Exercise 1

(a) selected actions pt for t = 1, . . . , 4 for strategy A and B

Strategy A:

For $t=1$, $b(t)=1$, $p_1=(1/3,1/3,1/3)$

For $t=2$, $b(t)=1$, $p_2=(1,0,0)$

For $t=3$, $b(t)=1$, $p_3=(1,0,0)$

For $t=4$, $b(t)=2$, $p_4=(0,1,0)$

Strategy B:

$$L_t^i=\sum_{s-1}^{t} z_s^i \quad p_t^i=\frac{e^{-L_{t-1}^i}}{C_{t-1}} \tag{1.1}$$

For $t=1$, $p_1=(1/3,1/3,1/3)$

For $t=2$, $p_2=(0.3672,0.3322,0.3006)$

For $t=3$, $p_3=(0.378,0.342,0.28)$

For $t=4$, $p_4=(0.1827,0.4494,0.3679)$

(b) Compute the total mix loss obtained by all strategies and compute the corresponding expert regrets of all strategies after n = 4 rounds.

Mix loss:

$$l_m(p_t,z_t)=-\ln\left(\sum_{i=1}^{d} p_t^i e^{-z_t^i}\right) \tag{1.2}$$

Expert regret:

$$R_n^E=\sum_{t=1}^{n} l_m(p_t,z_t)-\min_i \sum_{t=1}^{n} z_t^i \tag{1.3}$$

Implementing the above formulas in Matlab, yields the corresponding value for mix loss as,

Strategy A:

$loss_{mix}(t=1)=0.0967$, $loss_{mix}(t=2)=0$,

$loss_{mix}(t=3)=1, \quad loss_{mix}(t=4)=0.9$

$R_{n=4}^{E}=1.6967$

Strategy B:

$loss_{mix}(t=1)=0.0967$, $loss_{mix}(t=2)=0.29$,

$loss_{mix}(t=3)=0.273, \quad loss_{mix}(t=4)=0.3102$

$R_{n=4}^{E}=0.4089$

(c) Upper bound

From the formula (1.3), the theoretical guarantee for AA strategy can be rewritten and re-arranged as the following procedure,

$$R_n^E=\sum_{t=1}^{n} l_m(p_t,z_t)-\min_i \sum_{t=1}^{n} z_t^i \le \log(d)$$
$$\sum_{t=1}^{n} l_m(p_t,z_t) \le \log(d)+\min_i \sum_{t=1}^{n} z_t^i \qquad (1.4)$$

Substituting the adversaries and d=3, n=4, gives the numerical value for the right hand side as

$$\log(3)+\sum_{t=1}^{n} z_t^3$$

Therefore the upper bound can be found as C=1.3986

The total mix loss for strategy A and B at n=4 are denoted as 1.9967 and 0.7089, respectively. Apparently the upper bound for the cumulative mix loss of AA strategy is larger than the total mix loss for AA strategy, while smaller than strategy A in the question sheet. It indicates the effectiveness of applying the theoretical guarantee in bounding the expert regret for AA strategy. On the contrary, strategy A in the question sheet might result in unbounded increase of total mix loss due to the absence of upper bound.

(d) show that this theoretical guarantee RE n ≤ log(d) cannot be improved

Firstly the situation when n=1,d=2, should be considered. The confidence on each expert is randomly distributed from [0,1], and the adversary moves $z_t \subset [-10,10]^5$. We can easily observe that when the adversary moves are chosen as $z_t \subset (-\infty,-1]^d$, the expert regret is always higher than log(d)=0.6931, no matter what the value of confidence is. For instance, when we pick $z_t=[-3.8063\ -6.165]$, the theoretical guarantee is determined as a value always exceeding

log3=0.6931 no matter what the confidence distribution is.

