EE6550 Machine Learning, Spring 2016

Homework Assignment #1

Please turn in your solutions and programs (electronic source code, printed report and user manual) to EECS 608 by 6:00 pm on April 6th Wednesday (after the spring break). You are encouraged to consult or collaborate with other students while solving the problems but you will have to turn in your own solutions and programs with your own words. Copying will not be tolerated. If you consult or collaborate, you must indicate all of your consultants or collaborators with their contributions.

Part I: Word problem set.

1. Problem 2.7 of Chapter 2 of the textbook.
2. Problem 2.9 of Chapter 2 of the textbook.
3. Problem 3.9 of Chapter 3 of the textbook.
4. Problem 3.12 of Chapter 3 of the textbook.
5. Problem 3.21 of Chapter 3 of the textbook.

Part II: Programming problem: Implementation of a consistent PAC-learning algorithm $\mathbb{A}$ for the concept class $\mathcal{C}$ of all axis-aligned rectangular areas in the plane.

Input: a random sample $S = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ of size m drawn i.i.d. according to a fixed but unknown probability distribution P over the input space $\mathbb{R}^2$ with labels $(c(\mathbf{x}_1), c(\mathbf{x}_2), \ldots, c(\mathbf{x}_m))$, where c is a fixed but unknown concept.

- We assume that the "unknown" probability distribution P is a bivariate normal distribution over $\mathbb{R}^2$ with a pdf

$$\begin{aligned} f_{XY}(x,y) &= \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-r_{X,Y}^2}} \exp\left\{-\frac{1}{2(1-r_{X,Y}^2)}\left(\frac{(x-\mu_X)^2}{\sigma_X^2}\right.\right. \\ &\qquad \left.\left. -2r_{X,Y}\frac{(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y} + \frac{(y-\mu_Y)^2}{\sigma_Y^2}\right)\right\}, \end{aligned}$$

  where μ_X, σ_X^2 and μ_Y, σ_Y^2 are mean and variance of x-coordinate and y-coordinate respectively and $r_{X,Y} = E[(X-\mu_X)(Y-\mu_Y)]/(\sigma_X\sigma_Y)$ is the correlation coefficient of x-coordinate and y-coordinate.

- You have to use the mean vector $MU = [\mu_X \ \ \mu_Y]$ and the covariance matrix $SIGMA = [\sigma_X^2 \ \ r_{XY}\sigma_X\sigma_Y; r_{XY}\sigma_X\sigma_Y \ \ \sigma_Y^2]$ as parameters to specify a bivariate normal distribution P.

- You may use the function mvnrnd$(MU, SIGMA, m)$ in Matlab to generate a sample of bivariate normal random vector of size m, where $MU = [\mu_X \ \ \mu_Y]$ is the mean vector and $SIGMA = [\sigma_X^2 \ \ r_{XY}\sigma_X\sigma_Y; r_{XY}\sigma_X\sigma_Y \ \ \sigma_Y^2]$ is the covariance matrix.

- To represent a fixed but unknown concept c, we use the pair of the lower left corner point $\mathbf{v}$ and the upper right corner point $\mathbf{u}$ of the axis-aligned rectangular area, i.e., $c = [\mathbf{v} \;\; \mathbf{u}]$. You should be able to select an "unknown" concept c in one of the two ways: either by direct specification or by random selection.
- You must select an "unknown" concept c such that $P(c) \geq 2\epsilon$, where ϵ is the upper bound of generalization error guarantee we will use. It is usually difficult, if not impossible, to compute $P(c)$ for given P and c. We resort to an estimation of $P(c)$ as follows. Let $\tilde{S} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$ be a sample of size N drawn i.i.d. according to P. Then $c(\mathbf{y}_i)$ is 1 if $\mathbf{y}_i \in c$ and 0 if $\mathbf{y}_i \notin c$. Let $\hat{p} = \frac{1}{N}\sum_{i=1}^{N} c(\mathbf{y}_i)$. It can be seen that $\hat{p}$ is an unbiased estimator of $P(c)$, i.e., $E[\hat{p}] = P(c)$. The variance σ^2 of the estimator $\hat{p}$ is $P(c)(1 - P(c))/N$ which will be very small for large N. Since $0 < P(c) < 1$, it is clear that $\sigma^2 \leq 1/(4N)$. By the central limit theorem, the distribution of $(\hat{p} - P(c))/\sigma$ is well approximated by the standard normal distribution for large enough N. Since $P((\hat{p} - P(c))/\sigma \leq 3.719) \simeq 0.9999$, with probability well approximated by 0.9999, we have $(\hat{p} - P(c))/\sigma \leq 3.719$, i.e., $P(c) \geq \hat{p} - 3.719\sigma$. Since $\sigma \leq 1/(2\sqrt{N})$, with probability at least 0.9999, $P(c) \geq \hat{p} - 1.8595/\sqrt{N}$. Let N_ϵ be an integer such that $1.8595/\sqrt{N_\epsilon} \leq \epsilon$. Then given a sample $\tilde{S}$ of size $N_\epsilon = \lceil (1.8595/\epsilon)^2 \rceil$ drawn i.i.d. according to P, with a probability at least 0.9999, we have $P(c) \geq \hat{p} - \epsilon$. Thus with a confidence at least 0.9999, the probability $P(c)$ of a selected "unknown" concept c will be no less than 2ϵ if the empirical probability $\hat{p} = \frac{1}{N_\epsilon}\sum_{i=1}^{N_\epsilon} c(\mathbf{y}_i)$ is no less than 3ϵ.

