1 Lecture 3: 2017.02.06

Recall the definition of a class being PAC learnable.

Definition 1.1 (PAC Learning)**.** A class $\mathcal{C}$ is *Probably Approximately Correct (PAC) learnable* if there exists an algorithm, L, such that:

$$\forall c \in \mathcal{C}, \quad \forall D \text{ over } X, \quad \forall \varepsilon, \delta > 0 \tag{1.1}$$

- (Learning) With probability $\geq 1-\delta$, L outputs a hypothesis, h in $\mathcal{C}$ such that $D[h \triangle c] < \varepsilon$, i.e. we have error at most ε

$$Err(h) \coloneqq \mathbb{P}_{x \sim D}[h(x) \neq c(x)] \leq \varepsilon. \tag{1.2}$$

- (Efficient) L runs in time/sample *poly* $\left(\frac{1}{\varepsilon}, \frac{1}{\delta}, n\right)$. *Samples & computation.*

Remark 1.2. *Usually when we talk about computation time we are in the realm of complexity theory and we talk about samples we are really asking statistics/information-theory questions about what sample size do we need to be able to draw some conclusion.*

1.1 PAC learning 3-term DNF by 3-CNF

In the following we see how to overcome the intractability of the 3-DNF PAC learning by using a different representation. It amounts to expanding the input space and the class we are learning.

Definition 1.3. A 3-CNF is a conjunction of disjunctions of length three.

Given a 3-DNF, we can rewrite it as a 3-CNF in the following way:

$$T_1 \vee T_2 \vee T_3 \equiv \bigwedge_{\substack{u \in T_1 \\ v \in T_2 \\ w \in T_3}} (u \vee v \vee w), \tag{1.3}$$

for every assignment of $x_1, \ldots, x_n$, both sides evaluate to the same boolean value. Notice that the length of the 3-CNF can be much bigger (but still polynomial): 3-DNF is at most as $3n$ but 3-CNF could be up to n^3. In a sense, this corresponds to feature generation.

Remark 1.4. *Notice that the reverse is NOT true. Given a 3-CNF it might not be representable as a 3-DNF.*

Another important point to notice is that after the transformation into 3-CNF we are changing the distribution of the initial input space. But the definition of PAC learnable allows for any *distribution, so we are fine.*

The upshot here is that we can learn 3-CNF by 3-CNF but this problem contains the intractable problem of learning 3-DNF by 3-DNF. The trick is that we have a bigger solution space so we the 3-CNF algorithm is fed a 3-DNF, it has the option to output something outside the 3-DNF class, namely a 3-CNF.

Theorem 1.5. *3-CNF is PAC-learnable and 3-DNF. Further, by the discussion above, 3-DNF is learnable "by" 3-CNF.*

This motivates the more general definition for PAC-learnable that takes into account the solution class.

Definition 1.6 (PAC Learning)**.** A class $\mathcal{C}$ is *Probably Approximately Correct (PAC) learnable* by $\mathcal{C} \subset \mathcal{H}$ if there exists an algorithm, L, such that:

$$\forall c \in \mathcal{C}, \quad \forall D \text{ over } X, \quad \forall \varepsilon, \delta > 0 \tag{1.4}$$

- (Learning) With probability $\geq 1-\delta$, L outputs a hypothesis, $\underline{h \in \mathcal{H}}$ such that $D[h \triangle c] < \varepsilon$, i.e. we have error at most ε

$$Err(h) \coloneqq \mathbb{P}_{x \sim D}[h(x) \neq c(x)] \leq \varepsilon. \tag{1.5}$$

- (Efficient) L runs in time/sample *poly* $\left(\frac{1}{\varepsilon}, \frac{1}{\delta}, n\right)$. *Samples & computation.*

Remark 1.7. *$\mathcal{C}$ is usually called the target class and $\mathcal{H}$ the hypothesis class.*

1.2 Consistency implies PAC-learnable

- Suppose we have some target and hypothesis classes, $\mathcal{C} \subset \mathcal{H}$.
- Let A be a *consistent* algorithm:
  - i) Given $\underline{\text{any}}$ finite sample, $S = \{\langle x_1, y_1 \rangle, \ldots, \langle x_m, y_m \rangle\}$, where for all i we have $y_i = c(x_i)$ for some $c \in \mathcal{C}$.
  - ii) A outputs $h \in \mathcal{H}$ such that $h(x_i) = y_i = c(x_i)$ for all i.