Inspired by this observation in the simplest case, we proceed to artificially choose the setting for the adversary moves. The n is fixed as n=1 for simplicity. Implementing this in Matlab we can find that for any strategy pt and for any d, there is always an adversary move $z_t \subset \left(\text{-}\infty, 0\right]^d$ making the theoretical guarantee higher than logd. This means that the bound is tight for AA strategy.

Exercise 2

For any adversary, for any n > 0, if l is the dot loss ld , we have for the Exp Strategy that

$$R_n^E \le n\frac{\eta}{8} + \frac{\ln(d)}{\eta} \tag{2.1}$$

(a) n=2

Substituting the n=2 in equation (2.1), gives the upper bound for exp strategy A

$$C_n^A = \sqrt{\log d}, \quad C_n^B = \frac{3\sqrt{2\log d}}{4} \tag{2.2}$$

Clearly C_n^A is smaller, so the bound of A is tighter

(b) n=4

Substituting the n=2 in equation (2.1), gives the upper bound for exp strategy B

$$C_n^A = \frac{3\sqrt{\log d}}{2}, \quad C_n^B = \sqrt{2\log d} \tag{2.3}$$

Clearly C_n^B is smaller, so the bound of B is tighter

(c) informal argument

The known inequalities can be found in the questions as

$$\begin{aligned} R_n^B &\le C_n^B \\ R_n^A &\le C_n^A \\ C_n^B &< C_n^A \end{aligned} \tag{2.4}$$

This can be proved by the informal argument as below,

If the upper bound of B is tighter, or $C_n^B < C_n^A$, and $R_n^B \le C_n^B, R_n^A \le C_n^A$, for any adversary moves Zt for t=1,2,…,n, there always exists a pair of $\left(R_n^B, R_n^A\right)$ which satisfies $R_n^B > R_n^A$.

This argument can be easily derived from the basic theorem of inequality. The example can be $R_n^A < R_n^B < C_n^B < C_n^A$, which complies with inequalities (2.4) but violates $R_n^B \leq R_n^A$.

(d) Generally, how does the strategy pt change when we increase the learning rate? What do you expect to happen if we set the learning rate extremely high (η = ∞)?

The strategy pt for the exp case can be denoted as

$$p_t^i = \frac{e^{-\eta L_{t-1}^i}}{C_{t-1}} \tag{2.5}$$

From the mathematical formulation of the exp strategy, the pt is evenly distributed on each dimension when $\eta=0$. With increased value of learning rate, the exponential function $e^{-\eta L_{t-1}^i}$ decreases its value from 1 towards 0, and pt becomes closer to the boundary (either 0 or 1). When the learning rate is infinitely large, it can be expected that $p_t^i \subset [0,1]$.

Exercise 3

(a) Optimal proof

The OCO regret bound for Rn can be found as

$$C_n = \frac{R^2}{2\eta} + \frac{\eta G^2 n}{2} \tag{3.1}$$

The derivative of the bound with respective to learning rate η is

$$\frac{\partial C_n}{\partial \eta} = -\frac{R^2}{2\eta^2} + \frac{G^2 n}{2} \tag{3.2}$$

When the derivative is zero, the argument of minimum of the bound can be determined as

$$\frac{\partial C_n}{\partial \eta} = -\frac{R^2}{2\eta^2} + \frac{G^2 n}{2} = 0 \Rightarrow \eta_{opt} = \frac{R}{G\sqrt{n}} \tag{3.3}$$

Substituting this optimal learning rate back in equation (3.1), gives the OCO regret bound

$$C_n = RG\sqrt{n} \tag{3.4}$$

Combining $R_n \leq C_n$ and equation (3.4) to eliminate C_n, yields

$$R_n \leq RG\sqrt{n} \quad \text{when} \quad \eta = \frac{R}{G\sqrt{n}} \tag{3.5}$$

