Assignment 1 INF367

Johanna Jøsang

September 6, 2020

1 PAC learning

1.1 Show that a CNF is PAC learnable in polynomial time

Let the learning framework F and its components be defined as in the assignment description.

Algorithm for learning a conjunction of literals

- Start with the hypothesis being the set of all possible literals.

$$h = \{v_1 \wedge \neg v_1 \wedge v_2 ... \wedge v_n \wedge \neg v_n\}$$

- Pick an example e according to distribution D. e can be either a positive example or a negative example.
  - Case 1: e is a positive example. Go through all literals v_i in h, and check what it is evaluated in e to see if it needs to be removed. Naturally, if v_i already is removed from h, the algorithm simply continues.
    * If v_i is evaluated to 0 in e, remove v_i from h.
    * If v_i is evaluated to 1 in e, remove $\neg v_i$ from h.
  - Case 2: e is a negative example. Since h initially entails all negative examples, these cannot be used to remove literals from h. Negative examples are therefore ignored by the algorithm.
- Continue to pick examples and update h, m number of times.

A valid example can have at most n literals (since a variable and it's negation cannot be present). Hence the algorithm must go through at most n literals per example it processes, and so the runtime for each hypothesis update is $O(n)$. This will happen m times, so we want to find an upper bound for m.

Since we are talking about PAC learning, we need enough examples so that with a probability $1-\delta$ the algorithm creates a hypothesis which on average has error less than ϵ.

A hypothesis h misclassifies when there is a literal z in h which is evaluated to 0 in a positive example. Let E^+ denote the set of positive examples. z would have been removed from h if the algorithm previously had received such an example. So, the probability of receiving an example $e \in E^+$ such that this z is removed from h can be written as:

$$D(\{e \in E^+ : z \text{ is evaluated to 0 in } e)\}$$

Before the algorithm starts receiving examples, $h = \{v_1 \wedge \neg v_1 \wedge v_2 ... \wedge v_n \wedge \neg v_n\}$. So in the worst case, there might be $2n$ literals that need to be removed. We can therefore say that if a literal z has a probability less that $\epsilon/(2n)$ of being removed, it is considered a "bad" literal. So the probability of that "bad" literal to have not been removed after m independently drawn examples is:

$$(1-\epsilon/(2n))^m$$

Since there are $2n$ possible literals, the probability of some bad literal to not have been removed from h is $2n(1-(\epsilon/(2n))^m$, by union bound[1].

Since δ represents the probability of getting a bad sample we have the bound:

$$2n(1-(\epsilon/(2n))^m \leq \delta$$

We now use the inequality $1+x \leq e^x$, where we let $x = -\epsilon$, in order to rewrite this as:

$$2ne^{-m\epsilon/(2n)} \leq \delta$$

If we rearrange the inequality with respect to m we get:

$$m \geq (2n)/\epsilon(\ln(2n) + \ln(1/\delta))$$

Therefore we can conclude that if the algorithm takes at least $(2n)/\epsilon(\ln(2n) + \ln(1/\delta))$ examples, it will construct a hypothesis that with a probability of at least $1-\delta$ will have a true error less than ϵ. Hence, the upper-bound runtime for learning a conjunction of literals with PAC is:

(upper bound for example processing)$\times$(upper bound for number for examples needed)

Which is:

$$n \times (2n)/\epsilon(\ln(2n) + \ln(1/\delta))$$

Hence, the runtime is bounded by a polynomial in n, $1/\delta$ and $1/\epsilon$, so F is PAC learnable polynomial time.

[1]Union bound: $D(A \cup B) \leq D(A) + D(B)$

1.2 If H is finite F is PAC learnable

Let us define the learning framework F as previously, with the additional specification that the hypothesis class H is finite. If H is finite then the we can show that the ERM rule will not overfit, meaning that, with a $1-\delta$ probability, the true error of a resulting hypothesis is less than ϵ, given that the training set is sufficiently large. This encompasses the definition of H being PAC-learnable.

