Computational Learning Theory Lecture Notes

Martin Citoler-Saumell

CIS625 Spring 2017

These are notes from Prof. Michael Kearns's CIS625. This course is currently being taught at the University of Pennsylvania and attempts to provide algorithmic, complexity-theoretic and probabilistic foundations to modern machine learning and related topics. You can find out more at the course website. A word of caution, it is likely that some typos and/or errors have been introduced during the taking of these notes. If you find any, please let me know at martinci@math.upenn.edu.

1 Lecture 1: 2017.01.23

1.1 Course Outline/Description

1.1.1 Formal Models of ML

- Assumptions about data generation process.
- Assumptions about what algorithms "knows".
- Sources of info that the algorithms has
- Criteria/objective of learning is.
- Restrictions on algorithms.

1.1.2 Examples of Models

- "PAC" model (in first 1/2 of the term).
- Statistical learning theory.
- "no-regrets" learning models.
- Reinforcement learning.
- ML & Differential privacy.
- "Fairness" in ML

1.2 A Rectangle Learning Problem

Suppose you are trying to teach an alien friend the ”shape” of humans in terms of abstract descriptions like ”medium build”, ”athletic”, etc. We are going to assume that each one of these descriptions represents a rectangular region on the height-weight plane but we are not aware of the exact dimensions. The only thing we are able to tell the alien is whether a particular individual is medium built or not. i.e. we can only label examples.

- <u>target</u> rectangle R, the *true* notion of ”medium built”.
- <u>hypothesis</u> rectangle $\hat{R}$.

Remark 1.1. *Note that the assumption that the classifier function is a rectangle is rather strong. There is always this trade-off to be able to actually compute things. From a Bayesian point of view, “we always need a prior”.*

Given a data cloud of examples, a reasonable hypothesis rectangle could be the tightest fit rectangle. However, this choice ignores the negative examples so it seems that that we are throwing away information. In a sense, this rectangle would be the least likely.

- Assume $\langle x_1, x_2 \rangle$ pairs of height-weight are i.i.d. from an arbitrary unknown probability distribution, D.

We want to be able to evaluate how our hypothesis rectangle is performing. We want bounds on the classification error, which can be thought as the size of the symmetric difference between R and $\hat{R}$

$$D[R \triangle \hat{R}] \equiv \mathbb{P}_D[R(\vec{x}) \neq \hat{R}(\vec{x})]. \tag{1.1}$$

Theorem 1.2. *Given $\varepsilon, \delta > 0$, there is some integer N such that if we have more than N training examples,*[1] *we have*

$$D[R \triangle \hat{R}] \equiv \mathbb{P}_D[R(\vec{x}) \neq \hat{R}(\vec{x})] < \varepsilon, \tag{1.2}$$

with probability at least $1 - \delta$.

Proof. First of all, note that using tightest fit, the hypothesis rectangle is always contained inside the target rectangle. Now, for each side of the target rectangle we may draw inward strips in such a way that each strip has $\mathbb{P}_D[Strip] < \frac{\varepsilon}{4}$. If the training set has a positive example in each of these four strips, then the inequality above is satisfied because the boundary of the hypothesis rectangle would be contained in the union of the strips. Next we need to deal with the required sample size to obtain this result with some certainty. Let m denote the sample size, since the distribution is i.i.d., we have

$$\mathbb{P}_D[\text{miss a specific strip m times}] = \left(1 - \frac{\varepsilon}{4}\right)^m,$$

$$\mathbb{P}_D[\text{miss any of the strips m times}] \geq 4\left(1 - \frac{\varepsilon}{4}\right)^m.$$

[1]This the same as saying that sample size is at least N.

By the discussion above, the last inequality implies

$$\mathbb{P}_{D^m}[D[R\triangle\hat{R}] \geq \varepsilon] \leq 4\left(1-\frac{\varepsilon}{4}\right)^m,$$

which can be chosen arbitrarily small for big enough m. One can obtain $N \geq \frac{4}{\varepsilon}\ln\left(\frac{4}{\delta}\right)$. $\square$

Remark 1.3. *This proof generalizes to d-dimensional rectangles. We only need to replace 4 with* $2d$*, the number of* $(d-1)$*-faces. We can also try to incorporate noisy data, where the labels have some probability of being wrong.*