(b) OCO regret

If we set $\eta_t = \frac{R}{G\sqrt{t}}$ and $\frac{1}{\eta_0} = 0$, t=1,2,…n

Let $g_t = \nabla l(a_t, z_t)$ and $a \subset A$. By definition of a subdiffierentiable loss, we have

$$\sum_{t=1}^{n} \left(l(a_t, z_t) - l(a, z_t) \right) \leq \sum_{t=1}^{n} g_t^T (a_t - a) \tag{3.6}$$

Based upon the definition $w_{t+1} = a_t - \eta g_t$, it can be derived that

$$\begin{aligned} g_t^T (a_t - a) &= \frac{2(a_t - w_{t+1})^T}{2\eta} \\ &= \frac{\|a_t - a\|_2^2 + \|a_t - w_{t+1}\|_2^2 - \|a - w_{t+1}\|_2^2}{2\eta} \\ &= \frac{\eta \|g_t\|_2^2}{2} + \frac{1}{2\eta}\left(\|a_t - w_{t+1}\|_2^2 - \|a - w_{t+1}\|_2^2 \right) \end{aligned} \tag{3.7}$$

Since A is a convex set, the distance between point w and an in convex set A is no smaller than the distance between two points inside the convex set A,

$$\|a_t - w_{t+1}\|_2^2 \geq \|a - a_{t+1}\|_2^2 \tag{3.8}$$

Substituting inequality (3.8) back in equation (3.7), and summing one directly, gives the following inequality

$$\sum_{t=1}^{n} g_t^T (a_t - a) \leq \frac{\|a - a_1\|_2^2}{2\eta_t} + \sum_{t=1}^{n} \frac{\eta_t \|g_t\|_2^2}{2} \tag{3.9}$$

Substituting $\eta_t = \frac{R}{G\sqrt{t}}$, $\sum_{t=1}^{n} \frac{1}{\sqrt{t}} \leq 2\sqrt{n}$, $\sqrt{t} \leq \sqrt{n}$ and $\|a - a_1\|_2 \leq R$ back in inequality (3.9),

yields

$$\begin{aligned} \sum_{t=1}^{n} g_t^T (a_t - a) &\leq \frac{R^2 G\sqrt{t}}{2R} + \sum_{t=1}^{n} \frac{RG^2}{2G\sqrt{t}} \\ &\leq \frac{R^2 G\sqrt{n}}{2R} + \frac{RG^2}{2G} 2\sqrt{n} = \frac{RG\sqrt{n}}{2} + \frac{2RG\sqrt{n}}{2} = \frac{3RG\sqrt{n}}{2} \end{aligned} \tag{3.10}$$

Combining inequality (3.10) and (3.6), we can obtain the bounded OCO regret as

$$\sum_{t=1}^{n} \left(l(a_t, z_t) - l(a, z_t) \right) \leq \frac{3RG\sqrt{n}}{2} \tag{3.11}$$

Exercise 4

(a) Assume that at time t we have a wealth of Wt ∈ R. Assume we distribute our wealth Wt at time t according to pt = (p 1 t , ..., pd t) ∈ Δd on the d different assets4 . Given the asset prices x i t and x i t+1, derive the value of our wealth Wt+1 on the next day.

The value of wealth on the next day can be derived as

$$W_{t+1} = W_t \sum_{i=1}^{d} p_t^i r_t^i \quad , where\ r_t^i = x_{t+1}^i / x_t^i \tag{4.1}$$

Based upon the derived formula, optimizing the mix loss in the investment is equivalent to maximizing the value of wealth on each day.

