Lecture 4: Selected Topics (1) — PAC Learning Framework

In this and the next lectures, we will focus on PAC learning and some algorithmic techniques developed in PAC learning.

> **An Important Message from O.W:** I choose these topics for illustrating the importance of a formal framework and also some new algorithmic approach that may be useful in many areas.

PAC learning framework was introduced by L. Valiant in his seminal 1984 *CACM* paper [3], which has been an important framework for designing/analyzing learning algorithms. A research area investigating (efficient) PAC learnability is now called as *PAC Learning Theory*, which produces many basic/important results for *Machine Learning*.

1 Basic Definitions for PAC Learning

A concept: A learning target.

In this lecture we may consider each concept is a Boolean[1] function f, a function mapping $\{+1,-1\}^n$ to $\{+1,-1\}$. Each element $\boldsymbol{x} = (x_1, \ldots, x_n)$ of $\{+1,-1\}^n$ is called an *input instance* and value $f(\boldsymbol{x})$ is called a *label*. Note that each x_i is considered as a value of the corresponding *attribute*.

A *target concept* is a concept that we want to learn. On the other hand, a *hypothesis* is a concept that we would produce (or a learning algorithm yields) as an approximation of the target.

A sample: A set of examples given for our learning task.

An *example* is a pair of an input instance and its label. A *sample* is the set of examples.

Example distribution: A probability distribution that each example is given.

For simplicity, we consider the case where a target concept is fixed to some function f. Thus, the probability of some $(\boldsymbol{x}, y)$ occurs as an example is determined that the probability that the instance $\boldsymbol{x}$ occurs (because then y is determined as $y = f(\boldsymbol{x})$). We denote this probability as $D_n(\boldsymbol{x})$ or $D(\boldsymbol{x})$.

PAC: *P*robabilistically and *A*pproximately *C*orrectness. A criterion of our learning task. Below we define this criterion step by step. Here we fix a target concept and a sample distribution; let f_* and D_* to denote them respectively.
(1) For any hypothesis h, its *error probability* is defined by

$$\mathrm{err}(h) \;=\; D_*\big(\, f_*(\boldsymbol{x}) \neq h(\boldsymbol{x})\,\big),$$

[1] Usually by "Boolean" we mean 0 or 1 value; but in Machine Learning we often use +1 for 1 and −1 for 0, and we follow this convention here.

where $D_*(\cdots)$ is the abbreviation of $D_*(\{\boldsymbol{x} : \cdots\})$, that is, the probability that some instance $\boldsymbol{x}$ satisfying $\cdots$ is given (as an example).

We may also use an expression such as $Pr_{\boldsymbol{x}:D_*}[\Phi(\boldsymbol{x})]$ to denote the probability that $\Phi(\boldsymbol{x})$ holds when $\boldsymbol{x}$ is given under the distribution D_*. Note that $Pr_{\boldsymbol{x}:D_*}[\Phi(\boldsymbol{x})] = D_*(\Phi(\boldsymbol{x}))$. Thus, a simpler notation $D_*(\Phi(\boldsymbol{x}))$ is usually used. On the other hand, for any $m \geq 1$, we use $Pr_{S:D_*^m}[\cdots]$ for

$$Pr_{S:D_*^m}\big[\,\Phi(S)\,\big] \;=\; \begin{array}{l}\text{the probability that } \Phi(S) \text{ holds when a sample } S\\ \text{of size } m \text{ given under the distribution } D_*,\end{array}$$

by which we can express more clearly that the probability is on the choice of S.
(2) We are given three parameters: n, ϵ, and δ, where $n \geq 1$ and $0 < \epsilon, \delta < 1$. n is the number of attributes and it is usually regarded as a *size parameter.* ϵ is called an *error bound* (or an *approximation parameter*) and δ is called a *reliability bound.* Parameters ϵ and δ are called *learning parameters.*
(3) The PAC goal for a given target concept f_* is defined as follows.

$$\begin{array}{l}\forall \epsilon, \delta,\ 0 < \epsilon, \delta < 1,\\ \exists m \geq 0,\\ \forall D_* \text{ (distribution over } \{+1,-1\}^n)\\ \Pr\limits_{S:D_*^m}\left[\begin{array}{c}\text{we can obtain } h \text{ from } S \text{ satisfying}\\ (*)\quad \Pr\limits_{\boldsymbol{x}:D_*}[\ f_*(\boldsymbol{x}) \neq h(\boldsymbol{x})\ ] \;\leq\; \epsilon\end{array}\right] \;\geq\; 1-\delta.\end{array}$$

A hypothesis h satisfying $(*)$ is called an *ϵ-approximation* (of f_*).
(4) As discussed in the previous lecture, algorithms should be designed to handle infinite number of instances, in particular, with increasing size. For PAC learning algorithm, we also consider target concepts with increasing size parameter n.

> **An Important Message from O.W:** Note here that the size parameter is n, the number of attributes. The sample size m is not considered as size because m can be determined by algorithms. Thus, m should be regarded as some efficiency measure like time complexity.

