

Lecture 6 : February 22, 2007

Lecturer: Rocco A. Servedio

Scribe: Andrew Wan

1 Today's Lecture

- Open questions on PTF bounds
- Uniform Distribution Learning:
 1. Definition
 2. Relationship with Online Mistake Bound and PAC Models
 3. Learning conjunctions, DNFs

2 Open Questions On PTF Bounds

1. What is the PTF degree of $\text{MAJ}(x_1, \dots, x_n) \wedge \text{MAJ}(y_1, \dots, y_n)$? Currently the bounds on the degree d are:

$$\Omega\left(\frac{\log n}{\log \log n}\right) \leq d \leq \Omega(\log n)$$

2. Give a stronger lower bound on the intersection of two halfspaces $h_1 \wedge h_2$ for arbitrary halfspaces h_1, h_2 . Currently, the best lower bound is $\Omega\left(\frac{\log n}{\log n \log n}\right)$.

$\omega\left(\frac{\log n}{\log \log n}\right)$ PTF bound on $h_1 \wedge h_2$.

3. Show that any intersection of two halfspaces has PTF degree $o(n)$. The upperbound is currently $O(\log W)$ where both halfspaces have weight at most W . Note that the $\Omega\left(\frac{\log n}{\log \log n}\right)$ lower bound rules out the approach from previous lecture: since there is a halfspace with weight $2^{\Theta(n \log n)}$, the approach can not give better than a $O(n)$ bound.
4. Give a $2^{o(n)}$ -time Online Mistake Bound algorithm for learning an intersection of halfspaces.

3 Uniform Distribution Learning

Readings on uniform distribution learning (see webpage for details):

- KV chapter 1 §4.1 and appendix 9 for more on the PAC (probably approximately correct) learning model and related tools
- Manseur '94
- Verbeurgt '88.

3.1 Definitions and Notation

We consider a model where the learner has access to an example oracle which outputs uniform random examples labeled according to the target function. This is a restricted variant of the usual PAC learning model.

Definition 1 Given $f : \{-1, 1\}^n \rightarrow \{-1, 1\}$, a **uniform example oracle** for f , written $EX(f)$, is a 'black box' that when invoked draws an example x from the uniform distribution \mathcal{U} over $\{-1, 1\}^n$ and outputs the ordered pair $(x, f(x))$.

Note that when the oracle is called multiple times in succession, each successive draw is independent.

Definition 2 Given functions $f, h : \{-1, 1\}^n \rightarrow \{-1, 1\}$, the **error** of h on f is

$$\text{error}(h, f) \equiv \Pr_{x \in \mathcal{U}}[h(x) \neq f(x)] = \frac{|\{x \in \{-1, 1\}^n : h(x) \neq f(x)\}|}{2^n}$$

Definition 3 Given any concept class C over $\{-1, 1\}^n$, a **uniform distribution (PAC) learning algorithm** for C is an algorithm A that on input $0 < \epsilon, \delta < 1$ and given access to a uniform example oracle $EX(c)$, where $c \in C$ is unknown to A , with probability at least $1 - \delta$ outputs a **hypothesis** h satisfying $\text{error}(h, f) \leq \epsilon$.

Notation.

- m – the **sample complexity** of the algorithm (number of samples required to get the desired output hypothesis, as a function of parameters of the algorithm)
- ϵ – the **accuracy** of the output hypothesis
- δ – the **confidence** in the accuracy of the output hypothesis

Note the following important considerations, which are true for PAC learning generally.

- Runtime: an **efficient** PAC algorithm should run in $\text{poly}(n, s, \frac{1}{\epsilon}, \frac{1}{\delta})$ time. Clearly, the runtime order is at least $\Omega(m)$.
- Evaluatability: a PAC algorithm may utilize any form of hypothesis representation that is **efficiently evaluatable**, i.e. if a PAC algorithm A runs in $p(n, s, \frac{1}{\epsilon}, \frac{1}{\delta})$ for some polynomial p , then evaluating h on any x should take at most $p(n, s, \frac{1}{\epsilon}, \frac{1}{\delta})$ time.
- Generality: uniform distribution learning is a special case of PAC learning, so results that are valid for general PAC learning are also valid for UD learning. (N.B.: The converse of this statement is not generally true.)

