Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

Zero Sum Games: In this game, there are two players: a maximizer and a minimizer. We generally write the payoff matrix M in perspective of the maximizer, so every row corresponds to an action that the maximizer can take, every column corresponds to an action that the minimizer can take, and a positive entry corresponds to the maximizer winning. M is a n by m matrix, where n is the number of choices the maximizer has, and m is the number of choices the minimizer has.

A linear program that represents fixing the maximizer's choices to a probabilistic distribution where the maximizer has n choices, and the probability that the maximizer chooses choice i is p_i is the following:

$$\max(z)$$
$$M_{1,1}(p_1) + \cdots + M_{n,1}(p_n) \geq z$$
$$M_{1,2}(p_1) + \cdots + M_{n,2}(p_n) \geq z$$
$$\vdots$$
$$M_{1,m}(p_1) + \cdots + M_{n,m}(p_n) \geq z$$
$$p_1 + p_2 + \cdots + p_n = 1$$
$$p_1, p_2, \cdots, p_n \geq 0$$

The dual represents fixing the minimizers choices to a probabilistic distribution.

By strong duality, the optimal value of the game is the same if you fix the minimizer's distribution first or the maximizer's distribution first.

1 Zero-Sum Games Short Answer

(a) Suppose a zero-sum game has the following property: The payoff matrix M satisfies $M = -M^\top$. What is the expected payoff of the row player?

(b) True or False: If every entry in the payoff matrix is either 1 or -1 and the maximum number of 1s in any row is k, then for any row with less than k 1s, the row player's optimal strategy chooses this row with probability 0. Justify your answer.

(c) True or False: Let M_i denote the ith row of the payoff matrix. If $M_1 = \frac{M_2+M_3}{2}$, then there is an optimal strategy for the row player that chooses row 1 with probability 0.

Solution:

(a) To get the column player's payoff matrix, we negate the payoff matrix and take its transpose. So we get that the row and column players' payoff matrices are the same matrix. In turn, they must have the same expected payoff, but also the sum of their expected payoffs must be 0, so both players must have expected payoff 0.

(b) False: Consider the 2-by-3 payoff matrix:

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix}$$

The row player's optimal strategy is to choose the two rows with equal probability - note that the column player doesn't care about choosing column 1 vs column 3, so this game is no different than the zero-sum game for the 2-by-2 payoff matrix:

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

(c) True: Consider the optimal strategy for the row player. If the row player chooses rows 1, 2, 3 with probabilities p_1, p_2, p_3, they can instead choose row 1 with probability 0, row 2 with probability $p_2 + p_1/2$, and row 3 with probability $p_3 + p_1/2$. The expected payoff of this strategy is the same, so this strategy is also optimal.

2 Permutation Games

A permutation game is a special form of zero-sum game. In a permutation game, the payoff matrix is n-by-n, and has the following property: Every row and column contains exactly the entries $p_1, p_2, \ldots p_n$ in some order. For example, the payoff matrix might look like:

$$P = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_2 & p_3 & p_1 \\ p_3 & p_1 & p_2 \end{bmatrix}$$

Given an arbitrary permutation game, describe the row and column players' optimal strategies, justify why these are the optimal strategies, and state the row player's expected payoff (that is, the expected value of the entry chosen by the row and column player).

Solution: Both players' optimal strategy is to choose a row/column uniformly at random. The expected payoff of this strategy is $\sum_k p_k/n$.

To show these are optimal, note that if the column player picks this strategy, any strategy the row player chooses has expected payoff $\sum_k p_k/n$. By symmetry, this also implies that if the row player picks this strategy, any strategy the column player picks achieves the same expected payoff. By duality, this must be the optimal pair of mixed equilibrium strategies.

3 Multiplicative Weights Intro

Multiplicative Weights

This is an online algorithm, in which you take into account the advice of n experts. Every day you get more information on how good every expert is until the last day T.

Let's first define some terminology:

- $x_i^{(t)}$ = proportion that you 'trust' expert i on day t
- $l_i^{(t)}$ = loss you would incur on day t if you invested everything into expert i
- total regret: $R_T = \sum_{t=1}^{T} \sum_{i=1}^{n} x_i^{(t)} l_i^{(t)} - \min_{i=1,\ldots,n} \sum_{t=1}^{T} l_i^{(t)}$

$\forall i \in [1, n]$ and $\forall t \in [1, T]$, the multiplicative update is as follows:

$$w_i^{(0)} = 1$$

$$w_i^{(t)} = w_i^{(t-1)}(1-\epsilon)^{l_i^{(t-1)}}$$

$$x_i^{(t)} = \frac{w_i^{(t)}}{\sum_{i=1}^{n} w_i^{(t)}}$$

If $\epsilon \in (0, 1/2]$, and $l_i^{(t)} \in [0, 1]$, we get the following bound on total regret:

$$R_T \leq \epsilon T + \frac{\ln(n)}{\epsilon}$$

Let's play around with some of these questions. For this problem, we will be running the randomized multiplicative weights algorithm with two experts. Consider every subpart of this problem distinct from the others.

