****

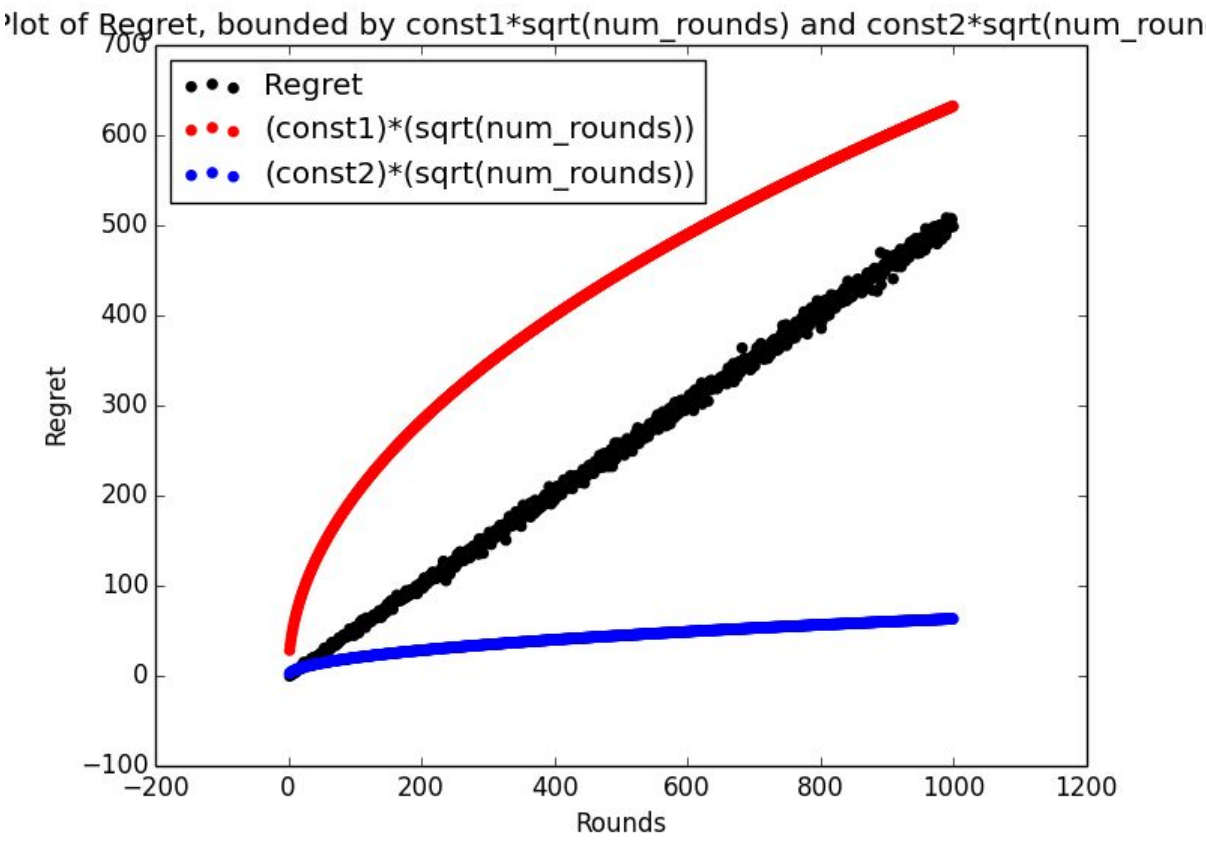
Alimulla Shaik
shal5122@colorado.edu
Algorithms problem set 4
No Collaborators
****

1) a)
Sample output for the AI game is displayed below.

