PAC-Learning Unambiguous NTS Languages

Franco M. Luque

Grupo de Procesamiento de Lenguaje Natural
Fa.M.A.F., Universidad Nacional de Córdoba

CONICET

July 22 2009

Outline of the Talk

Probably Approximately Correct Learning

M. Anthony and N. Biggs. *Computational Learning Theory*. Cambridge University Press, 1992.

Some previous definitions...

- A concept over a domain X is a function $c : X \to \{0, 1\}$.
- An hypothesis space is a set of concepts H over a domain X.
- We want to learn a target concept $t \in H$.
- We are given a training sample $\mathbf{s} = ((x_1, t(x_1)), \ldots, (x_m, t(x_m))) \in S(m, t)$.
- Each x_i is sampled according to a probabilistic distribution μ over X.
- A learning algorithm is a function $L : S(m, t) \to H$.

Probably Approximately Correct Learning

Example: Learning rays.

- Take $X = R$, the real numbers.
- For each real number θ define the *ray* concept r_θ such that: $r_\theta(y) = 1$ iff $y \geq \theta$.
- We want to learn in the hypothesis space $H = \{r_\theta | \theta \in R\}$.
- Given a sample $\mathbf{s}$, take $\lambda = \lambda(\mathbf{s})$ as the minimum x_i with $b_i = 1$.
- The learning algorithm is $L(\mathbf{s}) = r_\lambda$.

Example of the example

If $\mathbf{s} = ((3,1),(2,0),(\mathbf{1},1),(4,1))$, then $\lambda = 1$.

Probably Approximately Correct Learning

Definition

- The error of a hypothesis $h \in H$ w.r.t. $t \in H$ is defined as

$$er_\mu(h,t) = \mu\{x \in X | h(x) \neq t(x)\}$$

- Intuitively, it can be seen as the probability volume of the misclassified elements.
- L is a PAC learning algorithm for H if, given $0 < \delta, \epsilon < 1$, there is $m_0 = m_0(\delta, \epsilon)$ such that if we take a sample **s** of size $m \geq m_0$, then

$$er_\mu(L(\mathbf{s}), t) < \epsilon$$

with probability greater than $1 - \delta$.

- δ is called the confidence and ϵ the accuracy.

Probably Approximately Correct Learning

Observations

- The learning algorithm for rays is PAC learnable with $m_0 = \lceil \frac{\log \delta}{\log(1-\epsilon)} \rceil$
- PAC learning is the best one can hope within a probabilistic framework: one can only expect that is probable that the training sample is useful.
- There is another concept called *identification in the limit*:
  - In Clark and Eyraud (2005) it is proved that substitutable CFGs are identifiable in the limit.
  - To guarantee identification, a characteristic set of elements of the language must be part of the sample, but this can't be checked.
- PAC learning is nicer (and more practical) than identification in the limit because we have a relationship between the size of the sample and the confidence/accuracy of the result.

Non Terminally Separated Languages

Alexander Clark. 2006. PAC-learning unambiguous NTS languages. In *Proceedings of ICGI-2006*.

Definition

- A grammar $G = \langle \Sigma, V, P, A \rangle$ is NTS iff

$$\left.\begin{array}{l} N \stackrel{*}{\Rightarrow} \alpha\beta\gamma \\ M \stackrel{*}{\Rightarrow} \beta \end{array}\right\} \Rightarrow N \stackrel{*}{\Rightarrow} \alpha M \gamma$$

- A language is NTS if it can be described by an NTS grammar.
- Unambiguous NTS grammars are those grammars such that every string has only one derivation.
- We assume non-redundant and non-duplicate non-terminals.

Non Terminally Separated Languages

Some Properties

- The sets of yields of the non-terminals are all disjoint: if $X \neq Y$, then $y(X) \cap y(Y) = \emptyset$.
  - For instance,

$$\left.\begin{array}{l} S \rightarrow NV \\ N \rightarrow n|nN \\ V \rightarrow vN \end{array}\right\} \Rightarrow \left\{\begin{array}{l} y(S) = \{n^+vn^+\} \\ y(N) = \{n^+\} \\ y(V) = \{vn^+\} \end{array}\right.$$

- Unambiguous NTS are much more restricted than NTS.
  - For instance, $L = \{a^n | n > 0\}$ is NTS but not UNTS.

$$\left.\begin{array}{l} S \stackrel{*}{\Rightarrow} aa \\ S \stackrel{*}{\Rightarrow} a = \beta \end{array}\right\} \Rightarrow \left\{\begin{array}{l} S \stackrel{*}{\Rightarrow} aS \\ S \stackrel{*}{\Rightarrow} Sa \end{array}\right.$$

Learning Algorithm: Definitions

PCFGs

- A PCFG is a CFG $G = \langle \Sigma, V, P, I, \iota, \pi \rangle$ where ι is an initial symbol probability function and π is a production probability function (note that there are several initial symbols).
- A PCFG defines a distribution P_D over Σ^*.