(a) Let's say we believe the best expert will have cost 20, we run the algorithm for 100 days, and epsilon is $\frac{1}{2}$. What is the maximum value that the total loss incurred by the algorithm can be?

(b) What value of ϵ should we choose to minimize the total regret, given that we run the algorithm for 25 days?

(c) We run the randomized multiplicative weights algorithm with two experts. In all of the first 140 days, Expert 1 has cost 0 and Expert 2 has cost 1. If we chose $\epsilon = 0.01$, on the 141st day with what probability will we play Expert 1? (Hint: You can assume that $0.99^{70} = \frac{1}{2}$)

Solution:

(a) total regret = loss of algorithm - offline optimum $\leq \epsilon T + \frac{\ln(n)}{\epsilon}$.

$$\text{loss of algorithm} - 20 \leq \frac{1}{2}(100) + \frac{\ln(2)}{\frac{1}{2}}$$

$$\text{loss of algorithm} \leq 50 + 2\ln(2) + 20$$

The maximum loss is roughly 71.39.

(b)

$$R_T \leq \epsilon T + \frac{\ln(n)}{\epsilon}$$

$$R_T \leq \epsilon 25 + \frac{\ln(2)}{\epsilon}$$

Take the derivative with respect to epsilon and set the derivative to 0 to get:

$$25 - \frac{\ln(2)}{\epsilon^2} = 0$$

$$25 = \frac{\ln(2)}{\epsilon^2}$$

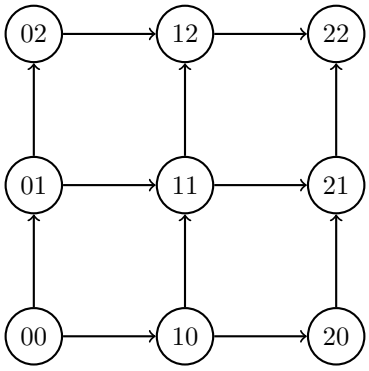
$$\epsilon = \sqrt{\frac{\ln(2)}{25}}$$

ϵ should be roughly 0.17.

(c) The weight assigned to expert 1 is $.99^{0 \cdot 140} = 1$, while the weight assigned to expert 2 is $.99^{1 \cdot 140} \approx 1/4$. So, the probability we play expert 1 is $\frac{1}{1+1/4} = 4/5$.

4 Multiplicative Weights

Consider the following simplified map of Berkeley. Due to traffic, the time it takes to traverse a given path can change each day. Specifically, the length of each edge in the network is a number between $[0, 1]$ that changes each day. The travel time for a path on a given day is the sum of the edges along the path.



For T days, both Tynan and Selina drive from node 00 to node 22.

To cope with the unpredictability of traffic, Selina builds a time machine and travels forward in time to determine the traffic on each edge on every day. Using this information, Selina picks the path that has the smallest total travel time over T days, and uses the same path each day.

Tynan wants to use the multiplicative weights update algorithm to pick a path each day. In particular, Tynan wants to ensure that the difference between his expected total travel time over T days and Selina's total travel time is at most $T/10000$. Assume that Tynan finds out the lengths of all the edges in the network, even those he did not drive on, at the end of each day.

(a) How many experts should Tynan use in the multiplicative weights algorithm?

(b) What are the experts?

(c) Given the weights maintained by the algorithm, how does Tynan pick a route on any given day?

(d) The regret bound for multiplicative weights is as follows:

Theorem. Assuming that all losses for the n experts are in the range $[0, 4]$, the worst possible regret of the multiplicative weights algorithm run for T steps is

$$R_T \leq 8\sqrt{T \ln n}$$

Use the regret bound to show that expected total travel time of Tynan is not more than $T/10000$ worse than that of Selina for large enough T.

Solution:

(a) 6 experts

(b) There is one expert for every path from 00 to 22. One can see that there are 6 different paths from 00 to 22.

(c) The algorithm maintains one weight for each path. Tynan picks a path with probability proportional to its weight.

(d) Let $P_1, \ldots, P_6$ be the 6 possible paths between 00 and 22. Let $\ell_i^{(t)}$ denote the length of path i on day t. Let $w_i^{(t)}$ denote the weight of path i on day t.

