On-line Algorithms in Machine Learning

[Portions taken from presentation given at 1998 School and Workshop on On-line Algorithms, Udine, Italy, and from CS Colloquium "Online Algorithms for Fun and Profit" at Cornell]

Machine Learning

Machine Learning: improve performance at a task based on experience.

- E.g., prediction, games, etc.
- Can do in batch or on-line.
- We'll be talking about on-line learning.

Plan for today:

- Introduce "Picking a winner" ("combining expert advice") problem.
- Exponential weights (weighted majority) algorithm
- Relations to min-max theorem, MTS problem, combining online algs online.

How to choose a job and decide when to switch?

- Say you are considering joining one of n startup companies.

- You watch their performance for a while and then join up with one.

- Of course, you only make money based on performance *while you're there*. And, past performance is no guarantee of future results.

- If your company is doing badly, you can switch to another. Of course, past performance is no guarantee of future results.

- What's a good strategy?

An idealized model

- n buckets.
- You are standing in one of them.
- At each time step, a ball falls into one of the buckets. If it's your bucket, you get \$1.
- Then you can choose to move if you want.
- Game ends when fullest bucket has d balls.

What can we say:

→ If we knew what was going to happen in advance, we could make \$$d$.

→ Bad news: No matter what we do, it is possible for us to make \$0.

→ Question: Can we pick a randomized strategy such that for any sequence of events, our expected gain is pretty good?

This is like assumption that the sequence of balls is preordained but unknown. Doesn't depend on where you are.

Strategies that don't work

- Pick a random company and stay there.

- Go to first company that gets $\$d/2$.

- Go to company (bucket) that has done best so far. Break ties at random.

But, it turns out there's a simple strategy with the following amazing guarantee:

- For any sequence of events σ, your expected gain is at least $d - \frac{3}{2}\sqrt{d \log n}$.

Exponential-Weights Algorithm [LW]

Give each bucket a "weight" starting at 1. Then repeat:

- Go to bucket i with probability w_i/W, where $W = \sum_i w_i$.
- When ball falls into some bucket, multiply its weight by $1 + \epsilon$.

E.g., what happens if all fall into one bin?

Analysis (tricky, tricky)

- Let W_t be total weight at time t.
- Let F_t be fraction of weight on bucket that ball falls in at time t.
- So, $W_{t+1} = W_t + \epsilon W_t F_t = W_t(1 + \epsilon F_t)$.
- $W_{\text{Final}} = n \prod_t (1 + \epsilon F_t)$.
- Our expected gain is $\sum_t F_t$.
- Note: $W_{\text{Final}} \geq (1 + \epsilon)^d$.

Now here's the tricky part: take logs....

Analysis, contd.

- Our expected gain is $\sum_t F_t$.
- $(1+\epsilon)^d \leq W_{\mathsf{Final}} = n \prod_t (1+\epsilon F_t)$.

Take logs:

- $d \ln(1+\epsilon) \leq \ln n + \sum_t \ln(1+\epsilon F_t)$.
- We know $\ln(1+x) \leq x$, so:

$$\begin{aligned} d \ln(1+\epsilon) &\leq \ln n + \sum_t \epsilon F_t \\ &= \ln n + \epsilon[\text{Our Expected Gain}]. \end{aligned}$$

- Using $\ln(1+\epsilon) \geq \epsilon - \epsilon^2/2$, we get:

$$\boxed{\text{Our expected gain} \geq d\left(1 - \frac{\epsilon}{2}\right) - \frac{\ln n}{\epsilon}}$$

Now, plug in $\epsilon = \sqrt{(\ln n)/d}$ to get

$$d - \frac{3}{2}\sqrt{d \ln n}$$

Generalizations

Suppose each step we get a gain vector $[g_1, \ldots, g_n]$ where $0 \leq g_i \leq 1$.

