Admin:

→ Final exam: Monday, 5/16, 1:30-4:30 PM, Johnson track
→ Covers everything up to last Friday recitations (this and Thursday's lecture not included)
→ Review recitations tomorrow (no rec. on Fri)
→ Problem solving session - led by instructors
  - Thursday, 5/12, 2-3 PM, 26-100
  - Pizza provided!
→ Practice (on past exams)!

→ Course evaluations due this Sunday!
  http://web.mit.edu/subject.evaluation
→ Tell us what you love (or hate) about this class!

Today:

→ How to get rich (if you have good advice)
→ Learning from expert advice framework
→ Regret minimization
→ Multiplicative weights update method

Motivation:

→ Getting rich (sorry!)
→ Prediction based on unreliable (or even adversarially altered) information
→ Think: Extracting valuable information out of the ocean of noisy data
  ⇒ AKA job of our analyst
(Simplistic) Stock Market Model:

Let \( X_t \) be evolving index (think: the value of NASDAQ at the closing on day \( t \)).

Fix \( X_0 = 0 \) and each day (round) \( t = 1, \ldots, T \).

Either:
1. \( \uparrow \) \( X_t = X_{t-1} + 1 \) ("market goes up")
2. \( \downarrow \) \( X_t = X_{t-1} - 1 \) ("market goes down")

Our task: Predict at the beginning of each day \( t \), whether it will be \( \uparrow \) or \( \downarrow \).
- If predicted correctly: gain \( \$1 \)
- Otherwise: lose \( \$1 \)

Not hard to see: No information about the future.

\( \Rightarrow \) Expected payoff is \( \boxed{0} \)

(The best we can do is to go with the 50-50 choice)

\( \Rightarrow \) Payoff of \( -T \) if \( X_t \) evolves in an adversarial manner

\( (T = \# \text{ of rounds}) \)
Need: n "experts" (or sources of information)

- At the beginning of each round t:
  Each "expert" i provides us with his/her prediction of whether \( \uparrow \) or \( \downarrow \) that day

- We produce our prediction based on this info

Challenge: Here: "expert" = someone with an opinion but not necessarily knowing what he/she is talking about

Think: financial "advisors"/talking heads on TV

\( \Rightarrow \) The advice might be contradictory, completely wrong, random, or even intentionally misleading!

Our goal: (the best we can hope for)

Make fairly good prediction provided at least one (initially unknown to us) expert is consistently providing good advice

Key question: How to identify this good expert?

(without making too many mispredictions)

Note: The identity of such best expert is clear only in hind sight
First approach: Always go with the majority prediction

\[ \Rightarrow \] Fails completely if all but one expert are always providing wrong prediction

Possible fix: Ignore predictions of experts that were already wrong in the past

Haldwain algorithm:

\[ \Rightarrow \] Start with all experts in the pool $S$ of "trustworthy" experts

\[ \Rightarrow \] In each round $t = 1, \ldots, T$:

\[ \Rightarrow \] Follow the majority prediction of experts in $S$ (ignore predictions of experts $\notin S$)

\[ \Rightarrow \] After seeing $X_t$, remove from $S$ all the experts that mispredicted

Observe: Each time this algorithm mispredicts

\[ \Rightarrow \] the size of $S$ at least doubles

\[ \Rightarrow \] If the best expert is perfect then the algorithm makes \( \leq \log n \) mistakes

Can show: Every (deterministic) algorithm needs to make \( \geq \log n \) mistakes in the worst-case (Exercise !)
What if the best expert is NOT perfect?

**Iterated halving algorithm:**

- Run the halving algorithm
- If the pool $S$ becomes empty:
  - Reset it by putting all the experts back in it

**Observe:** Whenever the pool $S$ becomes empty, every expert (including the best one) made a mistake

**Conclude:** The above algorithm makes at most

$$(m^*+1)(1+\log m)-1$$ mistakes,

where $m^*$ = # of mistakes of the best expert

**Can we do better?**

**Note:** In the above:

- Each time we reset $S$, we discard our knowledge of the past
- We do not differentiate between an expert with fewer mistakes and one who is wrong constantly
- Our "trust" is binary: either we trust fully or not at all
**Key idea:** Quantify "trustworthiness" of an expert via a weight

- Rely on advice of an expert proportionally to that weight

**Weighted majority algorithm:**

- Start with each expert having a weight $w_i = 1$
- In each round $t = 1, ..., T$:
  - Follow the weighted majority prediction
  - After seeing $x_t$, decrease the weight $w_i$ of each expert that was wrong by a factor of 2

**Claim:** This algorithm makes at most

$$m \leq 2.4(m^* + \log n)$$

**Proof:** Let $w_i^t = \text{weight of expert } i \text{ after round } t$

Consider potential function $W^t = \sum_i w_i^t$

Observe that $W^0 = m$

Also, whenever the algorithm makes a mistake

$$W^t \leq \frac{3}{4} W^{t-1}$$

$$W^T \leq \left(\frac{3}{4}\right)^m$$

(As at least half of the weight of the experts had to be wrong and their weights got halved)
On the other hand, if we look at the best expert i*:

\[
\omega_i^{T^*} = \left(\frac{1}{2}\right)^{m^*} \quad u_i^0 = \left(\frac{1}{2}\right)^{m^*}
\]

