CSE 513T Theory of Artificial Intelligence & Machine Learning January 18, 2018

Lecture 2

Lecturer: Brendan Juba *Scribe: Matthew Stern*

Contents

1 Warm-Up Problem **1**

2 Conjunction Learning Algorithm **1**
2.1 Last Lecture 1
2.2 Elimination Algorithm 2
2.3 Theorem 2
2.4 Proof 2

3 PAC-Learning **3**

4 Other Representations **3**
4.1 Disjunctions: ORs of literals 3
4.2 DNF Formulas (Disjunctive Normal Form): OR of ANDs of literals 3

1 Warm-Up Problem

Suppose that there is a 5% chance that I "Give you up" and a 10% chance that I "Let you down." What is the best lower bound on the probability that I don't "Give you up" or "Let you down?"

Answer:
G = "Give you up", L = "Let you down"
Union Bound Approximation: Pr(G $\vee$ L) $\leq$ Pr(G) + Pr(L) = 0.05 + 0.10 = 0.15
Pr($\neg$(G $\vee$ L)) = 1 - Pr(G $\vee$ L) $\geq$ 1 - 0.15 = 0.85

2 Conjunction Learning Algorithm

2.1 Last Lecture

Last lecture, we discussed giving an algorithm for finding a conjunction. Given "labeled examples" over some instance space $X = \{0,1\}^n$, we want to find a conjunction (an AND of "literals" [attributes or negations of attributes]) defining the unknown property. We do this under the hope/belief/assumption that there is some c(x) (c: x $\rightarrow$ {0,1}) given by a conjunction.
From last lecture, we cannot know c exactly unless we consider every possible x $\in X$. Even then, there can/might be multiple representations of the same concept (for example, $X_1 \wedge \neg X_1$ is the same as $X_n \wedge \neg X_n$ is the same as $X_1 \wedge X_2 \wedge X_3 \wedge \neg X_1 \wedge \neg X_2 \wedge \neg X_3$). As a result, we can't hope to get "the" representation c.
New objective: Assume that our x have been drawn from a common, unknown distribution D over X. Our goal is to find a representation h such that $Pr_{x \in D}(h(x) = c(x)) \geq 1 - \epsilon$ for some input error tolerance ϵ.

2.2 Elimination Algorithm

Data: Examples = $(x^1, b^1), (x^2, b^2), (x^3, b^3), ..., (x^m, b^m)$
Result: Our representation h
Initialize h = $x_1 \wedge x_2 \wedge ... \wedge x_n \wedge \neg x_1 \wedge \neg x_2 \wedge ... \wedge \neg x_n$;
for *i = 1,...,m* **do**
  if $b^{(i)}$ **then**
    for *j = 1,...,n* **do**
      if $x_j^{(i)}$ *= 1* **then**
        Remove $\neg x_j$ from h;
      else
        Remove x_j from h;
      end
    end
  end
end
return *h*

Algorithm 1: Elimination Algorithm

2.3 Theorem

Given $m \geq \frac{2n}{\epsilon}(ln(2n) + ln(\frac{1}{\delta}))$ examples, elimination returns a representation h such that $Pr_{x \in D}(h(x) = c(x)) \geq 1 - \epsilon$ with probability $\geq 1 - \delta$. Moreover, it runs in time $O(mn)$.