Since Tynan picks a path proportional to its weight, the expected total time on day t is

$$\sum_{i=1}^{6} w_i^{(t)} \cdot \ell_i^{(t)},$$

and the expected total time over T days is,

$$\sum_{t=1}^{T} \sum_{i=1}^{6} w_i^{(t)} \cdot \ell_i^{(t)}.$$

The regret bound to multiplicative weights asserts that for every path P_i,

$$\sum_{t=1}^{T} \sum_{i=1}^{6} w_i^{(t)} \cdot \ell_i^{(t)} - \sum_{t=1}^{T} \ell_i^{(t)} \leq 8\sqrt{T \ln 6}\ .$$

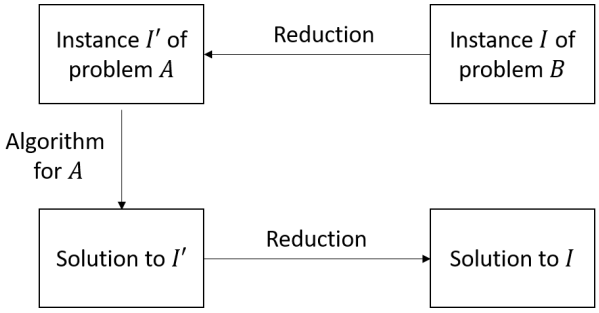
Here $n = 6$, since there are 6 different paths. Since Selina picks one of the paths P_i to use over all days, the above regret bound implies that total expected time of Tynan is at most $8\sqrt{T \ln 6}$ worse than that of Selina. For sufficiently large T, $8\sqrt{T \ln 6} < T/10000$, in particular $T > (8 \cdot 10000)^2 \cdot \ln 6$ suffices.

Reduction: Suppose we have an algorithm to solve problem A, how can we use it to solve problem B?

This has been and will continue to be a recurring theme of the class. Examples so far include

- Use LP to solve max flow.
- Use max flow to solve min s-t cut.
- Use minimum spanning tree to solve maximum spanning tree.
- Use Huffman tree to solve twenty questions.

In each case, we would transform the instance I of problem B we want to solve into an instance I' of problem A that we can solve, and also describe how to take a solution for I' and transform it into a solution for I:



Importantly, the transformation should be efficient, i.e. takes polynomial time. If we can do this, we say that we have reduced problem B to problem A.

Conceptually, a efficient reduction means that if we can solve problem A efficiently, we can also solve problem B efficiently. On the other hand, if we think that B cannot be solved efficiently, we also think that A cannot be solved efficiently. Put simply, we think that A is "at least as hard" as B to solve.

To show that the reduction works, you need to prove (1) if there is a solution for instance I' of problem A, there must be a solution to the instance I of problem B and (2) if there is a solution to instance I of B, there must be a solution to instance I' of problem A.

5 Some Sums

Given an array $A = [a_1, a_2, \ldots, a_n]$ of nonnegative integers, consider the following problems:

1 **Partition**: Determine whether there is a subset $P \subseteq [n]$ ($[n] := \{1, 2, \cdots, n\}$) such that $\sum_{i \in P} a_i = \sum_{j \in [n] \setminus P} a_j$

2 **Subset Sum**: Given some integer t, determine whether there is a subset $P \subseteq [n]$ such that $\sum_{i \in P} a_i = t$

3 **Knapsack**: Given some set of items each with weight w_i and value v_i, and fixed numbers W and V, determine whether there is some subset $P \subseteq [n]$ such that $\sum_{i \in P} w_i \leq W$ and $\sum_{i \in P} v_i \geq V$

For each of the following clearly describe your reduction and justify its correctness.

(a) Find a linear time reduction from Subset Sum to Partition.

(b) Find a linear time reduction from Subset Sum to Knapsack.

Solution:

(a) Suppose we are given some A with target sum t. Let s be the sum of all elements in A. If $s - 2t \geq 0$, generate a new set $A' = A \cup \{s - 2t\}$. If A' can be partitioned, then there is a subset of A that sums to t.

We know that the two sets in our partition must each sum to $s - t$ since the sum of all elements will be $2s - 2t$. One of these sets, must contain the element $s - 2t$. Thus the remaining elements in this set sum to t.

If $s - 2t \leq 0$, generate a new set $A' = A \cup \{2t - s\}$. If A' can be partitioned, then there is a subset of A that sums to t.

We know that the two sets in our partition must each sum to t since the sum of all elements will be $2t$. The set that does not contain $\{2t - s\}$ will be our solution to subset sum.

(b) Suppose we are given some set A with target sum t. For each element k of the set, create an item with weight k and value k. Let $V = t$ and $W = t$. We know Knapsack will determine if there is a combination of items with sum of weights $\leq t$ and values $\geq t$. Because the weights and values are the same, we know (Sum of chosen weights) = (Sum of chosen values) = t. And since each weight/value pair is exactly the value of one of the original elements of A, we know that there will be a solution to our Knapsack problem iff there is one for our subset sum problem.