(\star\star)

Putting (*) and (\star\star) together:

\[
\left(\frac{1}{2}\right)^{m^*} = u_i^T \leq W \leq \left(\frac{3}{4}\right)^m
\]

Taking logarithms of both sides gives:

\[
m^* \log \frac{1}{2} \leq m \log \frac{3}{4} + \log m
\]

\[
\Rightarrow \quad m \leq \frac{1}{\log \frac{3}{4}} (m^* + \log m) \leq 2.4 \left( m^* + \log m \right)
\]

Can we do even better?

If weight decreases one by a factor of \((1+\varepsilon)\) (instead of \(2\)):

\[
m \leq (2+\varepsilon) \left( m^* + \frac{\log \frac{3}{4}}{\varepsilon} \right)
\]

Can show: The factor of 2 has to be there if the algorithm is deterministic.
How about using randomness?

Randomized weighted majority alg.:

Same as its deterministic variant, except to predict choose an expert at random with prob.

\[ P_i^t = \frac{\omega_i^{t-1}}{\sum_j \omega_j^{t-1}} \]  

(= prob. of choosing i proportional to his/her weight)

and follow his/her advice

Can show: For any \( T > 0 \), this algorithm makes at most

\[ E[m] \leq (1 + \epsilon)m^* + \frac{\ln m}{\epsilon} \]  

mistakes  

( in expectation)

\( \rightarrow \) This is optimal!

Note: Setting \( T \to \infty \) and \( \epsilon \to 0 \) (as an appropriate function of \( T \)) gives that \( m \to m^* \) (in the limit)
General framework: Learning from expert advice

\( \Rightarrow \) In "experts" (think of them as "strategies" now)

\( \Rightarrow \) We play \( T \) rounds of the following game:

\( \Rightarrow \) In each round \( t = 1, \ldots, T \)

- Choose a convex combination \( p^+_1, \ldots, p^+_m \) of experts
- Once we made our choice, a "loss" \( l^+_t \in [0,1] \) is revealed for each expert
- Our loss in round \( t \) is:

\[ l^+_t = \frac{1}{\varepsilon} p^+_t l^+_t \quad (\star) \]

\( \Rightarrow \) Goal: Minimize the total loss

\[ L^T = \sum_{t=1}^{T} l^+_t \]

\( \Rightarrow \) Specifically: Minimize regret \( R^T \)

\[ R^T = L^T - \min_{\text{optimal decision}} \sum_{t=1}^{T} l^+_t \]

\( l^* \) - loss of the best expert/strategy

Note: \( (\star) \) can be viewed as the expected loss if we choose to follow expert \( i \) with prob. \( p^+_i \)

Also: To recover our stock market model, just make the losses be

\[ l^+_t = \begin{cases} 1 & \text{if expert } i \text{ mispredicted } T/0 \text{ in round } t \\ 0 & \text{otherwise} \end{cases} \]
How to play this new game?

Adapt our randomized weighted majority strategy!

**Multiplicative Weights Update (MWU) algorithm:**

1. Start with $w_i^0 = 1$ for all experts.
2. In each round $t = 1, \ldots, T$:
   - Choose $p_i^t = \frac{w_i^{t-1}}{\sum_j w_j^{t-1}}$
3. After seeing all $L_i^t$, set
   
   $$w_i^t := (1 - \epsilon L_i^t) w_i^{t-1}$$

Can show: For any $\frac{1}{2}, \epsilon > 0$, and $T$

$$L^T \leq (1 + \epsilon) L^* + \frac{m \ln m}{\epsilon}$$

$\Rightarrow$ Setting $\epsilon = \sqrt{\frac{m \ln m}{T}}$ gives us

$$R^T \leq \sqrt{T \ln m}$$

$\Rightarrow$ "Average" regret:

$$\frac{R^T}{T} \leq \frac{\sqrt{T \ln m}}{T} \rightarrow 0$$

$\Rightarrow$ Asymptotically, we do as well as the best expert/strategy!
MUV algorithm has fundamental in:

- (continuous) optimization
- Economics (hedging, playing zero-sum games)
- Machine learning (combining simple predictors into sophisticated ones)

Intrigued? Take 6.854 to learn more!

Example (toy) application: Prediction of biased coin flips

- Consider a biased coin:
  - tails with prob. \( p \)
  - heads with prob. \( 1-p \)

- We do not know \( p \), but would like to predict the coin flips (almost) as well as if we did

How?

- Consider two experts/strategies
  - "always predict tails" \( \to \) optimal if \( p \geq \frac{1}{2} \)
  - "always predict heads" \( \to \) optimal if \( p < \frac{1}{2} \)

- Use MUV algorithm with these two experts and the losses penalizing the experts for mispredictions

\[ \Rightarrow \text{Our performance} \approx \text{performance of the best expert, i.e., the best prediction if } p \text{ known} \]