Online learning, minimizing regret, and combining expert advice.

Maria Florina (Nina) Balcan

11/12/2018

- "The weighted majority algorithm" N. Littlestone & M. Warmuth
- "Online Algorithms in Machine Learning" (survey) A. Blum

Motivation

Many situations involve repeated decision making in an uncertain environment.

- Deciding how to invest your money (buy or sell stocks)
- What route to drive to work each day
- Playing repeatedly a game against an opponent with unknown strategy

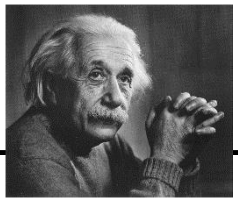
We will study:

Learning algos for such settings with connections to game theoretic notions of equilibria

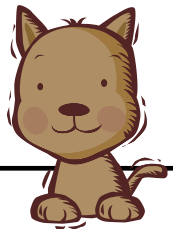
Online learning, minimizing regret, and combining expert advice.

Expert 1

Expert 2



Expert 3



Using "expert" advice

Assume we want to predict the stock market.

- We solicit n "experts" for their advice.
  - Will the market go up or down?
- We then want to use their advice somehow to make our prediction. E.g.,

Expt 1	Expt 2	Expt 3	neighbor's dog	truth
down	up	up	up	up
down	up	up	down	down
...	...	...	...	...

Can we do nearly as well as best in hindsight?

Note: "expert" $\equiv$ someone with an opinion.
[Not necessairly someone who knows anything.]

Formal model

- There are n experts.
- For each round $t=1,2, \ldots, T$
  - Each expert makes a prediction in $\{0,1\}$
  - The learner (using experts' predictions) makes a prediction in $\{0,1\}$
  - The learner observes the actual outcome. There is a mistake if the predicted outcome is different form the actual outcome.

    The learner gets to update his hypothesis.

Can we do nearly as well as best in hindsight?

Formal model

- There are n experts.
- For each round $t=1,2, \ldots, T$
  - Each expert makes a prediction in $\{0,1\}$
  - The learner (using experts' predictions) makes a prediction in $\{0,1\}$
  - The learner observes the actual outcome. There is a mistake if the predicted outcome is different form the actual outcome.

Can we do nearly as well as best in hindsight?

We are not given any other info besides the yes/no bits produced by the experts. We make no assumptions about the quality or independence of the experts.

We cannot hope to achieve an absolute level of quality in our predictions.

Simpler question

- We have n “experts”.
- One of these is perfect (never makes a mistake).
  We don’t know which one.
- A strategy that makes no more than lg(n) mistakes?

Halving algorithm

Take majority vote over all experts that have been correct so far.

I.e., if # surviving experts predicting 1 > # surviving experts predicting 0, then predict 1; else predict 0.

Claim: If one of the experts is perfect, then at most lg(n) mistakes.

Proof: Each mistake cuts # surviving by factor of 2, so we make $\leq$ lg(n) mistakes.

Note: this means ok for n to be very large.

Using "expert" advice

- If one expert is perfect, get $\leq$ lg(n) mistakes with halving algorithm.
- But what if none is perfect? Can we do nearly as well as the best one in hindsight?

Using "expert" advice

Strategy #1: Iterated halving algorithm.

- Same as before, but once we've crossed off all the experts, restart from the beginning.
- Makes at most $\log(n)*[OPT+1]$ mistakes, where OPT is #mistakes of the best expert in hindsight.

  Divide the whole history into epochs. Beginning of an epoch is when we restart Halving; end of an epoch is when we have crossed off all the available experts.

  At the end of an epoch we have crossed all the experts, so every single expert must make a mistake. So, the best expert must have made a mistake. We make at most log n mistakes per epoch.

- If $OPT=0$ we get the previous guarantee.

Using "expert" advice

Strategy #1: Iterated halving algorithm.

- Same as before, but once we've crossed off all the experts, restart from the beginning.
- Makes at most log(n)*[OPT+1] mistakes, where OPT is #mistakes of the best expert in hindsight.

Wasteful. Constantly forgetting what we've "learned".

Can we do better?

Weighted Majority Algorithm

Key Point:

A mistake doesn't completely disqualify an expert.

Instead of crossing off, just lower its weight.

Weighted Majority Algorithm

- Start with all experts having weight 1.
- Predict based on weighted majority vote.
  - If $\sum_{i:x_i=1} w_i \geq \sum_{i:x_i=0} w_i$ then predict 1

    else predict 0

Weighted Majority Algorithm

Key Point:

A mistake doesn't completely disqualify an expert.

Instead of crossing off, just lower its weight.

