ones parallel to the axes of coordinate system).

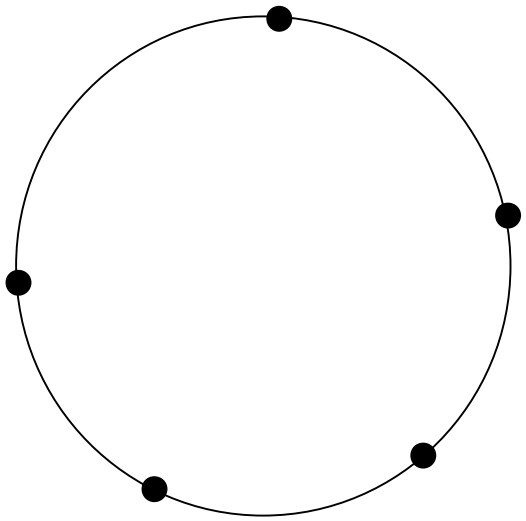
Answer: at least 5



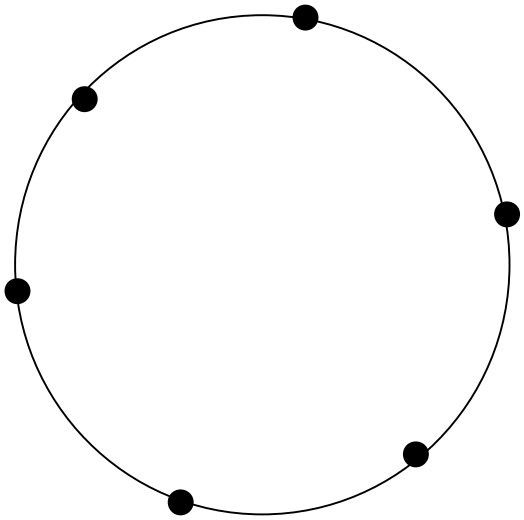
(c) H_d = real-valued, depth-2 decision trees. For example, the following trees are in H_d.



Answer 1: at least 5 (assume attributes used in 1st and 2nd splits must be different)



Answer 2: at least 6 (otherwise)



(e) (**Zero Points**) H_f = hypotheses with the form $f(x) = \theta(\sin(\alpha \cdot x))$, $x, \alpha \in \mathbf{R}$ where $\theta(z) = 1$ iff $z > 0$ and 0 otherwise. You can consider this question in 1D space. (**Zero Points: Only Do This For Fun If You Would Like To And If You Have Time**.)

Answer: infinite (read Burges' tutorial for detail)

2. Support Vector Machine (45 Points)

The following question requires you to use **MatLab**, but it has been designed to be just as easy for a MatLab novice as for a MatLab expert. Please read the appendix for how to use MatLab.

We will investigate Support Vector Machine with two toy datasets. The file "1d.clean" contains 100 examples each of which has a real-valued input attribute x and a class label y. The data set is generated in the following way:

$x \sim p(x)$ where $p(x) = 0.5$ iff $-1 \le x \le 1$, otherwise $p(x) = 0$.

$$y(x) = \begin{cases} 1 & x \ge 0 \\ -1 & x < 0 \end{cases}$$

In addition, we add noise to this dataset by negating the class of some examples that produces the noisy dataset "1d.noise".

Training a SVM involves setting up a convex quadratic optimization problem and solving it. MatLab is a common mathematical programming language that facilitates this by providing quadratic programming functions, **qp** or **quadprog**. A MatLab program, "svm.m", has been prepared for you that trains the SVM on each of the datasets and outputs results including margin width, training error, number of support vectors and number of misclassifications.

In this assignment, you are asked to investigate the impact of the trade-off weight C on margin width and training error. Considering the object function for non-separable case:

$$L = \tfrac{1}{2}\|\mathbf{w}\|^2 + C(\sum_i \xi_i)$$

the margin width is defined as: Margin = $2/\|\mathbf{w}\|$,

and the training set error is defined as: Error = $\sum_i \xi_i$.

(a) Read the program and complete the setting up of Hessian matrix H, and the lower and upper bounds for Lagrange multiplier α_i (Alpha). Note that the Hessian matrix H is exactly the Q matrix in our slides.

$H(i, j) = X(i)X(j)Y(i)Y(j)$
$LB = zeros(nsample,1)$
$UB = C(n) * ones(nsample,1)$

(b) Measure the impact of trade-off weight C on the margin width and training error with the clean dataset "1d.clean". Turn in plots, showing how margin width and training error vary with C, including the values 0.01, 0.1, 1, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000 and inf.