1.3 A More General Model

- Input/instance/feature space, X. (e.g. $\mathbb{R}^2$ in the example above)
- Concept/classifier/boolean function, $C : X \to \{0,1\}$ or we can also think about it as an indicator function of the positive examples or the subset of positive examples.
- Concept class/target class, $\mathcal{C} \subset \mathcal{P}(X)$, the admissible concepts/classifiers. (e.g. all rectangles in $\mathbb{R}^2$ in the example above)
- $\underline{\text{target}}$ concept, $c \in \mathcal{C}$. (e.g. target rectangle in the example above)
- Input distribution, D over X (arbitrary & unknown)
- Learning algorithm given access to examples of the form: $\langle \vec{x}, y\rangle$ where $\vec{x}$ is i.i.d. drawn from D and $c(\vec{x}) = y$.

Definition 1.4 (PAC Learning)**.** We say that a class of functions over X, $\mathcal{C}$, is *Probably Approximately Correct (PAC) learnable* if there exists an algorithm, L, such that given any c in $\mathcal{C}$, D a distribution over X and $\varepsilon, \delta > 0$, L with these parameters and random inputs $\vec{x}$'s satisfies:

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

$$Err(h) := \mathbb{P}_{x\sim D}[h(x) \neq c(x)] \leq \varepsilon. \tag{1.3}$$

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

2 Lecture 2: 2017.01.30

Remark 2.1 (PAC Learning)**.** *Fixing the class* $\mathcal{C}$ *is a strong assumption, it is the prior you are assuming about the true behavior of the data. For example, when you fit a linear regression, you are assuming that there is a true linear relation in the data.*

In contrast, the assumption on the distribution over X *is fairly general.*

Theorem 2.2. *The class of rectangles over* $\mathbb{R}^2$ *from example above is PAC learnable (with sample size* $m \sim \frac{1}{\varepsilon}\ln\left(\frac{1}{\delta}\right)$*).*

2.1 PAC learning boolean conjunctions

In the following we are going to see an example of problem that is PAC learnable.

- $X = \{0,1\}^n$
- Class $\mathcal{C}$ = all conjunctions over $x_1, \ldots, x_n$. $|\mathcal{C}| = 3^n$
  E.g.: If $c = x_1 \wedge \neg x_3 \wedge \neg x_{11} \wedge x_{26} \ldots$,
$$c(\vec{x}) = 1 \iff x_1 = 1, x_3 = 0, x_{11} = 0, x_{26} = 1, \ldots \tag{2.1}$$
- D over $\{0,1\}^n$.

2.1.1 Algorithm for monotone case (i.e. no $\neg$'s)

In this case we are trying to fit something like $c = x_1 \wedge x_5 \wedge x_{13} \ldots$. i.e. given some examples of sequences of bits and the result of c on them, we are trying to guess what the conjunction c actually is. We can use the following algorithm:

- $h \leftarrow x_1 \wedge x_2 \wedge x_3 \wedge \ldots \wedge x_n$, start with the conjunction of all the variables.
- For each positive example, $\langle \vec{x}, 1 \rangle$, delete any variable in h such that $x_i = 0$.
  This method ensures that the positives of h is a subset of the true c.

Analysis

Let p_i denote the probability we delete x_i from h in a single draw. In other words,
$$p_i = \mathbb{P}_{\bar{X} \sim D}[c(\vec{x}) = 1, x_i = 0]. \tag{2.2}$$

Then we have an a priori bound on error: $Err(h) \leq \sum\limits_{x_i \in h} p_i$. We can make a distinction between *bad* and *good* indices. An index, i, is bad if $p_i \geq \frac{\varepsilon}{n}$ and good otherwise. Note that if h contains no bad indices, then we have
$$Err(h) \leq \sum_{x_i \in h} p_i \leq n \left(\frac{\varepsilon}{n}\right). \tag{2.3}$$

Let's fix some index, i, such that $p_i \geq \frac{\varepsilon}{n}$. i.e. a bad index. We have that
$$\begin{aligned} \mathbb{P}[x_i \text{ "survives" m random samples}] &= (1 - p_i)^m \quad (iid) && (2.4) \\ &\leq \left(1 - \frac{\varepsilon}{n}\right)^m && (2.5) \\ \mathbb{P}[\textit{any} \text{ bad i "survives" m random samples}] &\leq n \left(1 - \frac{\varepsilon}{n}\right)^m. && (2.6) \end{aligned}$$