We want to show that we can use a low ϵ and low δ, and construct a hypothesis h in H. Since error occurs when we get a bad training set, we want to upper-bound the probability of getting such a training set. This probability can be written as:

$$D^m(\{S_e : error(h_s, t, D) > \epsilon\})$$

Where S_e denotes the set of examples in the training set with categorization removed.

Bad hypothesise are those with error w.r.t. the target t greater than ϵ. Let H_ϵ denote the set of such bad hypothesise, which we can write as:

$$H_\epsilon = \{h \in H : error(h, t, D) > \epsilon\}$$

The set of misleading samples, denoted by M, consists of unlabeled examples such that there exists a hypothesis h in the set of bad hypothesise, which has a training error of 0. Essentially this is the set of samples that causes a hypothesis to overfit.

$$M = \{S_e : \exists h \in H_\epsilon \text{ s.t. } error_s(h) = 0\}$$

The *realizability assumption* states that for target t there exists a hypothesis $h \in H$ such that $error(h, t, D) = 0$. So, by the realizability assumption, $error(h_s, t, D) > \epsilon$ only if there is a h in H_ϵ s.t. $error_s(h) = 0$. (Otherwise $error(h_s, t, D) \leq \epsilon$.)

So,

$$\{S_e : error(h_s, t, D) > \epsilon\} \subseteq \{S_e : \exists h \in H_\epsilon \text{ s.t. } error_s(h) = 0\} = M$$

$$\{S_e : error(h_s, t, D) > \epsilon\} \subseteq M$$

By the definition of M, we can rewrite it as:

$$M = \cup_{h \in H_{\epsilon]}} \{S_e : error_s(h) = 0\}$$

So we get:

$$D^m(\{S_e : error(h_s, t, D) > \epsilon\}) \leq D^m(M) = D^m(\cup_{h \in H_{\epsilon]}} \{S_e : error_s(h) = 0\})$$

And thereafter apply union bound to yield:

$$D^m(\{S_e : error(h_s, t, D) > \epsilon\}) \leq \sum\nolimits_{h \in H_\epsilon} D^m(\{S_e : error_s(h) = 0\}) \quad (1)$$

The examples in the training set are sampled i.i.d., therefore:

$$D(\{S_e : error_s(h) = 0\}) = (D(\{s_i : h(s_i) = t(s_i)\}))^m, h \in H_\epsilon$$

As h is in H_ϵ, for each individual sampling s of the training set we have:

$$D(\{s_i : h(s_i) = t(s_i)\} = 1 - error(h, t, D) \leq 1 - \epsilon$$

Hence,

$$D^m(\{s_i : h(s_i) = t(s_i)\} \leq (1 - \epsilon)^m = e^{-\epsilon m} \quad (2)$$

From earlier we had that

$$D^m(\{S_e : error(h_s, t, D) > \epsilon\}) \leq \sum\nolimits_{h \in H_\epsilon} D^m(\{S_e : error_s(h) = 0\})$$

and so by combining equation 1 and 2, and recalling that H is finite, we can conclude that:

$$D^m(\{S_e : error(h_s, t, D) > \epsilon\}) \leq |H_\epsilon|^{-\epsilon m} \leq |H| e^{-\epsilon m}$$

Since the upper bound is for the probability of getting $error(h, t, D) > \epsilon$ we bound this by δ, and thereafter isolate m.

$$|H| e^{-\epsilon m} \leq \delta$$

$$\frac{|H|}{\delta} \leq \frac{1}{e^{-\epsilon m}}$$

$$ln(|H|/\delta) \leq \epsilon m$$

$$m \geq \frac{ln(|H|/\delta)}{\epsilon}$$

So if we have an $m \geq \frac{ln(|H|/\delta)}{\epsilon}$, the ERM rule will be PAC learnable, and this m is obtainable given that H is finite.