# Computational Learning Theory

Pramook Khungurn

December 11, 2019

## 1 PAC Learning

- Computational learning theory seek to answer the following questions:
  - How good is the learned rule after $n$ examples?
  - How many examples do I need before the learned rule is accurate?
  - What can be learned and what cannot?
  - Is there a universally best learning algorithm?
- For the question of how many samples needed to learn, the answer is like this:

  > Any hypothesis that is seriously wrong will be "found out" with high probability after a samll number of examples, because it will make an incorrect prediction. Thus, any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong.

- A hypothesis that is "unlikely to be seriously wrong" is called **probably approximately correct** (PAC).
- Any learning algorithm that returns hypotheses that are probably approximately correct is called a **PAC learning algorithm**.
- To get a PAC learning algorith, we assume two things:
  - The probability distribution of samples is stationary. That is, all examples are drawn from the same distribution independently.
  - The target function $f$ is in the hypothesis space $\mathcal{H}$.
- Simplest PAC theorems deals with boolean functions. The lost function used is the 0/1 lost function.

$$L_{0/1}(y, \hat{y}) = \begin{cases} 1 & y \neq \hat{y} \\ 0 & y = \hat{y} \end{cases}.$$

- The **error rate** of a hypothesis $h$ is the expected generalization error for examples drawn from the stationary distribution:

$$\text{error}(h) = GenLoss_{L_{0/1}}(h) = \sum_{x,y} L_{0/1}(h(x), y) P(x, y).$$

  In other words, the error rate is the probability that $h$ misclassifies an example.
- A hypothesis $h$ is **approximately correct** if $\text{error}(h) < \epsilon$ where $\epsilon$ is a small constant.

- We can show that we can find $N$, such that, after seeing $N$ examples, all consistent hypothesis is approximately correct with high probability.
- We let $\mathcal{H}_{\text{bad}}$ denote the set of hypothesis that is not approximately correct.
- Let us calculate the probability that a bad hypothesis $h_b \in \mathcal{H}_{\text{bad}}$ is consistent with the first $N$ examples.

  We know that $\text{error}(h_b) > \epsilon$. So the probability of picking an example that $h_b$ classifies correctly is at most $1 - \epsilon$. So, the probability of picking $N$ such examples is less than $(1 - \epsilon)^N$. Hence,

  $$\Pr(\mathcal{H}_{\text{bad}} \text{ contains a consitent hypothesis}) \leq |\mathcal{H}_{\text{bad}}|(1 - \epsilon)^N \leq |\mathcal{H}|(1 - \epsilon)^N.$$

  The above probability is not more than a small number $\delta$ when

  $$|\mathcal{H}|(1 - \epsilon)^N \leq \delta$$

  Using the fact that $1 - \epsilon \leq e^{-\epsilon}$, we have that

  $$\begin{aligned} |\mathcal{H}|e^{-\epsilon N} &\leq \delta \\ e^{-\epsilon N} &\leq \delta/|\mathcal{H}| \\ -\epsilon N &\leq \ln \delta - \ln |\mathcal{H}| \\ N &\geq \frac{1}{\epsilon}\left( \ln \frac{1}{\delta} + \ln |\mathcal{H}| \right). \end{aligned}$$

- In conclusion, if the algorithm outpus a hypothesis with that many examples, then the hypothesis has error rate at most $\epsilon$ with probability at least $1 - \delta$.
- The number of required examples, as a function of $\epsilon$ and $\delta$ is called the **sampling complexity** of the hypothesis space.
- If $\mathcal{H}$ is the class of boolean functions, then $|\mathcal{H}| = 2^{2^n}$ where $n$ is the number of inputs. A PAC learning algorithm would require seeing at least $2^n$ examples, which is pretty much all the examples are there.
- How to escape seeing so many examples?

  We may insist that the hypothesis and the true function be simple. This limits the size of $|\mathcal{H}|$.

# 2 Learning Decision List

- An example of a simple hypothesis space is the **decision lists**.
- A decision list is a series of tests. Each of which is a conjunction of literals.

  If a test succeeds when applied to an example description, the decision list specifies the value to be returned.

  If the test fails, the processing continues with the next test in the list.
- A decision list is similar to a decision tree, but the branching structure is much simpler.
- If we allows tests of arbitrary size, the decision lists can represent any boolean function.
- If we allows tests to contain at most $k$ literals, we get a hypothesis space called $k$-**DL**.

  We use the symbol $k$-DL($n$) to denote the set $k$-DL using $n$ boolean attributes.

- We shall calculate the size of $k$-DL($n$).

  Let the set of all boolean expressions that are conjunctions of at most $k$ literals from $n$ boolean attributes by be denoted by $Conj(n, k)$.

  Because a decision list is constructed of tests, and because each test can be attached to either a YES or a NO outcome, or can be absent from the decision list, there are at most $3^{|Conj(n,k)|}$ distinct sets of component tests.

  Since we can order the tests in any particular order, we have that

  $$|k\text{-DL}(n)| \leq 3^{|Conj(n,k)|}|Conj(n,k)|!.$$

  We also have that

  $$|Conj(n,k)| = \sum_{i=0}^{k} \binom{n}{i} 2^i = O(n^k).$$

  Hence, using Sterling approximation, we have

  $$|k\text{-DL}(n)| = 2^{O(n^k \log n^k)}.$$

- This means that, after seeing

  $$N = \frac{1}{\epsilon}\left( \ln \frac{1}{\delta} + O(n^k \log n^k) \right)$$

  examples, any consistent hypothesis is approximately correct.

- There's a simple algorithm that finds a consistent decision list given a set of examples.

  The algorithm starts with an empty decision list. Then, it finds a test that evaluates to TRUE for some subset of the examples. It adds that test to the decision list and throws examples that agrees with the test away. It then continues on with the rest of the example until no example is left.

  Obviously, the algorithm is a PAC learning algorithm.