Topics in Machine Learning Theory

Lecture 7: Boosting

Avrim Blum
09/24/14

Boosting

- A great practical algorithm
- A great theoretical result about basic definitions in the PAC model.
- A surprising connection between topics in online and distributional learning.

PAC learning and Weak learning

- Def 1: Alg A PAC-learns class C if for any $c \in C$, any distribution D, any $\epsilon, \delta > 0$, A produces a hypothesis of error $\leq \epsilon$ with prob $\geq 1 - \delta$.
- Def 2: Alg A Weak-learns class C if for any $c \in C$, any distribution D, there exists $\gamma, \tau > 1/poly(n)$ s.t. A produces a hyp of error $\leq \frac{1}{2} - \gamma$ with prob $\geq \tau$.
  - In other words, A has a non-negligible chance of doing non-negligably better than random guessing.

PAC learning and Weak learning

- Def 1: Alg A PAC-learns class C if for any $c \in C$, any distribution D, any $\epsilon, \delta > 0$, A produces a hypothesis of error $\leq \epsilon$ with prob $\geq 1 - \delta$.
- Def 2: Alg A Weak-learns class C if for any $c \in C$, any distribution D, there exists $\gamma, \tau > 1/poly(n)$ s.t. A produces a hyp of error $\leq \frac{1}{2} - \gamma$ with prob $\geq \tau$.
- Suppose we defined the PAC model using Def 2. Would this change the notion of what is efficiently learnable and what is not?
- Ans: No.

PAC learning and Weak learning

- Def 1: Alg A PAC-learns class C if for any $c \in C$, any distribution D, any $\epsilon, \delta > 0$, A produces a hypothesis of error $\leq \epsilon$ with prob $\geq 1 - \delta$.
- Def 2: Alg A Weak-learns class C if for any $c \in C$, any distribution D, there exists $\gamma, \tau > 1/poly(n)$ s.t. A produces a hyp of error $\leq \frac{1}{2} - \gamma$ with prob $\geq \tau$.
  - Given any alg that satisfies Def 2, can "boost" it to an algorithm that satisfies Def 1. This was the weak⇒strong learning result of Schapire.
  - Later turned into very practical algorithm AdaBoost by Freund and Schapire.

PAC learning and Weak learning

- Def 1: Alg A PAC-learns class C if for any $c \in C$, any distribution D, any $\epsilon, \delta > 0$, A produces a hypothesis of error $\leq \epsilon$ with prob $\geq 1 - \delta$.
- Def 2: Alg A Weak-learns class C if for any $c \in C$, any distribution D, there exists $\gamma, \tau > 1/poly(n)$ s.t. A produces a hyp of error $\leq \frac{1}{2} - \gamma$ with prob $\geq \tau$.
  - Note: can handle $\tau \Rightarrow \delta$ easily: just repeat $\frac{1}{\tau}\log\left(\frac{1}{\delta}\right)$ times and whp at least one was successful. Then draw fresh data and use to pick out the good one.
  - The real issue is $\gamma \Rightarrow \epsilon$. From now on, we'll ignore δ and assume that each time we get a hyp of error $\leq \frac{1}{2} - \gamma$.

Boosting: discussion

- We're going to prove this in a very constructive way.
  - Given a weak-learning algorithm A, we will view it as a black box, feeding in different distributions, and either boost up its accuracy as much as we like or else find a distrib D where error $> \frac{1}{2} - \gamma$.
  - As a practical matter, can think of boosting procedure as a way of creating good "challenge distributions".

An easy case: algorithms that know when they don't know

- Suppose A produces a hypothesis that on any given x either makes a prediction or says "not sure".
  - Always correct when it predicts.
  - Says "not sure" at most $1 - \epsilon'$ fraction of time. (It's trivial to do this for $\epsilon' = 0$).
- In this case can boost using a decision list.
  - Run A on D to get h_1 and put at top of DL.
  - Run A on $D|_{h_1(x)=not\ sure}$ and get h_2, etc.
  - Just need to continue for $O\left(\frac{1}{\epsilon'}\log\left(\frac{1}{\epsilon}\right)\right)$ runs.

An easy case: algorithms that know when they don't know

- Basic idea: focus on where previous hypotheses had trouble. Force next one to learn something new.
- We will use this in the general case in the AdaBoost algorithm, but it won't be so simple.

AdaBoost preliminaries

- Will be most convenient to draw a sample S and then do our work on distributions defined over S.
- Let's assume A chooses h's from class C with $C[m] = O(m^d)$. Our final rule will be from larger class H with $H[m] = O(m^{O\left(\frac{1}{\gamma^2}\log\left(\frac{1}{\epsilon}\right)d\right)})$.
- So, just draw S sufficiently large to get uniform convergence. Can now focus on performance on S.
- Onto the board for the rest of the discussion….