Today

- Last lecture!
- More multiplicative weights
Last Time

- $n$ experts, set $P$ of events
- If outcome $j$, expert $i$ pays penalty $M(i, j)$
  - Will assume all $M(i, j) \in [-\ell, \rho]$, for $\ell < \rho$
  - Call $\rho$ the width
- Want strategy so that expected penalty is not much worse than best expert (in hindsight)
At time $t$, let expert $i$ have weight $w^t_i$

Play distribution $D^t = \{p_1^t, \ldots, p_n^t\}$, where

$$p_i^t = \frac{w_i^t}{\sum_k w_k^t}$$

**Multiplicative Weights Update Algorithm**

- **Initialize:** $w_i^1 = 1$ for all $i$

- **At step $t$:**
  - Pick an expert according to distribution $D^t$ and use it make prediction
  - Let $j^t \in P$ be outcome
  - Update weight of expert $i$ by:
    $$w_i^{t+1} = \begin{cases} 
    w_i^t (1 - \epsilon)^{M(i,j^t)/\rho} & \text{if } M(i,j^t) \geq 0 \\
    w_i^t (1 + \epsilon)^{-M(i,j^t)/\rho} & \text{if } M(i,j^t) < 0 
    \end{cases}$$

- If $\epsilon \leq \min\left\{\frac{\delta}{4\rho}, \frac{1}{2}\right\}$, then after $T = \frac{16\rho^2 \ln(n)}{\delta^2}$ rounds, showed avg. error per round obeys

$$\sum_{t=1}^{T} \frac{M(D^t, j^t)}{T} \leq \delta + \sum_{t=1}^{T} \frac{M(i, j^t)}{T}$$
Approximately Solving Zero-Sum Games

- Want to use this to approximately solve zero-sum games
- Two players, a “row player $R$ and a “column player $C$
- Each has a (possibly different) finite set of pure strategies
- Given payoff matrix $M$ whose rows are indexed by $R$’s strategies and whose cols are indexed by $C$’s strategies
- If $R$ plays strategy $i$ and $C$ plays strategy $j$, then payoff from $R$ to $C$ is given by $M(i, j)$
  - For simplicity, let’s assume normalized s.t. all $M(i, j) \in [0, 1]$
- Allowed to play mixed strategies = probability distributions over strategies
  - Let $D$ and $P$ be row and column mixed strategies, resp.
- von Neumann’s Minimax Theorem says:
  $$\lambda^* := \min_D \max_j M(D, j) = \max_P \min_i M(i, P)$$
- $\lambda$ is called value of the game
  - Goal is to find it up to some additive error $\delta$
Approximately Solving Zero-Sum Games with M-W

\[ \lambda^* := \min_{D} \max_{j} M(D, j) = \max_{P} \min_{i} M(i, P) \]

- Experts ↔ pure strategies for R
- Events ↔ pure strategies for C
- Penalty paid by expert i when have event j is \( M(i, j) \)
- For any given \( D \), we assume know how to find column strategy \( j \) that maximizes \( M(D, j) \)
  - This is always \( \geq \lambda^* \)
- So imagine following scenario, in which play game for many rounds:
  - Every round, we choose a distrib. over experts ↔ mixed strategy \( D \) for R
  - Get back the worst possible event ↔ col. strat. that maximizes \( M(D, j) \)
  - We pay penalty \( M(D, j) \)
- Exactly setup from earlier, and can choose \( D \) using M-W
Approximately Solving Zero-Sum Games with M-W (cont.)

\[ \lambda^* := \min_D \max_j M(D, j) = \max_P \min_i M(i, P) \]

- Let's see what our M-W analysis gives us
- For any distribution \( D \), know \( \sum_t M(D, j^t) \geq \min_i \sum_t M(i, j^t) \)
  - Since distribution is just a weighted average of pure strategies
- Also know that, for all \( t \), our distribution at time \( t \) has \( M(D^t, j^t) \geq \lambda^* \)
  - Since column player plays his best strategy given our choice of \( D \)
- M-W theorem thus gives that, after \( T = 16 \ln(n) / \delta^2 \) rounds and for any \( D \),
  \[
  \lambda^* \leq \frac{\sum_{t=1}^T M(D^t, j^t)}{T} \leq \delta + \min_i \left\{ \frac{\sum_{t=1}^T M(i, j^t)}{T} \right\} \leq \delta + \frac{\sum_{t=1}^T M(D, j^t)}{T}
  \]
- Take \( D = \) optimal row strategy \( D^* \), get \( \lambda^* \leq \frac{\sum_{t=1}^T M(D^t, j^t)}{T} \leq \delta + \lambda^* \)
- So \( \frac{\sum_{t=1}^T D^t}{T} \) is approximately optimal strategy, and \( \frac{\sum_{t=1}^T M(D^t, j^t)}{T} \) is approximately value of the game
  - Could also have used best \( D^t \) instead of average one
Solving Linear Programs

