**EE6550 MACHINE LEARNING**

**HW#1, REPORT OF MY PROGRAM**

102061210 王尊玄

1. Objective:

Implement a consistent PAC-learning algorithm A for the concept class C of all axis-aligned rectangular areas in the plane. Input is given as a random sample of size m drawn i.i.d. according to a fixed but unknown probability distribution P over the input space with labels , where c is a fixed but unknown concept. Output is a hypothesis , approximating unknown concept c. Let R be generalization error, our goal:

1. Method:

* Probability distribution P: We assume that the unknown probability distribution to draw sample is a bivariate normal distribution over input space with its pdf,

, where and are mean and variance of x-coordinate and y-coordinate respectively and is the correlation coefficient of x-coordinate and y-coordinate. Parameters that can be adjusted are specified,

* Concept c: Unknown concept c is selected such that , where is the upper bound of generalization error guarantee. As it is not realistic to compute , we use as an estimator of . By central limit theorem, we get with a probability at least 0.9999, we have . Thus, if we want to make , then we find c satisfying .
* PAC-learning algorithm A:

Algorithm A will generate a hypothesis that encircles all positive points (), and itself is an axis-aligned rectangle in . Algorithm A will search through all positive points for smallest and largest value of x-coordinate and y-coordinate respectively.

* Generalization error :

Given a labeled sample of size , we can get . Since directly compute is impractical, we use as an estimator of . By central limit theorem, we get with a probability at least 0.9999, we have

Thus, we can assert that if .

Also, if we want to make sure that with a probability at least , is true, then we run algorithm A times to find a unknown but fixed concept c and show that no more than 10 out of defies .

1. Result:

* Input:

|  |  |  |  |
| --- | --- | --- | --- |
| MU |  |  |  |
| [5 15] | 0.4 | 0.1 | 0.15 |
|  |  |  | |
| 0.1/0.01 | 0.01 |  | |

* Output:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  | 0.0338 | 0.0021 |

|  |  |
| --- | --- |
| 2694 | # of s.t. |
|  | {2, 6, 9, 4, 7, 5, 3, 5, 4, 6} |
|  | {3, 2, 2, 1, 5, 1, 1, 0, 2, 3} |
|  | {0, 2, 0, 1, 2, 0, 2, 0, 3, 1} |
|  | {0, 0, 0, 1, 0, 1, 0, 0, 0, 0} |

I took 10 trials over times algorithm A, and none of them have more than 10 such that . Also, I find that if we set smaller, we can have higher confidence over , i.e. , where x may be bigger than .