Enter number of rounds you would like to select:4
Number of rounds selected by user: [4]
Generated Payoff Matrix:
[5, -3, 7]
[3, 10, 6]
[6, -6, 6]
[1, -10, 1]
****
Round: [0]
****
Enter the row you would like to select:5
Action selected by user: [5]
Incorrect entry!!! Please select the row from 0 to 3
Enter the row you would like to select:1
Action selected by user: [1]
Probability distribution: [0.3333333333333333, 0.3333333333333333, 0.3333333333333333]
Action chosen by AI: [1]
Loss vector of AI: [3, 10, 6]
Weight vector of AI: [0.22313016014842982, 0.006737946999085467, 0.049787068367863944]
Old user score: [0], Updated user score: [10], Difference: [10]
Old AI score: [0], Updated AI score: [-10], Difference: [-10]
****
Round: [1]
****
Enter the row you would like to select:2
Action selected by user: [2]
Probability distribution: [0.7978760262069926, 0.0240937682868484, 0.17803020550615906]
Action chosen by AI: [2]
Loss vector of AI: [6, -6, 6]
Weight vector of AI: [0.011108996538242306, 0.1353352832366127, 0.0024787521766663585]
Old user score: [10], Updated user score: [16], Difference: [6]
Old AI score: [-10], Updated AI score: [-16], Difference: [-6]

****
Round: [2]
****
Enter the row you would like to select:3
Action selected by user: [3]
Probability distribution: [0.07459555713221444, 0.9087599242585134, 0.016644518609272355]
Action chosen by AI: [1]
Loss vector of AI: [1, -10, 1]
Weight vector of AI: [0.006737946999085467, 20.085536923187668, 0.0015034391929775726]
Old user score: [16], Updated user score: [6], Difference: [-10]
Old AI score: [-16], Updated AI score: [-6], Difference: [10]
****
Round: [3]
****
Enter the row you would like to select:0
Action selected by user: [0]
Probability distribution: [0.00033532503919087275, 0.9995898538311128, 7.482112969643796e-05]
Action chosen by AI: [1]
Loss vector of AI: [5, -3, 7]
Weight vector of AI: [0.0005530843701478336, 90.01713130052181, 4.5399929762484854e-05]
Old user score: [6], Updated user score: [3], Difference: [-3]
Old AI score: [-6], Updated AI score: [-3], Difference: [3]

1) b) We are asked to allow nature decides loss vector in [0,1] and use hedge algorithm in a) part. Below plot shows $\Theta\sqrt{T}$ regret when there are N = 2 actions.Regret is close to the given bound $\sqrt{(T/2)ln\,(N)}$ sometimes and it even deviate from it for some cases.



2) a) Sol.
We are given that, outcome of y = 1 with probability p and y = 0 with probability 1□ - p.
Loss function $L(x,y) = (x-y)^2$
Expected loss of predication 'x' is $E(L(x,y)) = pL(x,1) + (1-p)L(x,0)$
Substitute loss function in above,
$= p((x-1)^2) + (1-p)((x-0)^2)$
$= p(x^2+1-2x) + (1-p)(x)^2$
$= px^2+p-2px+x^2-px^2$
$= x^2-2px+p$

Every quadratic function in the form of $f(x) = ax^2 + bx + c$ has a minimum if $a > 0$ and that will be at f(-b/2a). Which means function will have minimum when x = - b / 2a.
Using this, the above expected loss of predication is minimum when $x = -(-2p)/ 2$
$\Rightarrow x = p$.

For the loss function $L(x,y) = |x-y|$,
Expected loss of predication 'x' is $E(L(x,y)) = pL(x,1) + (1-p)L(x,0)$
Substitute loss function in above,
$= p((|x-1|)^2) + (1-p)((|x-0|)^2)$
$= p(|x^2+1-2x|) + (1-p)(|x|)^2$
$= p(|x^2+1-2x|) + (|x|)^2 +(-p)(|x|)^2$

This can not be reduced to a quadratic function to determine minimum of the function. So, this is not the case for absolute loss and probability distribution is y = 1 with probability p and y = 0 with probability 1□ - p.

2) b) From the subgradient definition,
f (y) > = f (x) + g' (y − x) for all y then g' is subgradient of f.
We are given the loss form L(x,y) = $\phi(y) - \phi(x) - \phi'(x)\phi(y-x)$
L(x,y) = $\phi(y) - [\phi(x) + \phi'(x)\phi(y-x)]$
$[\phi(x) + \phi'(x)\phi(y-x)]$ = $\phi(y)$ - L(x,y)

Here we are given that $\phi(x)$ is strictly convex and differentiable function.
So $\phi'(x)$ is subgradient of $\phi(x)$ by definition.
⇒ $[\phi(x) + \phi'(x)\phi(y-x)]$ = $\phi(y)$ - L(x,y) <= $\phi(y)$
⇒ L(x,y) >= 0 ---> (1)

We are given that, outcome of y = 1 with probability p and y = 0 with probability 1□ - p.
Expected loss of predication 'x' is E(L(x,y)) = pL(x,1) + (1-p)L(x,0)
E(L(x,y)) = pL(x,1) + (1-p)L(x,0) ----> (2)
From equation (1) and (2) E(L(x,y)) > = 0 and it is only true when x = p.
So, this will be best prediction when x = p.