We need to give some restrictions to our target concepts. A *concept class* is a set of concepts satisfying a certain set of conditions; usually, we define a concept class in terms of a way to describe Boolean functions that belong to the class.

For a given target concept class $\mathcal{C}$, an algorithm A is called a *PAC-learning algorithm* for $\mathcal{C}$ if it satisfies the following condition:

$$\begin{array}{l}\forall \epsilon, \delta,\ 0 < \epsilon, \delta < 1,\quad \forall n \geq 1,\\ \exists m \geq 0 \text{ (which is determined by } A \text{ from } \epsilon, \delta, n),\\ \forall D_* \text{ (distribution over } \{+1,-1\}^n),\quad \forall f_* \in \mathcal{C}\\ \Pr\limits_{S:D_*^m}\left[\begin{array}{c}A \text{ given } S \text{ yields some } h \text{ satisfying}\\ (*)\quad \Pr\limits_{\boldsymbol{x}:D_*}[\ f_*(\boldsymbol{x}) \neq h(\boldsymbol{x})\ ] \;\leq\; \epsilon\end{array}\right] \;\geq\; 1-\delta.\end{array}$$

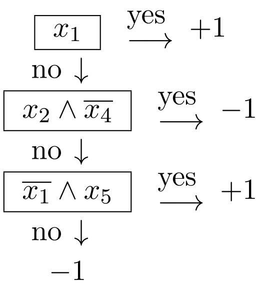
Furthermore, if A's time complexity is polynomially bounded w.r.t. n, we say that A is a *polynomoal-time PAC learning algorithm* (for class $\mathcal{C}$).

2 Example and Key Result on PAC Learning

Example 1. (DL: Decision List)

A *decision list* is a way to describe a Boolean function as illustrated by the right figure. In general, a decision list is a sequence of pairs of *Boolean term* and a label expressed as

$$(\ (t_1, b_1), (t_2, b_2), ..., (t_d, b_d), b\ ).$$



For example, the decision list of the right figure is expressed as

$$(\ (x_1, +1), (x_2 \wedge \overline{x_4}, -1), (\overline{x_1} \wedge x_5, +1), -1\ )$$

Each term is called a *branch*.

Here by "Boolean term" we mean the conjunction of literals such as $X_1 \wedge \overline{X_2} \wedge X_5$. A decision list is called an *k-decision list* if each branch is defined by a Boolean term with at most k literals. Let k-DL denote the class of Boolean functions expressed by k-decision lists. Though we omit stating here, there is a simple polynomial-time PAC learning algorithm for k-DL for any fixed k. (In the class I would explain an algorithm for 2-DL if I have time.) □

Now we state one of the key theorems established in the early stage of PAC Learning Theory [1].

Theorem 1. *(PAC learning is achieved by "Occam Razor")*
For any concept class $\mathcal{C}$, consider any algorithm L that yields a hypothesis consistent with a given sample. Let $\mathcal{H}_{n,m}$ be a class of hypotheses (i.e., Boolean functions) that algorithm L may yield on some sample of size m on some target concept in $\mathcal{C}$ of size n. (Note that m is determined by algorithm L from ϵ, δ, n.)

Let $M(n, m)$ denote the number of hypotheses of $\mathcal{H}_{n,m}$. For any learning parameters ϵ, δ, and for any n, if we can design the algorithm so that

$$m \ \geq\ \frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{\ln M(n, m)}{\epsilon}$$

holds, then L can be used as a PAC-learning algorithm for $\mathcal{C}$.

Note that $M(n, m)$ is related to the length of descriptions in $\mathcal{H}_{n,m}$. For example, $M(n, m)$ is trivially bounded by $2^{\ell(n,m)}$, where $\ell(n, m)$ is the bit length of the largest hypothesis of $\mathcal{H}_{n,m}$. In this sense, it would be better to use hypotheses of smaller length (and consistent with a given sample). This is why the above theorem is called *Occam's Razor*.

References

[1] A. Blumer, A Ehrenfeucht, D. Haussler, and M.K. Warmuth, Occaum's razor, *Information Processing Letters*, 24:377–380, 1987.

[2] M. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory*, The MIT Press, 1994.

[3] L. Valiant, A theory of the learnable, *Communications of ACM*, 27(11):1134–1142, 1984.

[4] 金森敬文，畑埜晃平，渡辺治，ブースティング，森北出版，2006. **Sorry!! This is in Japanese.**

Homework assignment from this lecture

Solve one of the following problems. (More problems will be given next week.)

Q3.1. It is not so difficult to prove Theorem 1, so why don't you prove it (without reading the referenced paper)! You can go back to the definition and consider the probability that one fixed hypothesis h is not an ϵ-approximation of a given target f_* even though h is consistent with f_* on m examples of S. (What is the randomness here for discussing the probability?) Then we can use the union bound to estimate the probability that this situation occurs on some hypothesis of $\mathcal{H}_{n,m}$.