Online Prediction & Decision Making

CompSci 590.04

Instructor: Ashwin Machanavajjhala

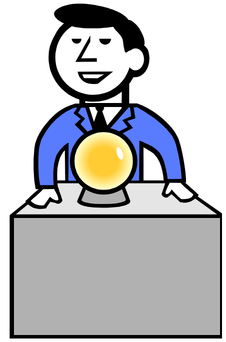
Duke UNIVERSITY

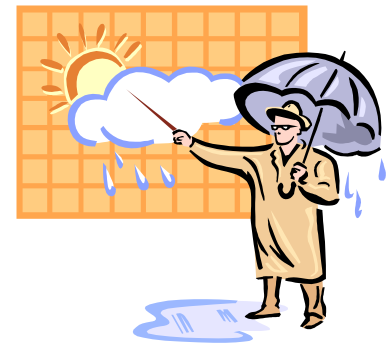
This Class

- Weighted Majority Algorithm
  - Multiple experts problem
- Follow the perturbed Leader
  - Online shortest paths
- Multi-armed bandit problems

Multiple Experts Problem

Will it rain today?





Truth = Yes	**Yes**	**Yes**	**Yes**	**No**
Truth = No	**Yes**	**No**	**No**	**Yes**
Truth = No	**Yes**	**Yes**	**No**	**No**

What is the best prediction based on these experts?

Multiple Experts Problem

- Suppose we know the best expert (who makes the least error), then we can just return that expert says.
  - This is the best we can hope for.

- We don't know who the best expert is.
  - But we can learn … we know whether it rained or not at the end of the day.

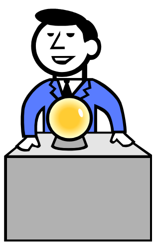
- Regret Minimization : number of mistakes made by our algorithms should be close to the number of mistakes made by the best expert.

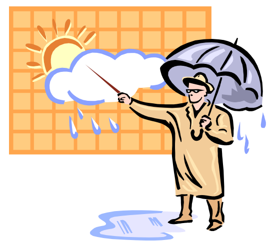
Weighted Majority Algorithm

[Littlestone&Warmuth ‘94]

“Experts”

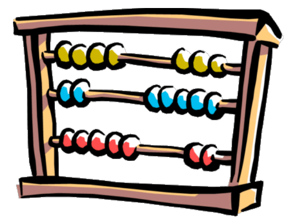
W_1 **W_2** **W_3** **W_4**



Y_1 **Y_2** **Y_3** **Y_4**

Algorithm



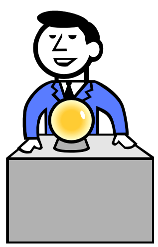
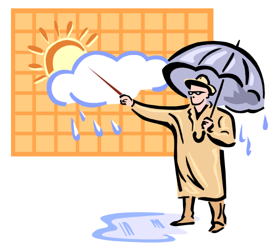
$$Is\ \frac{\sum_i w_i y_i}{\sum_i w_i} > \frac{1}{2}\ ?$$

Duke
UNIVERSITY

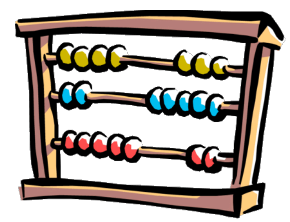
Weighted Majority Algorithm

[Littlestone&Warmuth ‘94]

“Experts”

1-ε	1-ε	1-ε	1
			
Yes	**Yes**	**Yes**	**No**

Algorithm



$$Is\ \frac{\sum_i w_i y_i}{\sum_i w_i} > \frac{1}{2}\ ?$$

Yes!

Truth

No

Duke UNIVERSITY

Weighted Majority Algorithm

- Maintain weights (or probability distribution) over experts.

Answering/Prediction:

- Answer using weighted majority, OR
- Randomly pick an expert based on current probability distribution. Use random experts answer.

Update:

- Observe truth.
- Decrease weight (or probability) assigned to the experts who are wrong.

Error Analysis

[Arora, Hazan, Kale '05]

Theorem:

After t steps,

- let m(t,j) be the number of errors made by expert j
- let m(t) be the number of errors made by algorithm
- let n be the number of experts,

$$\forall j, \qquad m(t) \leq \frac{2 \ln n}{\varepsilon} + 2(1+\varepsilon)m(t,j)$$

Duke UNIVERSITY

Error Analysis: Proof

- Let $\varphi(t) = \Sigma w_i$. Then, $\varphi(1) = n$.