(c) Measure the impact of trade-off weight C on the margin width and training error with the noise dataset "1d.noise". Turn in plots, showing how margin width and training error vary with C, including the values 0.01, 0.1, 1, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000 and inf.

(d) Briefly explain your findings.

Some hints:

(1) when C is small (C = 0.01, 0.1, 1, 5)

because $w = \sum_i \alpha_i x_i y_i$ **and** $0 \le \alpha \le C$ **===>** w **is small ===> large margin (**$2/|w|$**).**

Please note that the margin value calculated from the formula $2/|w|$ **may be wrong when C is small.**

(2) when C gets too large (C = inf)

The increase of margin and training error for noisy data when C = inf is probably due to numerical instability. Moreover, setting C to inf is equivalent to assume the dataset is separable which isn't true for the noisy data.

3. K Nearest Neighbor in Regression (25 Points)

Suppose we have a real-valued training dataset $\{(x_i, y_i) \mid 1 \le i \le M, x_i \in \mathbf{R}, y_i \in \mathbf{R}\}$ which is generated using the following distribution:

$y_i \sim N(c, \sigma^2)$ where c is unknown to us. (Note that we assume the variance σ^2 is known.)

$x_i \sim P(x)$ where $P(x) = 1$ for $0 \le x \le 1$, otherwise $P(x) = 0$.

Our task is to compare the performance of the following two regression algorithms.

Alg1: Use Maximum Likelihood Estimation (MLE) to learn c from the dataset. The MLE assumption results in $\hat{c} = \frac{1}{M}\sum_{i=1}^{M} y_i$. For any input x, the output is simply $\hat{y}(x) = \hat{c}$.

Alg2: Use 1- Nearest Neighbor to predict y. Namely, $\hat{y}(x) = y_i$, where y_i is the output value associated with the training set datapoint x_i that is the nearest neighbor of x. If there is a tie among multiple training datapoints for being the nearest neighbor of x, then we just randomly select one of them.

(a) Assume $M \to \infty$. What is the expected squared error of Alg1 and Alg2 on the training set?

For Alg1, $\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M} (\hat{y}(x_i) - y_i)^2 = ?$

For Alg2, $\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}(\hat{y}(x_i)-y_i)^2 = ?$

For Alg 1:

Answer:

$$\lim_{M\to\infty} \hat{c} = \lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M} y_i = c$$

$$\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}(\hat{y}(x_i)-y_i)^2 = \lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}(c-y_i)^2 = \sigma^2$$

For Alg 2:

Answer 1: Because x is real-valued continuous variable, so theoretically $P(x_i = x_j) = 0$ for any two training examples (x_i, y_i) and (x_j, y_j). Namely, no two training examples have the same x value. Therefore,

$$\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}(\hat{y}(x_i)-y_i)^2 = \lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}(y_i-y_i)^2 = 0$$

Answer 2: When we use a computer to generate x_i, we will find that x_i becomes discrete due to the accuracy loss of the computer. Moreover, there will be infinite number of training examples with the same x value when $M \to \infty$. In this case,

$$\lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}(\hat{y}(x_i)-y_i)^2 = \lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i - c + c - y_i)^2$$

$$= \lim_{M\to\infty} \frac{1}{M}\sum_{i=1}^{M}[(\hat{y}_i - c)^2 + (y_i - c)^2 - 2(\hat{y}_i - c)(y_i - c)]$$

$$= 2\sigma^2$$

(b) Assume $M \to \infty$. What is the expected squared error of Alg1 and Alg2 for predicting the output y of a future data point (x, y) generated in the same way as training data.

For Alg1, $E((\hat{y}(x)-y)^2) = ?$

For Alg2, $E((\hat{y}(x)-y)^2) = ?$

For Alg 1:

Answer:

$$E((\hat{y}(x)-y)^2) = E(c-y)^2 = \sigma^2$$

For Alg 2:

Answer:

$$E((\hat{y}(x)-y)^2) = \iint_{\hat{y},y} (\hat{y}-y)^2 P(\hat{y},y)d\hat{y}dy = \iint_{\hat{y},y} (\hat{y}-c+c-y)^2 P(\hat{y})P(y)d\hat{y}dy$$

$$= \int_y P(y)dy \int_{\hat{y}} (\hat{y}-c)^2 P(\hat{y})d\hat{y} + \int_{\hat{y}} P(y)d\hat{y} \int_y (y-c)^2 P(y)dy - 2\int_{\hat{y}} (\hat{y}-c)P(\hat{y})d\hat{y} \int_y (y-c)P(y)dy$$

$$= 2\sigma^2$$