L_∞ norm

- The L_∞ norm of a function F over a countable set X is

$$L_\infty(F) = \max_{x \in X} |F(x)|$$

  - Note that $L_\infty(F_1 - F_2)$ implies $F_1 = F_2$.
  - L_∞ is used as a distance measure between distributions.

Learning Algorithm: Definitions

Substrings and Contexts

- A context distribution is a function $C : \Sigma^* \times \Sigma^* \rightarrow [0, 1]$ such that $\sum_{l,r \in \Sigma^*} C(l, r) = 1$.
- Let P_D be a distribution over Σ^*.
- $E_D[u] = \sum_{l,r \in \Sigma^*} P_D(lur)$ is the expected number of times the substring u will occur (it can be > 1).
- The context distribution of a string u is

$$C_u^D(l, r) = \frac{P_D(lur)}{E_D(u)}$$

- Given a multiset M of contexts, we write $\hat{M}$ for the empirical distribution of the contexts.

Learning Algorithm

The PACCFG Algorithm

- Input: a sample $w_1, \ldots, w_N$. Parameters m_0, ν, μ_2.
- Steps:
  1. Take the substrings that appear more than m_0 times (set U).
  2. Compute the empirical context distribution $\hat{C}_u$ for each $u \in U$.
  3. Take the u's such that $L_\infty(\hat{C}) > \mu_2/2$ (set U_c).
  4. Build a graph with nodes U_c. Form an edge u, v iff $L_\infty(\hat{C}_u - \hat{C}_v) \leq 2\mu_3$ where $\mu_3 = \nu\mu_2/16$.
  5. Identify the set of connected components $\{[u]\}$ of the graph.
  6. Build the grammar: $\hat{V} = \{[u]\}, \hat{I} = \{[w_i]\}$ and $\hat{P}$ is
     - $[a] \to a$ for each $a \in \Sigma \cap U_c$.
     - $[uv] \to [u][v]$ for each $uv, u, v \neq \lambda \in U_c$.
- Example: for sample $ab, aabb$ and $m_0 = 2$ (in the blackboard).

Proof of PAC Property

(Unorganized) Hypothesis

- Assumes that the samples are all positive samples generated by a target PCFG G.
- Assumes that the PCFG is μ_1-distinguishable, μ_2-reachable and ν-separable with known values.
- Assumes known upper bounds on the number of non-terminals, productions, length of rhs's and expected number of substrings ($n, p,\ l$ and L resp.).
- Assumes a given confidence δ and precision ϵ.
- Defines the number of strings required $N = N(\mu_1, \mu_2, \nu, n, p, l, L, \delta, \epsilon)$.

Proof of PAC Property

(Implicit) Main Theorem

If the number of samples exceeds N, then with probability $> 1 - \delta$, PACCFG will generate a hypothesis grammar $\hat{G}$ such that $L(\hat{G}) \subseteq L(G)$ and $P_D(L(G) - L(\hat{G})) < \epsilon$.

(Reversed) Outline of the Proof

- Define μ_3-good sample.
- Prove that if the sample is μ_3-good (with $\mu_3 = v\mu_2/16$), then PACCFG will generate $\hat{G}$ with the desired properties.
- Prove that if the number of samples exceeds N, the sample is μ_3-good (same μ_3), with probability $> 1 - \delta$.
- Use some load of probability theory (Markov inequality, Chernoff bounds, negative association, etc.).

Discussion

From the paper

- There are no comparable results for PAC-learning CFGs.
- The results are incomplete: We are assuming μ-distinguishability, but it appears that it can be exponentially small (requiring exponentially big samples).

From my head

- PAC learning is a very interesting property but it is very difficult to achieve and prove by definition.
- Maybe PAC learnability of other classes of languages can be proved by some sort of reduction to PAC learnability of unambiguous NTS languages.
- This may be the case for k, l-substitutable context free languages, that are known to be identifiable in the limit (Yoshinaka, 2008).

Thank you!