Online Optimization Notes

Stephen J. Barr
ISOM
Foster School of Business
University of Washington
email: stevejb@uw.edu

February 13, 2013

1 Online Learning

Also called "learning from experts" or "experts problem". An example problem would be stock picking.

Setup:

- N experts
- Algorithm picks an expert
- $\forall$ expert i, he receives a payoff of $V_i \in [0, 1]$
- Goal: maximize the total payoff of the alogrithm

1.1 Differences from Online Matchine

- We make a choice *before* we see the payoffs, rather than in online matching where we see the weights first
- In online matching, there are global contraints which constrain what we pick accross time. In Online Learning, there are only local constraints (pick 1 expert per round)

1.2 Analysis

`OPT` := optimum solution in hindsight. Every round pick the best expert.

1.2.1 Deterministic Algorithm

Assume there are 2 experts. Let Algo. pick 1.

- In the **worst** case, $V_1 = 0$. In the **worst** repeated case, we pick the worst expert every time and get $V_1 = 0$ every time.

1.2.2 Randomized Algorithm

- Algo pick between experts 1 and 2 where $p_1 = p_2 = \frac{1}{2}$.
- $V_1 = 0, V_2 = 2$, $\texttt{ALG} = \frac{1}{2}$, $\texttt{OPT} = 1$

With N experts,

- algo picks expert i w.p. p_i
- In the worst case, everyone is 0 except for the expert i with smallest p_i.
- In this case, $\texttt{ALG} = \frac{1}{n}$ whereas $\texttt{OPT} = 1$.

What is happening is that OPT is too powerful, since it can pick the best one every single time. Thus there is no way to compete with it. So what we do is compete with a weakened OPT. We constrain OPT to "stick to one expert". Thus, this constrained OPT does not have foresight.

$$\texttt{OPT} = \max_i \left\{ \sum_{t=1}^{T} V_{i,t} \right\} \tag{1}$$

Denote:

$$w_i = \sum_{t=1}^{T} V_{i,t} \tag{2}$$

where w_i is how well expert i does over the period T.

We define regret

$$\text{Regret} = \texttt{OPT} - \texttt{ALG}. \tag{3}$$

Thus, we have an a-priori bound on opt. If $\texttt{ALG} = 0$, then $\texttt{OPT} = T$. What we want is $\text{Regret}(T) = o(T)$.

$$\frac{\text{regret}(T)}{T} \to 0 \text{ , as } T \to \infty \tag{4}$$

$$\frac{\texttt{ALG}}{T} \to \frac{\texttt{OPT}}{T} \text{as } T \to \infty \tag{5}$$

Define

$$\phi(w_1, ..., w_n) = \frac{1}{\lambda} \log \sum_i e^{\lambda w_i} \tag{6}$$

where the interpretation is that it is a "smooth approximation to max" where λ controlls the smoothness and the error.

Lemma 1. $\phi \geq \texttt{OPT}$ ***Proof:*** $\frac{1}{\lambda} \log \sum_i e^{\lambda w_i} \geq \frac{1}{\lambda} \log \left[\max_i e^{\lambda w_i}\right] = \frac{1}{\lambda} \log e^{\lambda \texttt{OPT}} = \texttt{OPT}$

The idea is that as $\lambda \uparrow$, error $\downarrow$.

Lemma 2. $\phi \leq \texttt{OPT} + \frac{1}{\lambda} \log n$. ***Proof:*** $\frac{1}{\lambda} \log \sum_i e^{\lambda w_i}$. *We now replace each term inside by the max.* $\leq \frac{1}{\lambda} \log \left[\sum_i e^{\lambda \texttt{OPT}}\right] = \frac{1}{\lambda} \log(n e^{\lambda \texttt{OPT}} = \frac{1}{\lambda} \log n + \texttt{OPT}$.

Now, think of $\phi(\cdot)$ as the objective function and try to optimize $\phi(\cdot)$. Why can we not pick a huge λ for tiny error? Because the smoothness goes down.

Since $\phi = \mathbb{R}^n \to \mathbb{R}$, $\Delta\phi : \mathbb{R}^n \to \mathbb{R}$. $\Delta \to = \left(\frac{\partial\phi}{\partial w_1}, ..., \frac{\partial\phi}{\partial w_n}\right)$. Integrating the value of the gradient from $t = 0$ to $t = 1$ of $w(t)$

Theorem 1. *$w : [0, 1] \to \mathbb{R}^n$.*

$$\int_{t=0}^{t=1} < \Delta\phi, \frac{dw}{dt} > dt = \phi(w(1)) - \phi(w(0))$$

By the chain rule

$$\frac{d\phi}{dt} = sum_i \frac{\partial\phi}{\partial w_i}\frac{dw_i}{d_t} =< \Delta\phi, dw/dt >$$

The LHS $= \int_{t=0}^{t=1} \frac{d\phi}{dt} dt = \phi(w(1)) - \phi(w(0))$.

Suppose we made a "jump".

Theorem 2.

$$< \Delta\phi(w(0)), w(1) - w(0) >\geq \phi(w(1)) - \phi(w(0)) - \lambda$$

if $||w(1) - w(0)||_\infty \leq 1$ *(bounding the step size to be less than* 1.

Note: Since ϕ is convex, $< \Delta\phi(w(0)), w(1) - w(0) > \leq \phi(w(1)) - \phi(w(0))$.