- Alg is to multiply weight i by $(1+\epsilon)^{g_i}$.
- Can show that same result holds:

$$\mathbf{E}[\text{Alg gain}] \geq (1-\epsilon/2)[\text{Best bucket}] - \frac{1}{\epsilon}\ln n$$

- In ML, view as choosing among predictions of multiple "experts". Losses rather than gains:

$$\mathbf{E}[\text{Alg loss}] \lesssim (1+\epsilon/2)[\text{best expert}] + \frac{1}{\epsilon}\ln n.$$

Details (gains):

- If weights were $[w_1, \ldots, w_n]$, then
- $F = \mathbf{E}[\text{Our gain}] = \frac{1}{W}(w_1 g_1 + \ldots + w_n g_n)$.
- New weights: $[w_1(1+\epsilon)^{g_1}, \ldots, w_n(1+\epsilon)^{g_n}]$.

  $\leq [w_1(1+\epsilon g_1), \ldots, w_n(1+\epsilon g_n)]$ since $g_i \leq 1$.
- so, new sum of weights $\leq W(1+\epsilon F)$.

Connection to MTS

- 0-step vs 1-step lookahead.
- Movement costs
- What are we comparing to?

Movement costs not so bad in oblivious adversary model.

E.g.,

$\mathbf{p}$	1/4	1/4	1/4	1/4
ℓ	1	1	0	0
$\mathbf{p}'$	1/8	1/8	1/4	1/2

In experts setting, we get:

$$\mathbf{E}[\text{Alg loss}] \lesssim (1+\varepsilon/2)[\text{best expert}] + \frac{1}{\epsilon}\log n.$$

In uniform MTS setting (distance $= d$),

$$\mathbf{E}[\text{Alg cost}] \lesssim (1+3\varepsilon/2)[\text{best state}] + \frac{d}{\epsilon}\log n.$$

Neat application: Combining online algorithms online.

Combining online algorithms online

Say we have n paging algorithms.

View online algs as "experts" and do nearly as well as best in hindsight?

- Experts have state. Loss is f(state).
- To switch from following expert X to Y need (at least lazily) to switch state too.

Can view as changing metric space, but always diameter $\leq k$. Apply EW to get:

$$\mathbf{E}[\text{cost}] \lesssim (1 + 3\varepsilon/2)[\text{best alg}] + \frac{1}{\epsilon} k \log n.$$

Which is better: Marking or EW?

Bad case for Marking:

$$\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ . & . & . & \end{array}$$

Bad case for EW:

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ . & . & . & \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ . & . & . & \\ 0 & 0 & 1 & 1 \end{array}$$

Weight-sharing and randomized WF algs give "best of both".

Best of both worlds

Weight-sharing version of EW: take half of weight removed from system and distribute evenly among experts.

- In "experts" setting, for all k,

$$\mathbf{E}[\text{Alg loss}] \lesssim (1+\varepsilon)L_k + (\frac{1}{\epsilon}\log n)(k+1)$$

  where L_k is loss of best strategy in hindsight that switches k times.

- In uniform MTS setting, can view as:

$$\mathbf{E}[\text{Alg cost}] \leq O(1) \times \text{OPT}_{local} + O(\log n) \times \text{OPT}_{move}.$$

Last topic: The min-max theorem

Consider a 2-player zero-sum game:

Payoff to row player:

	N	D
N	-5	5
D	10	-10

Suppose that for any (randomized) strategy of your opponent, there exists a deterministic counter-strategy for you that guarantees you an expected gain $\geq V$.

Then, there exists a randomized strategy for you such that for any counter-strategy of the opponent, you get an expected gain $\geq V$.

We will prove minimax for matrices where values are all in range $[0, 1]$. Why is this enough?

Proof of min-max theorem

Consider the following strategy for the row player:

- Create one bucket for each row.
- Pick a bucket (row) using our exponential-weights algorithm.
- Opponent's play gives us the gain vector.

What would it mean for min-max to be false?

$\rightarrow$ If we know opponents randomized strategy, we can get expected gain $\geq V$, but if we have to choose our randomized strategy first, then opponent can force us to get $\leq V - \delta$.

This contradicts our bound if we use $\epsilon = \delta$. Our gain per game is approaching OPT$(1 - \epsilon/2)$.

In other words

If there was a gap (V versus $V - \delta$), then for *any* randomized strategy we choose, opponent who knows our strategy should be able to force us to get no more than $V - \delta$ on average per play.

- But, in T games, OPT row in hindsight gets total of at least VT
- We get at least $VT(1 - \epsilon/2) - \frac{\ln n}{\epsilon}$.
- Per play, we are getting $V(1 - \epsilon/2) - o(1)$.