Weighted Majority Algorithm

- Start with all experts having weight 1.
- Predict based on weighted majority vote.
- Penalize mistakes by cutting weight in half.

					prediction	correct
weights	1	1	1	1		
predictions	Y	Y	Y	N	Y	Y
weights	1	1	1	.5		
predictions	Y	N	N	Y	N	Y
weights	1	.5	.5	.5		

Analysis: do nearly as well as best expert in hindsight

Theorem: If M = # mistakes we've made so far and OPT = # mistakes best expert has made so far, then:

$$M \leq 2.4(OPT + \lg n)$$

Analysis: do nearly as well as best expert in hindsight

Theorem: If M = # mistakes we've made so far and OPT = # mistakes best expert has made so far, then:

$$M \leq 2.4(OPT + \lg n)$$

Proof:

- Analyze W = total weight (starts at n).
- After each mistake, W drops by at least 25%.

  So, after M mistakes, W is at most $n(3/4)^M$.

- Weight of best expert is $(1/2)^{OPT}$. So,

$$\begin{aligned}(1/2)^{OPT} &\leq n(3/4)^M \\ (4/3)^M &\leq n2^{OPT} \\ M &\leq \underbrace{2.4}_{\text{constant ratio}}(OPT + \lg n)\end{aligned}$$

Randomized Weighted Majority

2.4(OPT + lg n) not so good if the best expert makes a mistake 20% of the time.

Can we do better?

- **Yes.** Instead of taking majority vote, use weights as probabilities & predict each outcome with prob. ~ to its weight. (e.g., if 70% on up, 30% on down, then pick 70:30)

Key Point: smooth out the worst case.

Randomized Weighted Majority

2.4(OPT + lg n) not so good if the best expert makes a mistake 20% of the time.

Can we do better?

- **Yes.** Instead of taking majority vote, use weights as probabilities. (e.g., if 70% on up, 30% on down, then pick 70:30)

- Also, generalize $\frac{1}{2}$ to $1-\varepsilon$.

Equivalent to select an expert with probability proportional with its weight.

Randomized Weighted Majority

Initially: $w_1^i = 1$ and $p_1^i = 1/n$, for each expert i.

At time t:

See experts predicions. Select an expert i with probability p_t^i.

See correct answer; induces loss vector ℓ_t.

Update the weights.

If $\ell_t^i = 1$, then let $w_{t+1}^i = w_t^i(1-\epsilon)$;

else ($\ell_t^i = 0$) let $w_{t+1}^i = w_t^i$.

Let $p_{t+1}^i = w_{t+1}^i / W_{t+1}$, where $W_{t+1} = \sum_{i \in X} w_{t+1}^i$

Formal Guarantee for Randomized Weighted Majority

Theorem: If M = expected # mistakes we've made so far and OPT = # mistakes best expert has made so far, then:

$$M \leq (1+\varepsilon)OPT + (1/\varepsilon)\log(n)$$

Analysis

- Key idea: if the algo has significant expected loss, then the total weight must drop substantially.

- Say at time t we have fraction F_t of weight on experts that made mistake.

  i.e., $F_t = (\sum_{i:l_t^i=1} w_t^i)/W_t$

- For all t, $W_{t+1} = W_t(1 - \epsilon F_t)$

- F_t is our expected loss at time t; probability we make a mistake at time t.

Analysis

- Say at time t we have fraction F_t of weight on experts that made mistake.
- So, we have probability F_t of making a mistake, and we remove an εF_t fraction of the total weight.
  - $W_{final} = n(1-\varepsilon F_1)(1 - \varepsilon F_2)\ldots$
  - $\ln(W_{final}) = \ln(n) + \sum_t [\ln(1 - \varepsilon F_t)] \leq \ln(n) - \varepsilon \sum_t F_t$ (using $\ln(1-x) < -x$)

    $= \ln(n) - \varepsilon M.$ $(\sum F_t = E[\#\text{ mistakes}])$
- If best expert makes OPT mistakes, $\ln(W_{final}) > \ln((1-\varepsilon)^{OPT})$.
- Now solve: $\ln(n) - \varepsilon M > OPT \ln(1-\varepsilon)$.

$$M \leq \frac{-OPT \ln(1-\varepsilon) + \ln(n)}{\varepsilon} \approx (1+\varepsilon/2)OPT + \frac{1}{\varepsilon}\log(n)$$

Randomized Weighted Majority

Solves to:

$$M \leq \frac{-OPT \ln(1-\varepsilon) + \ln(n)}{\varepsilon} \approx (1+\varepsilon/2)OPT + \frac{1}{\varepsilon}\ln(n)$$

$$M \leq 1.39OPT + 2\ln n \quad \leftarrow \varepsilon = 1/2$$

$$M \leq 1.15OPT + 4\ln n \quad \leftarrow \varepsilon = 1/4$$

$$M \leq 1.07OPT + 8\ln n \quad \leftarrow \varepsilon = 1/8$$

Summarizing

- $E[\#\text{ mistakes}] \le (1+\varepsilon)OPT + \varepsilon^{-1}\log(n)$

- If set $\varepsilon=(\log(n)/OPT)^{1/2}$ to balance the two terms out (or use guess-and-double), get bound of

  - $E[\text{mistakes}] \le OPT + 2(OPT \cdot \log n)^{1/2}$

Note: Of course we might not know OPT, so if running T time steps, since $OPT \le T$, set ϵ to get additive loss $(2T \log n)^{1/2}$

  - $E[\text{mistakes}] \le OPT + 2(T \cdot \log n)^{1/2}$ regret

- So, $regret/T \to 0$. [no regret algorithm]

What if have n options, not n predictors?