- When the algorithm makes a mistake,
  $\varphi(t+1) \leq \varphi(t)\ (1/2 + ½(1-\varepsilon)) = \varphi(t)(1-\varepsilon/2)$
- When the algorithm is correct,
  $\varphi(t+1) \leq \varphi(t)$

- **Therefore,**
  $\boldsymbol{\varphi(t) \leq n(1-\varepsilon/2)^{m(t)}}$

Duke UNIVERSITY

Error Analysis: Proof

- $\varphi(t) \leq n(1-\varepsilon/2)^{m(t)}$

- Also, $W_j(t) = (1-\varepsilon)^{m(t,j)}$

- $\varphi(t) \geq W_j(t) \Rightarrow n(1-\varepsilon/2)^{m(t)} \geq (1-\varepsilon)^{m(t,j)}$

- Hence, $m(t) \geq 2/\varepsilon \ln n + 2(1+\varepsilon)m(t,j)$

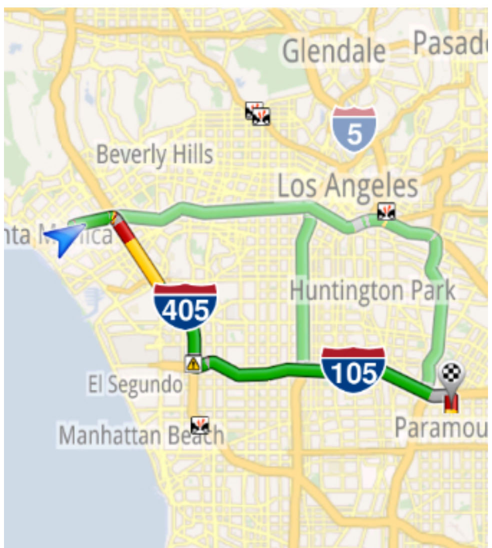
Application: Online Learning

- Mistake bound model
  - Algorithm receives an unlabeled example **x** (like our experts)
  - Algorithm predicts a classification of this example *p* (either -1 or +1)
  - Environment produces the correct answer *y* (either -1 or +1)

- Winnow algorithm
  - Learn a weight function **w** such that *sign*(**w x**) = *p*
  - Same as the Weighted Majority algorithm

Online Shortest Paths Problem

- Input: A directed graph G = (V,E), and a fixed pair of nodes (u,v)
- Each period (time t), we pick a path from u to v, and the length of the path is revealed.
- Cost at time t = length of chosen path.



Online shortest paths

- We could have used weighted majority, where each path is an expert
- But, number of paths (experts) is exponential

Follow the perturbed leader (FPL)

Randomized variant …

Initialization:

- Each expert *j* is assigned a cost *c(j, 0)* = 0

Prediction (time t):

- For each expert j select p(j, t) >= 0 from an exponential distribution ($\mu(x) \sim \varepsilon e^{-\varepsilon x}$)
- Make the same prediction as expert with smallest c(j, t) – p(j, t)

Update:

- If expert j's prediction is correct, c(j, t+1) = c(j, t)
- Else, c(j, t+1) = c(j,t) + 1

Duke UNIVERSITY

Online shortest paths

- We could have used weighted majority, where each path is an expert
- But, number of paths (experts) is exponential