3) Sol.

Algorithm for meta_hedge:
Steps: 1. Generate horizon guess T' and set learning rate (eta) = 1/sqrt(T'). 2. Pass T' and eta to hedge algorithm (defined in question 1). 3. Based on the outcome, keep on update eta and T' untill algorithm runs. 4. Ensure, corresponding weights and regret values are updated.

Correctness:
There are 2 cases possible here, horizon guess T' can be greater or lesser than T.

Case #1:
If T' > T, then our meta algorithm stops before and runs lesser times than what we have expected it to run. As we overestimated T' (our guess), eta value will be lesser as it is inversely proportional to T'. And as so weights and losses (which are directly proportional to eta). This will ensure that regret is also smaller as it progresses. In question 1 we have seen that hedge algorithm shows a regret bound of $\Theta\sqrt{T}$. In this case it even reduces further and will fall in lower bound of $\sqrt{T}$ with complexity $O\sqrt{T}$ though T is not known ahead of time.

Case #2:

If T' < T, then our meta hedge algorithm runs more than what is expected. In this case, update eta and corresponding weights and regrets. Now, we will try to update T' and ensures it is greater than previous value. This will lead to similar situation like case 1 where regret will reduce further as T' will increase going ahead and $O\sqrt{T}$ is achieved.

Running Time:
Our meta algorithm runs hedge algorithm and keep on update T' and eta with corresponding weights and regrets. So it will have same running time of hedge algorithm. In the correctness, I have showed that regret values achieved are always within $O\sqrt{T}$.

4) a)
We are given that, hardware store has exactly one each of the n products available. Deterministic algorithm or online solution is shown below.

Online solution:
Steps: 1. Customer 'i' comes to the store. 2. Check the item requested by customer and look for the same in store. 3. If that item is available then sell it to her, otherwise report her that you do not have the item. 4. Repeat this procedure for all the customers.

But we can develope offline optimal solution using maximal matching to serve more customers. Let's say you know all of the items that customers need then you can use stable matching algorithm to maximize the number of customers served. In the store, you know the requirement of all the customers and generate stable matching so that all the items are matched with at most one customer if the customer needs that particular item. To do so, first sort all the customers according to their size of the items set, start with customer who is having least items in the set and match the item. Keep on matching till all the items in the store are either empty or left with items which are not required for any customer.

Optimal solution:
Steps: 1. Create a bipartite graph with customers and items and generate maximal matching. 2. Check the items each customer may need and sort them according to size of the items set. 3. Start with customer who has least items in the set and if any of the item is available in matching then match customer with that item. 4. Otherwise, move to the next customer. 5. If an item is not matched and left alone then check which customer needs it. 6. And the customer is found and already matched then check that customer to match with some other item and continuously update the customer with new matching.

7. Continue step 3 until all the items in the store are either empty or left with items which are not required for any customer.

Example:
Let's say you have items {1,2,3,4} in the store and customers A = {1,2,4} B = {2,3}, C = {2}, D = {1}. Then online solution will assign A - {1}, B - {2}. It served only 2 customers. But our optimal solution will start with least set first, means D - {1}, C = {2}, B - {3} and A - {4}.

From the above example, optimal solution served 'n' customers and online solution served 'n/2' customers at least. So we can say that online solution sells at least half as many items as optimal solution. Competitive ratio OPT/ALG <= n / (n/2)
⇒ OPT/ALG <= 2

Correctness:
Online solution only checks whether item requested by customer is present in the store or not. If present then sell it to the customer and continue the procedure for all the customers. It does not optimize the allocation of items to the customers and does not care to maximize the number of customers served. To achieve this, we developed an optimal strategy using maximal matching which helps to serve more customers than online solution. It takes care of the cases where no items are matched to any customer or vice versa. It simply return in that cases. It even handles the case where size of items of all the customers are large.