- We're not combining n experts, we're choosing one.
- Can we still do it?

- Nice feature of RWM: can be applied when experts are n different options
    - E.g., n different ways to drive to work each day, n different ways to invest our money.

Randomized Weighted Majority

Initially: $w_1^i = 1$ and $p_1^i = 1/n$, for each expert i.

At time t:

See experts predicions. Select an expert i with probability p_t^i.

See correct answer; induces loss vector ℓ_t.

Update the weights.

If $\ell_t^i = 1$, then let $w_{t+1}^i = w_t^i(1-\epsilon)$;

else $(\ell_t^i = 0)$ let $w_{t+1}^i = w_t^i$.

Let $p_{t+1}^i = w_{t+1}^i / W_{t+1}$, where $W_{t+1} = \sum_{i \in X} w_{t+1}^i$

Note: Did not see the predictions to select an expert (only needed to see their losses to update our weights)

What if have n options, not n predictors?

- We're not combining n experts, we're choosing one.
- Can we still do it?

- Nice feature of RWM: can be applied when experts are n different options
  - E.g., n different ways to drive to work each day, n different ways to invest our money.
- We did not see the predictions in order to select an expert (only needed to see their losses to update our weights)

Decision Theoretic Version; Formal model

- There are n experts.
- For each round $t=1,2, \ldots, T$
  - No predictions. The learner produces a prob distr. on experts based on their past performance p_t.
  - The learner is given a loss vector l_t and incurs expected loss $l_t \cdot p_t$.
  - The learner updates the weights.

The guarantee also applies to this model!!!

[Interesting for connections between GT and Learning.]

Can generalize to losses in [0,1]

- If expert i has loss l_i, do: $w_i \leftarrow w_i(1-l_i\varepsilon)$.

  [before if an expert had a loss of 1, we multiplied by (1-epsilon), if it had loss of 0 we left it alone, now we do linearly in between]

- Same analysis as before.

Summarizing

- $E[\#\text{ mistakes}] \le (1+\varepsilon)OPT + \varepsilon^{-1}\log(n)$
- If set $\varepsilon=(\log(n)/OPT)^{1/2}$ to balance the two terms out (or use guess-and-double), get bound of
  - $E[\text{mistakes}] \le OPT+2(OPT\cdot\log n)^{1/2}$

Note: Of course we might not know OPT, so if running T time steps, since $OPT \le T$, set ϵ to get additive loss $(2T \log n)^{1/2}$

  - $E[\text{mistakes}] \le OPT+\underbrace{2(T\cdot\log n)^{1/2}}_{\text{regret}}$

- So, $\text{regret}/T \to 0$. [no regret algorithm]

Lower Bounds

Consider $T < \log n$. $\exists$ a stochastic series of expert predictions and correct answers such that for any alg, we have $E[\#\text{mistakes}]=T/2$ and yet the best expert makes 0 mistakes.

- At $t=1$, half the experts predict 0, half predict 1. Correct answer given by fair coin flip.
- At $t=2, 3, \ldots$: of experts correct so far, half predict 0 and half predict 1. Correct answer given by fair coin flip.
- Any online algorithm incurs an expected loss of $1/2$ at each time step.
- Yet, for $T < \log n$ there will always be some expert with 0 mistakes.

Lower Bounds

Our second lower bound shows the dependence on sqrt(T) is needed. Consider n=2.

- Expert 1 always predicts 0. Expert 2 always predicts 1. Correct answer given by fair coin flip.
- Any online algorithm has 50% chance of predicting correctly at each time step, so expected loss of T/2.
- E(loss of best expert) = E(min(#heads, #tails)) = T/2 - \Theta(\sqrt(T))

Summary

- Can use to combine multiple algorithms to do nearly as well as best in hindsight.
- Can apply RWM in situations where experts are making choices that cannot be combined.
  - E.g., repeated game-playing.
  - E.g., online shortest path problem
- Extensions:
  - Online linear and more generally convex optimization.

e.g., "Online Convex Programming and Generalized Infinitesimal Gradient Ascent", ICML 2003. Test of Time Award at ICML

Summary

- Can use to combine multiple algorithms to do nearly as well as best in hindsight.
- Can apply RWM in situations where experts are making choices that cannot be combined.
  - E.g., repeated game-playing.
  - E.g., online shortest path problem

- Extensions:
  - "bandit" problem.
  - efficient algs for some cases with many experts.
  - Sleeping experts / "specialists" setting.