Output: a hypothesis $h_S = \mathbb{A}(S; c, \mathcal{H})$, where $\mathcal{H}$ is the hypothesis set which is equal to the concept class $\mathcal{C}$ in this problem.

Generalization guarantee: Given a labeled sample S of size $m = \lceil \frac{4}{\epsilon} \ln \frac{4}{\delta} \rceil$, with probability at least $1 - \delta$, the generalization error $R(h_S)$ of the output h_S is upper bounded by ϵ. Please verify this guarantee of your PAC-learning algorithm $\mathbb{A}$ by running algorithm $\mathbb{A}$ for $\lceil 10/\delta \rceil$ times and showing that at most 10 out of the $\lceil 10/\delta \rceil$ h_S have $R(h_S) > \epsilon$. Similar to the computation of $P(c)$, it is usually difficult, if not impossible, to compute $R(h_S)$ for given h_S, P and c. We resort to an estimation of $R(h_S)$ as we did for $P(c)$ in above. Let $\Delta_S = c \setminus h_S$ be the error region. Then $R(h_S) = P(\Delta_S)$. Let $\breve{S} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M)$ be a sample of size M drawn i.i.d. according to P. Then $\Delta_S(\mathbf{z}_i)$ is 1 if $\mathbf{z}_i \in \Delta_S$ and 0 if $\mathbf{z}_i \notin \Delta_S$. Let $\hat{q} = \frac{1}{M}\sum_{i=1}^{M} \Delta_S(\mathbf{z}_i)$. It can be seen that $\hat{q}$ is an unbiased estimator of $P(\Delta_S)$, i.e., $E[\hat{q}] = P(\Delta_S)$. The variance ${\sigma'}^2$ of the estimator $\hat{q}$ is $P(\Delta_S)(1 - P(\Delta_S))/M$ which will be very small for large M. Since $0 < P(\Delta_S) < 1$, it is clear that ${\sigma'}^2 \leq 1/(4M)$. By the central limit theorem, the distribution of $(\hat{q} - P(\Delta_S))/\sigma'$ is well approximated by the standard normal distribution for large enough M. Since $P(|(\hat{q} - P(\Delta_S))/\sigma'| \leq 3.8906) \simeq 0.9999$, with probability well approximated by 0.9999, we have $|(\hat{q} - P(\Delta_S))/\sigma'| \leq 3.8906$, i.e., $\hat{q} - 3.8906\sigma' \leq P(\Delta_S) \leq \hat{q} + 3.8906\sigma'$. Since $\sigma' \leq 1/(2\sqrt{M})$, with probability at least 0.9999, $\hat{q} - 1.9453/\sqrt{M} \leq P(\Delta_S) \leq \hat{q} + 1.9453/\sqrt{M}$. Let M_ϵ be an integer such that $1.9453/\sqrt{M_\epsilon} \leq \epsilon/10$. Then given a sample S of size $M_\epsilon = \lceil (19.453/\epsilon)^2 \rceil$ drawn i.i.d. according to P, with a probability at least 0.9999, we have $\hat{q} - \epsilon/10 \leq P(\Delta_S) \leq \hat{q} + \epsilon/10$. Thus with a confidence at least 0.9999, the generalization error $R(h_S) = P(\Delta_S)$ of the output hypothesis h_S is equal to the empirical probability $\hat{q}$ within $\pm 0.1\epsilon$ error.

What to submit? You should submit the following items:

1. The electronic source code of your PAC-learning algorithm. (It is recommended for you to use MatLab to write your programs, although C, C++ or Python are acceptable. If you prefer to use C, C++ or Python, please send me a notice by email by 03/29 and please have your code compilable by a standard compiler.) I will email you where to upload the source code.
2. A printed report consisting of at least:
   (a) A chosen "unknown" bivariate normal distribution P whose correlation coefficient r_{XY} has $0.3 \leq |r_{XY}| \leq 0.7$.
   (b) A randomly chosen "unknown" concept c together with a verification of $P(c) \geq 2\epsilon$.
   (c) An output hypothesis $h_S = \mathbb{A}(S; c, \mathcal{H})$ for a labeled sample S of size $m = \lceil \frac{4}{\epsilon} \ln \frac{4}{\delta} \rceil$, for each of the two cases: (1) $\delta = 0.01$ and $\epsilon = 0.1$; (2) $\delta = 0.01$ and $\epsilon = 0.01$, together with an estimation of generalization error $R(h_S)$ within $\pm 0.1\epsilon$ error.
   (d) Verification of generalization guarantee of your PAC-learning algorithm $\mathbb{A}$ for each of the two cases: (1) $\delta = 0.01$ and $\epsilon = 0.1$; (2) $\delta = 0.01$ and $\epsilon = 0.01$.
3. A user manual which should include instructions of
   (a) how to compile the source code;
   (b) how to run, i.e., execute the PAC-learning algorithm, including the required formats of input parameters or input file which contains all parameters needed;
   (c) what results are reported.

   These instructions should support the test scenario in the grading session.

Test scenario in the grading session: the grader will test your algorithm with your source code by

1. inputting generalization guarantee parameters δ and ϵ;
2. inputting a set of parameters MU and $SIGMA$ to specify an "unknown" bivariate normal distribution P;
3. inputting an "unknown" concept c till the requirement $P(c) \geq 2\epsilon$ is met;
4. seeing the output $h_S = \mathbb{A}(S; c, \mathcal{H})$ together with an estimated generalization error $R(h_S)$;
5. checking the estimated generalization error $R(h_S)$ by using a prepared program;
6. seeing the verification of (δ, ϵ) generalization guarantee of your PAC-learning algorithm $\mathbb{A}$;
7. evaluating the validity of this verification.