For the concept classes we've discussed so far, the runtime of the best *known* general PAC learning algorithm has roughly the same runtime as the best *known* algorithm in the Online Mistake Bound model. But in the uniform distribution setting, we can sometimes do better.

class	OLMB	PAC	Uniform
size s decision trees	$n^{\log s}$	$\frac{1}{\epsilon} n^{\log s} \log \frac{1}{\delta}$	$\frac{1}{\epsilon} n^{\log s} \log \frac{1}{\delta}$
s -term DNF	$2^{n^{1/3} \log s}$	$\frac{1}{\epsilon} 2^{n^{1/3} \log s} \log \frac{1}{\delta}$	$\frac{1}{\epsilon} n^{\log s / \epsilon} \log \frac{1}{\delta}$
intersection of weight- W halfspaces	$n^{O(\log W)}$	$\frac{1}{\epsilon} n^{O(\log W)} \log \frac{1}{\delta}$	$n^{\frac{1}{\epsilon^2}} \log \frac{1}{\delta}$

3.2 Useful facts about PAC learning

The following facts, which are true for general PAC learning, will be useful for getting results on UD learning.

1. **Fact 1** A PAC algorithm's runtime dependence on δ can always be made $O(\log \frac{1}{\delta})$.

How is this possible? Suppose we want to get arbitrarily small confidence and accuracy of ϵ' and δ' respectively. Then we can use an algorithm A with input parameters $\delta = \frac{1}{4}$, $\epsilon = \frac{\epsilon'}{2}$ and run it $T = O(\log \frac{1}{\delta'})$ times. Because each hypothesis of the series h_1, \dots, h_T was independently generated, the probability that none of them is $\frac{\epsilon'}{2}$ -accurate will be at most $\frac{\delta'}{2}$ (with a suitable choice of T).

Now run each of the h_1, \dots, h_T on M fresh examples to estimate the error rate, and pick as h the h_i with the lowest error rate. We can show that, for a suitable value of M , $\Pr[\text{error}(h_i) > \epsilon'] \leq \frac{\delta'}{2}$.

Hence, we can more or less disregard δ in finding good algorithms.

2. **Fact 2** *If we ignore runtime, we can learn any finite concept class C with $\frac{1}{\epsilon}(\ln |C| + \ln \frac{1}{\delta})$.*

The simple algorithm draws M examples from the example oracle and searches all $h \in C$ until it finds one that gets all M examples right. Let c be the concept class and call $h \in C$ **bad** if $\Pr[h(x) \neq c(x)] > \epsilon$. Then for any fixed bad h ,

$$\Pr[h \text{ gets } M \text{ examples right}] \leq (1 - \epsilon)^M$$

There are at most $|C|$ bad hypotheses, so

$$\Pr[\text{any bad } h \text{ is output}] \leq |C|(1 - \epsilon)^M$$

Taking M to be $\frac{1}{\epsilon}(\ln |C| + \ln \frac{1}{\delta})$ bounds this probability by δ .

3. **Fact 3** *If A is an online learning algorithm for C with mistake bound M that runs in time T , then there is a PAC algorithm that uses*

$$m = M + \frac{M + 1}{\epsilon} \log\left(\frac{M + 1}{\delta}\right)$$

many calls to $EX(c, \mathcal{D})$ and runs in time $\text{poly}(T, m)$.

The idea is to run A until it labels $\frac{1}{\epsilon} \ln \frac{M+1}{\delta}$ many examples correctly. Since A will output at most $M + 1$ hypotheses, the one it outputs will be ϵ -bad with probability at most δ . Since A will make at most M mistakes, it can make a mistake then a streak of examples of length at most $\frac{1}{\epsilon} \ln \frac{M+1}{\delta}$ at most M times before it will label a streak of examples correctly.

3.3 Learning Conjunctions

Now we give two basic but useful learning results.

Let $C = \{\text{all conjunctions over } x_1, \dots, x_n\}$. For example, $c(x) = \bar{x}_1 \wedge x_5 \wedge x_6$. We want a uniform distribution algorithm that learns c to accuracy ϵ .