4) b) From the part a) we know that OPT/ALG <= 2. Now we have to prove that every deterministic algorithm has competitive ratio of at least 2 which means, we need to prove OPT/ALG >=2. In the part a) we have come up with online solution and optimal solution which is possible. Proposed deterministic solution guarantees to serve at least half of the items as optimal solution but we can not optimize this solution further. So we can say that, any deterministic algorithm has at least competitive ratio 2 and OPT/ALG >=2 for any deterministic algorithm.

4) c) In the part a) proposed an optimal solution using maximal matching which helps to serve more number of customers and gives optimized version of deterministic algorithm. Competitive ratio of the same algorithm is OPT/ALG <=2.
Here we are given a randomized algorithm which picks item uniformly random from those still available and sell it to the customers. It picks item randomly and only checks whether the particular item is available or not. This algorithm uses randomized strategy and will not look to maximize the possibility of serving more customers like our optimal solution. So randomized algorithm can not produce as efficient results as optimal solution and can not optimize it to further levels. And the best competitive ratio randomized algorithm can achieve is always less than optimal solution.
Hence randomized algorithm competitive ratio is OPT/ALG < 2 for any number of items.

5)
```
if __name__ == '__main__':
    M = np.array([[1,0,5,8,3],
```

```
            [5,0,2,6,1],
            [0,5,8,3,5],
            [0,2,6,1,0],
            [5,8,3,5,0]])

i = 0
T = 10
Wts = [1]*len(M[0])

wt_size = 5
row_weights = np.ones((wt_size ,1))
column_weights = np.ones((wt_size ,1))
eta = 1.0/math.sqrt(T)

payoff_row = np.zeros((wt_size,1))
payoff_col = np.zeros(wt_size)

print ( ‘eta:', eta)
print (' row payoff', payoff_row )
print (' column payoff', payoff_col)

print (‘ row weights', row_weights)
print (‘ column weights', column_weights)

while (i != 100):
    print ('****************************************')
    print (' Round: [', i, ']')
    print ('****************************************')

    arr = []
    for j in range(len(row_weights)):
        arr.append(j)
    prob_distr = []
    for wt in row_weights:
        prob_distr.append(wt/sum(row_weights))

    print ('Probability distribution for Row:', prob_distr)

    arr = []
    for j in range(len(column_weights)):
        arr.append(j)
    prob_distr = []
    for wt in column_weights:
        prob_distr.append(wt/sum(column_weights))

    print ('Probability distribution for column:', prob_distr)
```

```
    for j in range(len(Wts)):
        loss = -(eta)*(loss_vector[j])
        Wts[j] = (Wts[j])*(math.exp(loss))
    print ('Weight vector of AI:', Wts)
```

6) Sol.
We can use stable marriage matching for this.
Consider below example,

Men's preferences:

1 3 2 1
2 2 3 1
3 2 1 3

Women's preferences:

1 3 2 1
2 2 3 1
3 1 3 2

Man 1 proposes to women 3 and (accepted), Man 2 proposes to women 2 and (accepted), Man 3 proposes to women 2 and (accepted). Now woman 2 rejects man 2. Man 2 proposes for woman 3 and will get reject. Will proposes for woman 1.
so stable matching generated is
{(1,3), (2,1), (3,2)}.

References:

1) http://web.eecs.umich.edu/~jabernet/eecs598course/fall2013/web/notes/lec4_091613.pdf
2) http://www.cs.berkeley.edu/~bartlett/courses/281b-sp08/21.pdf
3) https://web.stanford.edu/class/ee392o/subgrad.pdf
4) http://theory.stanford.edu/~tim/f13/l/l10.pdf
5) http://www.eecs.berkeley.edu/~luca/cs261/lecture17.pdf
6) http://web.eecs.umich.edu/~jabernet/eecs598course/fall2013/web/notes/lec8_093013.pdf
7) http://web.eecs.umich.edu/~jabernet/eecs598course/fall2013/web/notes/lec2_090913.pdf
8) http://web.eecs.umich.edu/~jabernet/eecs598course/fall2013/web/notes/lec1_090413.pdf