To make this precise, we first define a lemma that sees the gradient as a probability distribution:

Lemma 3. *Gradient as a probability distribution.* $\sum_i \Delta_i\phi(w) = 1$. ***Proof:*** $\frac{\partial\phi}{\partial w_i} = \frac{1}{\lambda}\frac{1}{\sum_{j=1}^n e^{\lambda w_j}}\lambda e^{\lambda w_i} = \frac{1}{\sum_{j=1}^n} \sum_i e^{\lambda w_i} = 1$.

Given the above lemma, we can define the algorithm:

$$w_{i,t} = \sum_{t'=1}^{t} v_{i,t'}$$

and denote vector $\vec{w_t} = (w_{it})_{i=1,...,n}$ and $w_0 = (0, ..., 0)$.

- In round t, pick expert i w.p. $\frac{\partial\phi}{\partial w_i}(\vec{w_{t-1}})$.
- $\vec{w_t} = \vec{w_{t-1}} + \vec{v_t}$

$$E[\texttt{ALG}] = \sum_{t=1}^{T} < \Delta\phi(\vec{w_{t-1}}, \vec{v_t} >\geq \sum_{t=1}^{T}[\phi(\vec{w_t}) - \phi(\vec{w_{t-1}}) - \lambda] = \phi(\vec{w_T}) - \phi(\vec{w_0}) - \lambda T \tag{7}$$

Thus

$$\text{regret} = \texttt{OPT} - \texttt{ALG} \leq \phi(\vec{w_T}) - \texttt{ALG} \leq \phi(\vec{w_0}) + \lambda T = \frac{1}{\lambda}\log n + \lambda T \tag{8}$$

We want to minimize Eq. (8) so that $\frac{1}{\lambda}\log n = \lambda T \Rightarrow \lambda^2 = \frac{\log n}{T}$. $\lambda = \sqrt{\log n/T}$, and thus regret $= 2\sqrt{T \log n}$.

1.2.3 Proof of Thm. 2:

$\phi(w(1))-\phi(w(0)) = \frac{1}{\lambda}\log\sum_i e^{\lambda w_{1,i}} - \frac{1}{\lambda}\log\sum_i e^{\lambda w_{0i}} = \frac{1}{\lambda}\log\left[\frac{\sum_i e^{\lambda w_{1i}}}{\sum_i e^{\lambda w_{0i}}}\right] = \frac{1}{\lambda}\log\left[\frac{\sum_i e^{\lambda w_{1i}+\lambda\Delta w_i}}{\sum_i e^{\lambda w_{0i}}}\right]$. Note that $\sum_i \mu_i = 1$, so we can take a convex combination of λ_i's. $= \frac{1}{\lambda}\log[\sum_i \mu_i e^{\lambda_i}] = e^{\lambda[\phi(w(1))-\phi(w(0))]} = \sum_i \mu_i e^{\lambda_i}$.

How are we going to use this? Consider the line going from 0 to λ_i. $\sum_i \mu_i\lambda_i = \hat{\lambda} \leq \lambda$ $= 1 + \frac{e^{\lambda}-1}{\lambda}\hat{\lambda}e^{\lambda^2}e^{\hat{\lambda}}$.

$< \Delta\phi(\vec{w_0}), \Delta w >= \sum_i \mu_i\lambda_i/\lambda = \frac{\hat{\lambda}}{\lambda}$. So we get $e^{\lambda[\phi(w(1))-\phi(w(0))]} \leq e^{\lambda^2}e^{\hat{\lambda}}$. $\phi(w(1)) - \phi(w(0)) \leq \lambda + \frac{\hat{\lambda}}{\lambda}$.

We claim: If $\lambda > 1$, then the slope of e^{λ} at $\lambda = e^{\lambda}$, which is greater than the slope of the line from (0,1) to (λ, e^{λ}). If $\lambda < 1$, then $e^{\lambda} \leq 1 + \lambda + \lambda^2$, so $\frac{e^{\lambda}-1}{\lambda} \leq 1 + \lambda$ so $1 + ((e^{\lambda-1})/\lambda) * \hat{\lambda} = 1 + \hat{\lambda} + \lambda\hat{\lambda} \leq e^{\lambda+\lambda\hat{\lambda}} \leq e^{\hat{\lambda}+\lambda^2}$. □

1.3 Application of Online Matching to "boosting"

Suppose we have a function that we want to learn. $f : X \to \{-1, +1\}$. f is "is this a cat video?". We do not know what this function looks like, but we want to approximate it by a nice function. $H :=$ hypothesis class. There is a prob. distribution D over X. We want to approximate f with a function in H, where $E_D[H] \approxeq E_D[f]$.

strong learning: $\forall D$, finds an $h \in H$ s.t.

$$P_{x\in D}[f(x) = h(x)] \geq 1 - \epsilon \tag{9}$$

weak learning: It is similar. For all distributions, it finds an H, but the probability

$$P_{x\in D}[f(x) = h(x)] \geq \frac{1}{2} + \delta \tag{10}$$

This means tha we are doing *slightly better* than random guessing.

What "boosting" does is takes a weak learning algorithm and converts it into a strong learning algorithm. D is uniform distribution on X, where X is a training set. We know $f(x)\forall x \in X$.

Now consider $X =$ experts. Repeat for $t = 1, .., T$

- We want a MWU algorithm D_t over X
- Use WL to get $h_t \in H$ s.t. $P_{x\in D_t}[h_t(x) = f(x)] \geq \frac{1}{2} + \delta$.
- $V_{x,t} = 1\!\!\not\,\{h_t(x) \neq f(x)\}$.

This is using the weak learning algorithm on many different distributions. $SL = h(x) = sgn(\sum_t h_t(x)) \equiv$ majority.