- Given $\epsilon > 0$, draw $\text{poly}(\frac{1}{\epsilon}, \log \frac{1}{\delta})$ many examples to estimate $\Pr_{x \in \mathcal{U}}[c(x) = 1]$ to within an additive factor of $\pm \frac{\epsilon}{4}$. If the estimate is $< \frac{\epsilon}{2}$, output “ $h \equiv 0$ ”.
- Otherwise, we can assume that $\Pr_{x \in \mathcal{U}}[c(x) = 1] \geq \frac{\epsilon}{4}$, so we know that every $O(\frac{1}{\epsilon})$ examples gives at least one positive example.
- We say that x_j is **bogus** if it does not appear in the target conjunction c . Given k independent positive examples, if x_j is bogus, then

$$\Pr[x_j \text{ has the same value in all } k \text{ examples}] = \frac{1}{2^{k-1}}$$

So

$$\Pr[\text{any bogus variable has the same value in all } k \text{ examples}] \leq \frac{n}{2^{k-1}}$$

We require $\frac{n}{2^{k-1}} < \delta$, which implies that

$$k = O(\log \frac{n}{\delta}).$$

So after k positive examples, with probability at least $1 - \delta$ the set of variables consistent with all k examples is the exactly the set of variables in c . The total number of examples needed is $\text{poly}(1/\epsilon) + O(\frac{\log(n/\delta)}{\epsilon})$.

3.4 Learning s -term DNF over $\{0, 1\}^n$

Consider any s -term DNF f such that $f = T_1 \vee \dots \vee T_s$, where T_1, \dots, T_s are conjunctions of any length.

Fact 4 For $f = T_1 \vee \dots \vee T_s$, let f' be f , but with all terms of length $> \log \frac{2s}{\tau}$ removed. Then $\Pr_{x \in \mathcal{U}}[f(x) \neq f'(x)] \leq \frac{\tau}{2}$.

Proof: Any given term of length k is satisfied with probability $\frac{1}{2^k}$, so any term discarded is satisfied with probability $\leq \frac{\tau}{2^s}$. Since $\leq s$ terms can be discarded, the probability one of these long terms will affect the output of $f(x)$ is at most $\frac{\tau s}{2^s} = \frac{\tau}{2}$. ■

Recall the Winnow algorithm, which can be used to learn a disjunction of k unknown variables with mistake bound $O(k \log n)$. We can use this algorithm to PAC learn (for any distribution) any length k disjunction with $O(\frac{k \log n}{\epsilon})$ examples.

Let $r \geq \log(\frac{2s}{\tau})$. Given example $x \in \{0, 1\}^n$, we can view x as a $N = \sum_{i=1}^r 2^i \binom{n}{i} \approx n^r$ dimensional vector $X \in \{0, 1\}^N$ where coordinates are conjunctions of length at most r . Note that the uniform distribution over $\{0, 1\}^n$ induces a distribution \mathcal{D} over $\{0, 1\}^N$ that is not uniform.

The DNF f' is a disjunction of $\leq s$ of these variables over N -dimensional space. If we can get $(\frac{s \log N}{\epsilon}) = M \leq \frac{sr \log n}{\epsilon}$ examples from the oracle $EX(f', \mathcal{D})$, then Winnow can be used to get a hypothesis h such that $Pr_{X \in \mathcal{D}}[h(X) \neq f'(X)] \leq \frac{\epsilon}{2}$ which implies that $Pr_{x \in \mathcal{U}}[h(x) \neq f'(x)] \leq \frac{\epsilon}{2}$. By the previous fact we have that $error(f, f') \leq \tau/2$, so $error(h, f) \leq \tau/2 + \epsilon/2 \leq \epsilon$ if $\tau \leq \epsilon$.

We can take $\tau = \frac{1}{100} \frac{\epsilon}{sr \log n}$ and with probability at least 99/100, a group of M examples drawn from $EX(f, \mathcal{U})$ will be distributed identically to a group of M examples drawn from $EX(f', \mathcal{D})$. With this τ , $\log \frac{2s}{\tau} = \log(\frac{200s^2 r \log n}{\epsilon})$. If we assume $s \geq \log n$, we can show that choosing $r = 10 \log(s/\epsilon)$ will give $r \geq \log(\frac{200s^2 r \log n}{\epsilon})$.

It is not hard to see that the running time of this algorithm is at most $\text{oly}(n^r)$. This yields the following theorem:

Theorem 1 *For $s \geq \log n$, this algorithm learns s -term DNF under the uniform distribution in $n^{O(\log \frac{s}{\epsilon})}$ time.*

4 Acknowledgements

Bert Huang and Mathew Davies contributed to these notes.