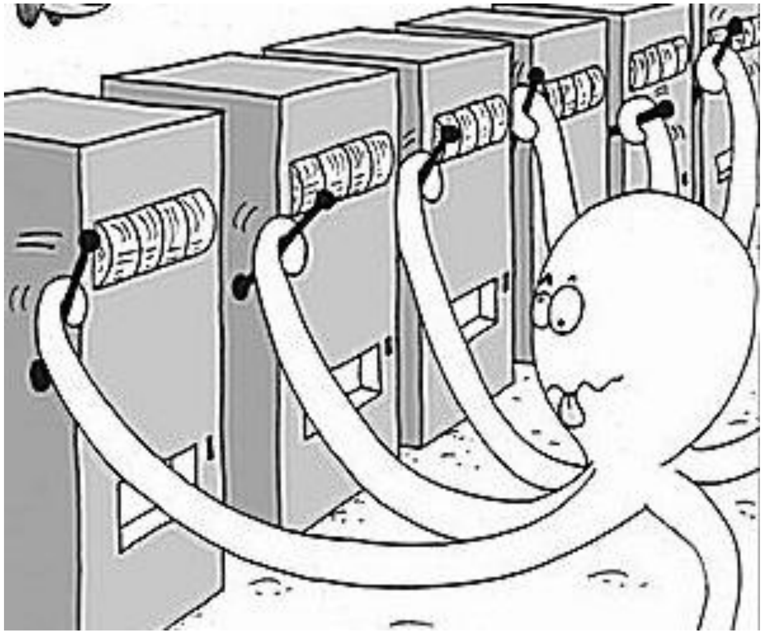
- FPL allows solving the problem in polynomial time.

$$E[cost] \leq (1+\varepsilon)(\text{best-time in hindsight}) + \frac{O(mn \log n)}{\varepsilon}$$

Duke UNIVERSITY

Multi-armed Bandit Problem

- A set of actions (or arms)
- Selecting action a in A (or pulling an arm) results in a reward from an unknown probability distribution P(r | a)
- At time=t, agent selects action a_t
- Environment generates reward r_t
- Goal is to maximize $\Sigma_t r_t$



Applications

- Web advertising
  - What is the best ad/article to show a user?
- Clinical trials
  - Identifying efficient drugs with minimal patient loss/side-effects
- Web search
  - Which result must be ranked at the top?
- …

Regret

- Action value: $Q(a) = E(r \mid a)$ (mean reward)
- Optimal value: $V^* = Q(a^*) = \max_a Q(a)$
- Regret at time t : $E[\, V^* - Q(a_t)]$
- Maximizing cumulative reward is equivalent to minimizing total regret.

Duke UNIVERSITY

Explore vs Exploit

- Exploit: Make the best decision given the current information
  - Keep pulling the arm with the current best estimate for the reward

- Explore: Gather more information
  - Pull a different arm

- We can estimate the action value Q(a) by Monte Carlo estimation if lever a was pulled $N_t(a)$ times as follows.

$$\widehat{Q_t}(a) = \frac{1}{N_t(a)} \sum r_t 1_{(a_t = a)}$$

Greedy Algorithm

- Start with some initial estimate for Q(a) for all a
- Keep pulling the lever with the estimated action value.

$$a^* = argmax_{a \varepsilon A} \widehat{Q_t}(a)$$

- Continuous Exploitation

- Can get stuck in suboptimal action forever

ε-Greedy

- With probability 1-ε, pull the best level
- With probability ε, choose a random different lever to pull

- Constant Exploration

- Let Δ_a = V* - Q(a). Then total regret at t steps is at least:

$$t \cdot \frac{\varepsilon}{|A|} \sum_{a \varepsilon A} \Delta_a$$

UCB1

[Auer et al 2002]

- Optimism in the face of uncertainty

- Do not dismiss an action unless it is pretty certain that it has a low value.

UCB1

- Estimate an upper confidence bound for each action value

$$P\left[Q(a) > \widehat{Q_t}(a) + \widehat{U_t}(a)\right] < \delta$$

- This depends on the number of times action *a* is selected
  - Small N(a) => Large upper bound (we are not sure Q(a) is small)
  - Large N(a) => small upper bound (estimate of Q(a) is very good)

- Select the action maximizing Upper Confidence Bound (UCB)

$$a_t = argmax_{a \varepsilon A} \widehat{Q_t}(a) + \widehat{U_t}(a)$$

UCB1

Theorem:
The UCB1 algorithm achieves logarithmic asymptotic total regret

$$\lim_{t \to \infty} R_t \geq 8 \log t \sum_a \Delta_a$$

References

Littlestone & Warmuth, “The weighted majority algorithm”, Information Computing ‘94

Arora, Hazan & Kale, “The multiplicative weights update method”, TR Princeton Univ, ’05

A. Kalai, S. Vempala “Efficient algorithms for online decision problems.” In Journal of Computer and System Sciences, 2005.

P. Auer, N. Cesa-Bianchi, P. Fischer, “Finite Time analysis of Multi-Armed Bandit Problem”, JMLR 2002