(b) Proof

Substitute the $z_t^i = -\log\left(r_t^i\right)$ in the left hand side:

$$\sum_{t=1}^{n} -\log\left(\sum_{i=1}^{d} p_t^i e^{-z_t^i}\right) = \sum_{t=1}^{n} -\log\left(\sum_{i=1}^{d} p_t^i e^{\log(r_t^i)}\right) = \sum_{t=1}^{n} -\log\left(\sum_{i=1}^{d} p_t^i r_t^i\right)$$

Substituting the formula (4.1), gives

$$\sum_{t=1}^{n} -\log\left(\sum_{i=1}^{d} p_t^i e^{-z_t^i}\right) = -\log\left(W_{t+1} / W\right) \tag{4.2}$$

which complies with the right hand side of the target formula

The setting of the adversary in this application should be $z_t^i = (-\infty, \infty]$, $Z = (-\infty, \infty]^d$, which is suitable for adopting mix loss instead of the dot loss. In this case it can be correlated to the stock changes we make for the investment at time t. Minimizing the total mix loss is equivalent to maximizing the ratio of W_{t+1} / W, yielding the highest theoretical gain.

(c) AA

1. Inspect files
2. Compute the adversary moves z i t = − log(r i t) and the total loss of each expert.

Please check the matlab in appendix

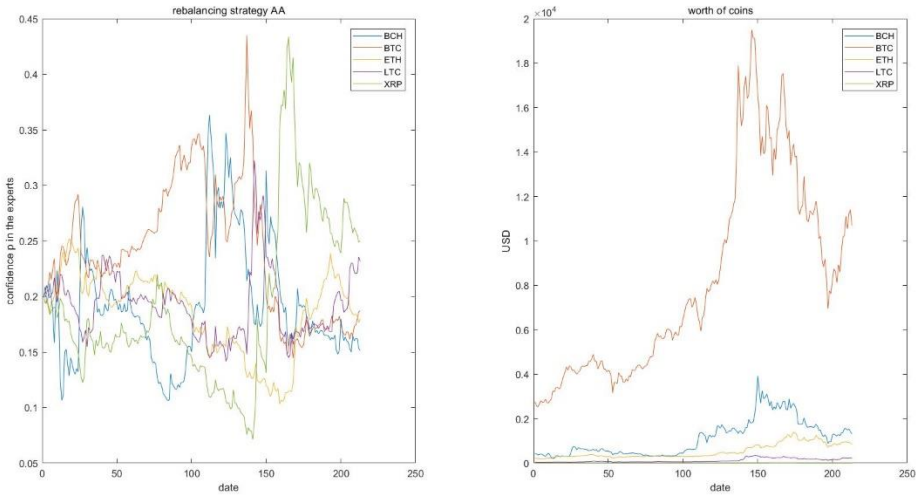
3. Implement AA (AA.m) and compute the expert regret of AA (use n = 213).

The expert regret is computed as -0.6853

4. How does the total loss of AA compare with the total loss of the experts? How does the expert regret of AA compare with the guaranteed expert regret? Is the adversary generating 'difficult' data?

Total loss the AA=-2.3396, larger than the total loss of the experts=-7.0744. The guaranteed AA regret ln5=1.6094, apparently larger than the expert regret of AA. The data generated by the adversary is not 'difficult' in this case since the bound of the expert regret is quite tight, and the expert regret is successfully bounded by the AA algorithm.

5. Visualize the strategy of AA using a plot of pt vs t. Also visualize the values of the coins, xt vs t. What do you observe? Can you explain the strategy?



Several observations have been spotted as follows,

(1) For all the cryptocurrency, the worth of coins increased until around t=150, followed by a sudden drop in the worth, revealing the ups and downs of the coins market.

(2) XRP is the most table cryptocurrency as its value barely changed (Low gain, low risk), while the BTC has the largest change in the wealth throughout the process. (High gain, high risk)

(3) The rebalance strategy can be correlated to the tendency of the worth of coins. When one of the cryptocurrency shows a rapid increase, the strategy will share higher distribution towards it; when the whole market is shrinking, the strategy will increase the share hold by XRP to minimize the loss

Obviously the aggregating algorithm has a strategy to minimize the regret between the learner and the experts for long-term investment.

6. If we would have invested according to the AA strategy, how much would our wealth have increased?

The total gain of the investment is obtained as 10.3766. Specifically it means the value of the

wealth is 10.3766% higher than its original value after 213 days investment.