If we want the right-hand-side of the last inequality to be less than some $\delta > 0$, we end up with $m \geq \frac{n}{\varepsilon} \ln\left(\frac{n}{\delta}\right)$. In other words, we just proved the following theorem

Theorem 2.3. *Conjunctions over $\{0,1\}^n$ are PAC learnable with sample size $m \geq \frac{n}{\varepsilon} \ln\left(\frac{n}{\delta}\right)$.*

Remark 2.4. *An analogous argument proves this theorem for conjunctions that are not necessarily monotone. The only difference is that we have to keep track the extra $\neg$ variables.*

Remark 2.5. *We can identify some pattern for this kind of analysis. We identify some "bad" things that may happen and then prove that the probability of them happening decreases fast when we increase the number of samples seen.*

2.2 Hardness of PAC learning 3-term DNF

Now we are going to see an example of non PAC learnable problem. However, we will be able to slightly modify it and achieve PAC learning. This motivates an better definition of PAC learnable.

- Input space $X = \{0,1\}^n$
- Class $\mathcal{C}$ = all disjunctions of three conjunctions. $|\mathcal{C}| = 3^{3n}$
  E.g.: If $c = T_1 \vee T_2 \vee T_3$ where T_i is a conjunction over X.
  $c = (x_1 \wedge x_7 \wedge \neg x_8) \vee (x_2 \wedge x_5 \wedge \neg x_7 \wedge \neg x_8) \vee (x_6 \wedge x_{12})$.

To see that this problem is *hard*, we prove that the graph 3-coloring problem reduces to the 3-term DNF problem.

Graph 3-coloring to PAC learning 3-term DNF

Suppose that you have a 3-colorable (undirected) graph G. That is, a graph such that we can color the vertices with 3 colors in such a way that there are no edges between vertices of the same color. We see an example of how to transform such a graph into a set of labeled examples for the PAC learning 3-term DNF.

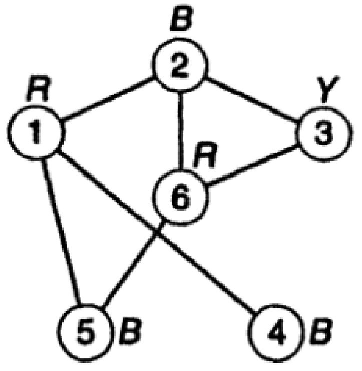

Figure 1: A graph with a 3-coloring.

First, for the i-th node we create a positive example that is represented by the vector, $v(i)$, with a 0 in the i-th entry and 1's everywhere else. E.g. from node one we have $\langle 011111, + \rangle$. Then we create a negative example from each edge that is represented by a vector, $e(i,j)$, of 1's except for 0's at the positions that determine the edge. E.g. from the edge connecting 2 and 3 we obtain $\langle 100111, - \rangle$. Next, the coloring can be used to define a 3-term DNF in the following way. Given a color, we define T_{color} as the conjunction of the variables/vertices that are *not* of that color. In this example we have

$$T_R = x_2 \wedge x_3 \wedge x_4 \wedge x_5, \tag{2.7}$$
$$T_B = x_1 \wedge x_3 \wedge x_6, \tag{2.8}$$
$$T_Y = x_1 \wedge x_2 \wedge x_4 \wedge x_5 \wedge x_6. \tag{2.9}$$

Claim 2.6. *G is 3-colorable $\iff$ there is a 3-term DNF* consistent *with the labeled sample above.*

Proof. We have just seen how to obtain a 3-term DNF from a 3-colorable graph. We only need to check that it is consistent with the sample. By construction, all the positive examples satisfy T_{color} where color is the vertex's color, since the only 0 in the vector is in the position that is dropped. Similarly, it follows that the examples coming from the edges do not satisfy any of the T's. In any edge there are two colors corresponding to its vertices. The extra 0 in the example ensures that the T's from those colors are not satisfied. Finally, the T of the remaining color cannot be satisfied because both vertices are included in the conjunction and they are both 0 in the example.