- It turns out that solving any LP can be cast as solving a zero-sum game
  - Will discuss something even more general next
  - Need to make sure to pay attention to width
- How parameters that dictate rate of convergence compare to ellipsoid or interior point methods is very interesting
- We get error $\delta$ after something $O(\ln n/\delta^2)$ iterations
  - Don’t forget: in the width 1 case
  - Otherwise, need to throw in a $\rho^2$
- Ellipsoid / IP algs get error $\delta$ after $\text{poly}(n) \ln(1/\delta)$
  - And only depend logarithmically on what would correspond to the width
- So interesting tradeoff: IP algs are much better w.r.t. error and size of numbers, whereas M-W algs are much better w.r.t. dimension.
The Plotkin-Shmoys-Tardos Framework

- Instead, let's try approaching optimization problems more directly
- Want to check feasibility of linear program

\[ Ax \geq b, \quad x \geq 0, \]

where \( A \) is an \( m \times n \) matrix, and \( x \in \mathbb{R}^n \)
- Will need to construct an oracle that answers the following question:

\[ \exists \exists x \geq 0 \quad s.t. \quad c^T x \geq d \]

- Will easily generalize to other problems just by changing oracle
- Will use it with \( c = \sum_i p_i A_i \) and \( d = \sum_i p_i b_i \)

- Constructing the oracle here is very easy
  - Just has one constraint in addition to nonnegativity
  - Only way infeasible is if \( d > 0 \) and all \( c_i < 0 \)
- Will find an approximately feasible solution, i.e., an

\[ x \geq 0 \quad s.t. \quad A_i x \geq b_i - \delta \quad \forall i \]
The Plotkin-Shmoys-Tardos Framework (cont.)

- **Want:** \( x \geq 0 \) \( \text{s.t.} \ A_i x \geq b_i - \delta \ \forall i \)
- **Have oracle that answers:** \( \exists? x \geq 0 \text{ s.t. } c^T x \geq d \)
  (with \( c = \sum_i p_i A_i \) and \( d = \sum_i p_i b_i \))

**Mapping onto M-W:**
- Expert for each of the \( m \) constraints
- Events correspond to points \( x \geq 0 \)
- Penalty for expert \( i \) given event \( x \) is \( A_i x - b_i \)
  - Assume \( A_i x - b_i \in [-\rho, \rho] \) (so width \( \leq \rho \))

So, in each round:
- We pick a distribution over experts \( \leftrightarrow \) weighted sum of constraints
  - \( \sum_i p_i A_i \leq \sum_i p_i b_i \)
- Oracle gives us back a point that satisfies this weighted sum
  - Just one constraint!
- We pay penalty \( \sum_i p_i (A_i - b_i) \) and pick new weights
The Plotkin-Shmoys-Tardos Framework (cont.)

- **Want**: \( x \geq 0 \) s.t. \( A_i x \geq b_i - \delta \ \forall i \)
- **Have oracle that answers**: \( \exists? x \geq 0 \) s.t. \( c^T x \geq d \)

(with \( c = \sum_i p_i A_i \) and \( d = \sum_i p_i b_i \))

- If ever give oracle infeasible query, know original question was infeasible since just giving pos. linear combinations of original constraints
- Otherwise, oracle always gives us back a feasible \( x \), and MW says that \( \forall i \)

\[
\frac{\sum_{t=1}^T \sum_j p_j [A_j x^t - b_j]}{T} \leq \delta + \frac{\sum_{t=1}^T [A_i x^t - b_i]}{T}
\]

after \( T = \frac{16 \rho^2 \ln(n)}{\delta^2} \) rounds
- For so-called “packing and covering” LPs, \( \rho^2 \) becomes a \( \rho \)
- So if take

\[
\bar{x} = \frac{\sum_t x^t}{T},
\]

then \( A_i \bar{x} \geq b_i - \delta \ \forall i \), and have approx solution
Some comments on Plotkin-Shmoys-Tardos

- Note the sign of the penalties: Penalty is higher when constraint is better satisfied
- Idea is that we want higher weight on unsatisfied constraints so that we’re forced to satisfy them better
- **One way to think of this:** We’re really trying to prove constraints are infeasible by deducing an infeasible constraint from them
  - Could interpret as trying to construct a dual solution
  - If never come up with one, MW optimality analysis says one doesn’t exist, up to a $\delta$
- Note that actual solution is average of solutions at each time step
  - Common feature of MW algorithms
- **Can interpret this as a zero-sum game**
  - We want to prove infeasibility, play a positive combination of constraints
  - Other player wants to prove feasibility, plays a point
  - Amount we pay given by how well other player satisfies our comb. of constraints
  - If LP is feasible, game has nonnegative value, and finding optimal strategies gives feasible point
- Can replace $x \geq 0$ with $x \in P$ for any “easy” $P$ for which we can construct oracle for “$\exists x \in P$ s.t. $c^T x \geq d$”
  - Often can implement combinatorially
Boosting