2.4 Proof

First, note that every literal in the unknown c is included initially. We only delete a literal l from h when c(x) = 1 but l(x) = 0.
Claim: l can't be in c. Why? If c included l, $l(x^{(i)}) = 0$, so then we'd have c($x^{(i)}$) = 0, but c($x^{(i)}$) = 1 so we have a contradiction. Therefore, the h we return contains all of the literals of the unknown c.
So, if c(x) = 0, some l in c is false on x, and therefore the h we return also satisfies h(x) = 0.
So, the only possible disagreements occur because h contains literals that c does not, and then h(x) = 0 but c(x) = 1.
Let's say that a literal $\widetilde{l}$ is "bad" if $Pr_{x \in D}(\widetilde{l}(x) = 0 \wedge c(x) = 1) > \frac{\epsilon}{2n}$
If h does not contain any bad literals, $Pr_{x \in D}(h(x) \neq c(x)) \subseteq \sum_{l \ in \ h \ but \ not \ c} Pr_{x \in D}(l(x) = 0 \wedge c(x) = 1)$
$\sum_{l \ in \ h \ but \ not \ c} Pr_{x \in D}(l(x) = 0 \wedge c(x) = 1) \leq \frac{\epsilon}{2n} * 2n = \epsilon \rightarrow Pr(h(x) = c(x)) \geq 1 - \epsilon$
So, it suffices to show that with probability $1 - \delta$, h does not contain any bad literals.
A bad literal $\widetilde{l}$ is going to be deleted if we draw an example $x^{(i)}$ on which $c(x^{(i)}) = 1$ and $\widetilde{l}(x^{(i)}) = 0$
Since $\widetilde{l}$ is bad, $Pr(c(x^{(i)}) = 1 \wedge \widetilde{l}(x^{(i)}) = 0) > \frac{\epsilon}{2n}$
Recall that each $x^{(i)}$ is drawn independently from D. The probability that $\widetilde{l}$ survives all m examples is $\leq (1 - \frac{\epsilon}{2n})^m$.
By another union bound over the bad literals, the probability that <u>any</u> bad literal survives all examples is at most $\leq \sum_{bad \ \widetilde{l}} (1 - \frac{\epsilon}{2n})^m \leq 2n(1 - \frac{\epsilon}{2n})^m$
$1 - x \leq e^{-x}$ for all x $\rightarrow 1 - \frac{\epsilon}{2n} \leq e^{-\frac{\epsilon}{2n}} \rightarrow (1 - \frac{\epsilon}{2n})^m \leq e^{-\frac{\epsilon}{2n}m} \rightarrow 2n(1 - \frac{\epsilon}{2n})^m \leq 2ne^{-\frac{\epsilon m}{2n}}$
So, for our choice of m (i.e. solving for $2ne^{-\frac{\epsilon m}{2n}} = \delta$), we find that $Pr(h \ contains \ bad \ \widetilde{l}) < \delta$.
So, with probability 1 - δ, h does not contain bad literals and thus $Pr(c(x) = h(x)) \geq 1 - \epsilon$

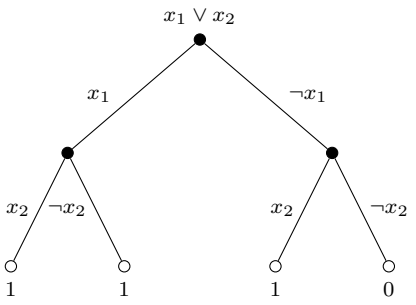
3 PAC-Learning

Let C_n (for int n) be a class of representations over an n-attribute instance space x_n. Then $\bigcup_{n \in N} C_n$ is PAC-learnable if there is an algorithm that given input $\epsilon \in (0, \frac{1}{2}), \delta \in (0, \frac{1}{2})$, and given access to labeled examples of some $c \in C$ drawn independently from some D over X_n runs in polynomial time in $n, \frac{1}{\epsilon}, \frac{1}{\delta}$, and the size of the representation c, and with probability $\geq 1 - \delta$ over both the draws of examples from D and any random choices of the algorithm, returns a representation h such that $Pr_{x \in D}(h(x) = c(x)) \geq 1 - \epsilon$.
We saw that for C being conjunctions (where any conjunction over n attributes is represented using $O(n)$ symbols), conjunctions are PAC-learnable. We saw decision trees last time, and we know that decision trees can express all conjunctions, but not vice versa. Decision trees are not yet known to be PAC-learnable.

4 Other Representations

4.1 Disjunctions: ORs of literals

Can every disjunction be written as a conjunction? We claim no.
Consider: $x_1 \vee x_2$. Since this is true when $x_1 = 1$ and $x_2 = 0$, the conjunction can't contain x_2. Similarly, since the rule is true when $x_1 = 0$ and $x_2 = 1$, the conjunction can't contain x_1 either. The conjunction can only contain $\neg x_1$ or $\neg x_2$, but when $x_1 = 0$ and $x_2 = 0$, $x_1 \vee x_2 = 0$, so the conjunction can't contain $\neg x_1$ or $\neg x_2$. Therefore, we are left with the always true empty conjunction.
By a similar construction to last lecture, decision trees can also express disjunctions.



4.2 DNF Formulas (Disjunctive Normal Form): OR of ANDs of literals

Example: $(x_1 \wedge \neg x_3 \wedge x_2 \wedge x_4) \vee (x_2 \wedge x_3 \wedge \neg x_4) \vee (\neg x_5) \vee ...$
We sometimes call the ANDs "terms" of the DNF (in the previous example, we have a "term of size 4, a "term" of size 3, a "term" of size 1, ...)
DNFs can express both conjunctions (single term) and disjunctions (terms of size n).
Is there anything DNFs cannot express? No, since we can simply write out the truth table, write a conjunction for each "true" line in the table, and OR them together.
There is, nevertheless, a sense in which DNFs are more powerful than decision trees. There is a DNF of size s that requires an exponential size decision tree of size $2^{\Omega(s)}$, but every decision tree of size s has a DNF of size $O(s^2)$.