Conversely, given a graph and the labeled examples as above, if there is a consistent 3-term DNF we can find a coloring in the following way. Label each term of the formula with a color, say $T_R \vee T_Y \vee T_B$, and remember the order of the labels. Then, we define the color of vertex i (corresponding to vector $1 \ldots 101 \ldots 1$) as the label of the first formula that is satisfied by $v(i)$. Since the formula is consistent with the sample, every vertex must be actually colored. We only need to argue that this is a valid coloring. Suppose to the contrary that i and j are to vertices that are connected by an edge and have the same color. This means that both $v(i)$ and $v(j)$ satisfy T_{C_0}. However, we also have $v(i) \& v(j) = e(i,j)$ where & denotes the bit-wise and operation and it follows that $e(i,j)$ satisfies T_{C_0}, a contradiction with the consistency of the formula since edges are negative examples. $\square$

This concludes the argument that finding a coloring of a graph is the same as producing a consistent 3-term DNF. We are only left with the computational aspects. Namely, given the labeled sample associated to a graph, we need to find a way to feed it to a PAC learning algorithm. First we need a distribution over vectors of bits. For this we can just sample the examples uniformly. Finally, to ensure consistency we choose ε any quantity less than $\frac{1}{\#examples}$[2]. This way the algorithm cannot make any mistakes is forced to be consistent. The δ can be arbitrary. In conclusion, if the 3-term DNF problem were PAC learnable, we could solve the coloring problem in random polynomial time. Another way to say this is in the form of the following theorem.

Theorem 2.7. *If 3-term DNF are PAC learnable, then $NP = RP$.*

Of course, it is strongly believed that $RP \subsetneq NP$ so this is rather strong evidence against the easiness of the 3-term DNF problem.

[2]This epsilon is allowed because $\frac{1}{\varepsilon}$ is polynomial in the size of input. This is not true in general.

Remark 2.8. *The upshot is that a slight generalization of the conjunction problem, for which we have a randomized polynomial time solution, is almost assured to not be PAC learnable (unless NP = RP)*

3 Lecture 3: 2017.02.06

Recall the definition of a class being PAC learnable.

Definition 3.1 (PAC Learning)**.** A class $\mathcal{C}$ is *Probably Approximately Correct (PAC) learnable* if there exists an algorithm, L, such that:

$$\forall c \in \mathcal{C}, \quad \forall D \text{ over } X, \quad \forall \varepsilon, \delta > 0 \tag{3.1}$$

- (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

$$Err(h) \coloneqq \mathbb{P}_{x \sim D}[h(x) \neq c(x)] \leq \varepsilon. \tag{3.2}$$

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

Remark 3.2. *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.*

3.1 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 3.3. 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:

$$T_1 \vee T_2 \vee T_3 \equiv \bigwedge_{\substack{u \in T_1 \\ v \in T_2 \\ w \in T_3}} (u \vee v \vee w), \tag{3.3}$$

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 3.4. *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 3.5. *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 3.6 (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:

$$\forall c \in \mathcal{C}, \quad \forall D \text{ over } X, \quad \forall \varepsilon, \delta > 0 \tag{3.4}$$

- (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

$$Err(h) := \mathbb{P}_{x \sim D}[h(x) \neq c(x)] \leq \varepsilon. \tag{3.5}$$

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

Remark 3.7. *$\mathcal{C}$ is usually called the target class and $\mathcal{H}$ the hypothesis class.*

3.2 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 3.8. *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

$$\mathbb{P}_S[h \text{ consistent with } S] \leq (1-\varepsilon)^m. \tag{3.6}$$

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 3.9. $|\mathcal{H}|\,(1-\varepsilon)^m \le e^{\ln|\mathcal{H}|+m\ln(1-\varepsilon)} \approx e^{\ln|\mathcal{H}|-c_0\varepsilon m} \to 0$ *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* $\ln|\mathcal{H}_m| \le c \cdot m^\beta$*, for some* $\beta < 1$ *and* $c = c(dim)$*. From here we obtain* $m \ge \left(\frac{c}{\varepsilon}\right)^{\frac{1}{1-\beta}}$.

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 $\underline{\text{ordered}}$ 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 3.10. *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

$$\begin{aligned} VCD(\mathcal{H}) &= \text{ size of the } \underline{\text{largest}} \text{ shattered set} && (3.7)\\ &= \max_d \Big\{\exists S \subseteq X : |S| = d \quad \& \quad \Pi_{\mathcal{H}}(S) = \{0,1\}^d\Big\} && (3.8)\end{aligned}$$

Example. If $\mathcal{H}$ is finite, then $VCD(\mathcal{C}) \le \ln|\mathcal{C}|$ 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.