- Question from machine learning
- Have domain $X$ try to learn function $c : X \to \{0, 1\}$ from some concept class $C$
- Get sequence of training examples $(x, c(x))$, where $x$ comes from some fixed but unknown distribution $\mathcal{D}$ on $X$
- Goal is to generate a hypothesis $h : X \to \{0, 1\}$
- Error is defined to be $E_{x \sim \mathcal{D}}[|h(x) - c(x)|]$
- Want a strong learning algorithm: For every distribution $\mathcal{D}$ and any given $\epsilon, \delta > 0$, outputs with prob. $\geq 1 - \delta$ a hypothesis whose error is $\leq \epsilon$
- Given $\gamma$-weak learning algorithm: Same, but error is $\leq 1/2 - \gamma$
- Boosting shows that if $\gamma$-weak learning algorithm exists for $C$, then a strong learning algorithm exists
- Very useful in both theory and practice for combining weak “rules of thumb” into strong predictions
- We’ll show this in case with fixed training set with $N$ examples and where strong algorithm has small error w.r.t. uniform distribution on the training set
  - Can get general result with a VC dimension argument and proper choice of training set
Boosting (cont.)

- Each round, give a different distribution $D^t$ on examples to weak learning algorithm and get back a hypothesis $h^t$ with error $\leq 1/2 - \gamma$ w.r.t. $D^t$
- Final hypothesis $h_{\text{final}}(x)$ will be obtained by taking majority vote among $h^1(x), h^2(x), \ldots, h^T(x)$
- Experts ↔ samples in the training set
- Events ↔ hypotheses produced by the weak learning algorithm
- Penalty for expert $x$ on hypothesis $h$ is 1 if $h(x) = c(x)$ and 0 otherwise
  - Want to increase weight of an example if our hypothesis got it wrong
- Start with uniform distribution, and update it according to M-W
- Get that error rate of $h_{\text{final}}$ on training set under uniform distribution is $\leq \epsilon$ after

$$T = \frac{2}{\gamma^2} \ln \frac{1}{\epsilon}$$

rounds
- Doing it in general involves sampling to get training set, and that’s where the $\delta$ comes from
  - Size $N$ of training set depends on VC dimension of concept class
Approximation Algorithms

- Can also use to get $O(\log n)$ approximation algorithms for many NP-hard problems
  - We’ll do one example, and many others are similar
- Problem is SET COVER:
  - Given universe $U = \{1, \ldots, n\}$ and collection $C$ of subsets of $U$ whose union equals $U$
  - Want to pick a minimum number of sets from $C$ to cover all of $U$
- MW will actually end up giving the greedy algorithm and will prove approximation bound
- Experts $\leftrightarrow$ elements of the universe
- Events $\leftrightarrow$ sets $C_j \in C$
- Penalty $M(i, C_j)$ for expert $i$ and set $C_j$ will be 1 if $i \in C_j$ and 0 otherwise
- Will take $\epsilon = 1$ and use update rule

$$w_{i}^{t+1} = w_{i}^{t} (1 - \epsilon M(i, C_j))$$

- Pretty much the same as what we’ve been using, but cleans up analysis a bit
- Gives elements weight 0 if covered by sets chosen so far, 1 otherwise
In each round:

- We pick a set of weights $w_1, \ldots, w_n$, and thus a distribution with $p_i = w_i / \sum_j w_j$
  - But this will just be the uniform distribution over uncovered elements
- Get maximally adversarial event $\leftrightarrow$ set that covers max number of uncovered elements
- Update our weights
  - Which just means zero out the elements we’ve covered

So our algorithm is really just the greedy one: repeatedly pick the set that covers the most uncovered elements.

For any distribution $p_1, \ldots, p_n$ on the elements, know that OPT sets cover everything:

- I.e., total weights of sets involved w.r.t. the $p_i$ is at least 1
- So at least one set must cover at least $1 / \text{OPT}$ fraction
- Gives $\max_{C_j} \sum_{i \in C_j} p_i \geq 1 / \text{OPT}$

So every round, total penalty drops significantly:

$$\Phi^{t+1} < \Phi^t e^{-\epsilon / \text{OPT}} = \Phi^t e^{-1 / \text{OPT}}$$

- Strict inequality because always get strictly positive penalty
- $\Phi^1 = n$
- So after $\ln n \cdot \text{OPT}$ iterations, $\Phi < 1$, and thus $= 0$
- So cover everything with $\ln n \cdot \text{OPT}$ sets, and thus get $\ln n$ approximation
That's It!

Thanks for coming, and have a great winter vacation!