Theorem 1.8. *For any finite $\mathcal{H}$, a consistent algorithm for $\mathcal{H}$ is a PAC-learnable algorithm.*

Proof. We call a hypothesis, $h \in \mathcal{H}$, ε-bad if $Err(h) > \varepsilon$. Now we generate a size-m S according to $Ex(c, D)$, which is the subroutine that generates samples from D. For any $\underline{\text{fixed}}$ ε-bad h, we have an upper bound on the probability of h being consistent with S

$$\mathbb{P}_S[h \text{ consistent with } S] \leq (1-\varepsilon)^m. \tag{1.6}$$

Therefore, $\mathbb{P}_S[\exists h \in \mathcal{H}$ that is both ε-bad and consistent with $S] \leq |\mathcal{H}|\,(1-\varepsilon)^m$. Now we choose $\delta > 0$ such that $|\mathcal{H}|\,(1-\varepsilon)^m < \delta$. We can conclude that as long as $m \geq \frac{const}{\varepsilon} \ln \frac{|\mathcal{H}|}{\delta}$ the PAC learning definition will be satisfied. □

Remark 1.9. $|\mathcal{H}|\,(1-\varepsilon)^m \leq e^{\ln|\mathcal{H}| + m\ln(1-\varepsilon)} \approx e^{\ln|\mathcal{H}| - c_0\varepsilon m} \to 0$ *as* $m \to \infty$. *A key point of this is that we can let the complexity of the hypothesis to grow as the sample size gets bigger and keeping this quantity under control. That is, the more data you have, the more complex the model you can train without over-fitting too much. If $\mathcal{H}_m$ is the hypothesis for data of size m, we only need* $\ln|\mathcal{H}_m| \leq c \cdot m^\beta$, *for some* $\beta < 1$ *and* $c = c(dim)$. *From here we obtain* $m \geq \left(\frac{c}{\varepsilon}\right)^{\frac{1}{1-\beta}}$.

Next we want to deal with the case of a possibly infinite hypothesis class $\mathcal{H}$. The first approach could be to try and force feed a discretization of the hypothesis into the finite case one but this might not work because it depends on the interaction between the distribution of the data and the chosen discretization. We want something better.

- Class $\mathcal{H}$ over X.
- $h \in \mathcal{H}$ as functions $h(x) \in \{0,1\}$; or as sets $h \subset X$.
- Let $S = \{x_1, \ldots, x_m\}$ be an ordered subset of X.
- $\Pi_{\mathcal{H}}(S) = \big\{\langle h(x_1), \ldots, h(x_n)\rangle : h \in \mathcal{H}\big\} \subseteq \{0,1\}^m$

  Remark 1.10. *If the inclusion above is saturated then it means that our hypothesis can classify ANY labeling of that size. That is bad for learning because it is saying that there is no structure in the data.*

- [**Definition**] We say that S is *shattered* if the the equality case holds, $\Pi_{\mathcal{H}}(S) = \{0,1\}^m$, or equivalently, $\big|\Pi_{\mathcal{H}}(S)\big| = 2^m$.
- [**Definition**] The Vapnik-Chervonenkis dimension of $\mathcal{H}$ as

$$VCD(\mathcal{H}) = \text{ size of the largest shattered set} \tag{1.7}$$

$$= \max_d \Big\{\exists S \subseteq X : |S| = d \quad \& \quad \Pi_{\mathcal{H}}(S) = \{0,1\}^d\Big\} \tag{1.8}$$

  Example. If $\mathcal{H}$ is finite, then $VCD(\mathcal{C}) \leq \ln|\mathcal{C}|$ because shattered d points requires 2^d functions.

  Rectangles in $\mathbb{R}^2$. The VCD dimension is 4 in this case because the rectangular convex hull of a set of points is always given by the four extremal points in each direction.