(d) OCO

1. For this example, how is the OCO regret different from the expert regret considered above?

The OCO regret compared the strategy to the best fixed action, while the expert regret compared that to the best expert.

2. Show that the gradient of the mix loss is equal to

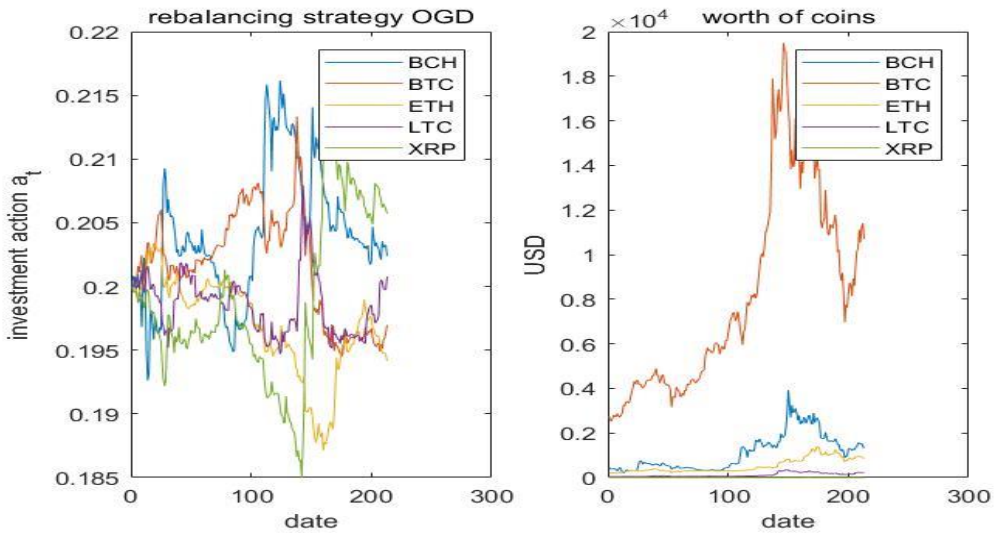
The gradient of the mix loss can be derived by using the chain rule as follows,((lnx)'=-1/x)

$$\nabla_a l_m(a, z_t) = \nabla_a\left[-\log\left(a_t^T r_t\right)\right] = \frac{\partial\left[-\log\left(a_t^T r_t\right)\right]}{\partial a_t^T} = \frac{1}{a_t^T r_t}\frac{\partial\left(a_t^T r_t\right)}{\partial a_t^T} = \frac{-r}{a_t^T r_t} \quad (4.3)$$

3. Implement mix_loss.m
4. Implement online gradient descent (OGD) in OGD.m. Use the learning rate η = R G √ n . For the OCO regret guarantee of this value of η see also exercise 3a.
5. We have provided values R and G for you to use for this data (in OGD.m). Explain why these values are OK to for these particular adversary moves zt

For the theorem guarantee, the bounds need to satisfy $\|a\| \le R, \|\nabla l(a, z)\| \le G$ .The R is chosen as 1 to guarantee that the action vector belong to A-space (within the ball with a radius=1). The G is chosen as $\sqrt{\max_t \sum_{i=1}^{d}\left(r_t^i\right)^2}$, representing the best increasing rate of the wealth over all the 5 different cryptocurrency throughout the investment. These values are suitable to perform OCO to make the strategy towards the best action.

6. Run OGD for n = 213. Visualize the strategy of OGD as in 4c. How much would our wealth have increased if we would have used OGD to invest?



The total gain of the investment would be 6.1428. Specifically it means the worth of the coins increased 6.1428% after the 213 days investments.

7. Compute the optimal 'fixed' action a ∈ A (you may use an optimization toolbox, and you may use loss_fixed_action.m). Compute the OCO regret of OGD.

The optimal fixed action is found by gradient descent with random initialization. Details can be found in the appendix. The stopping criteria is when the number of iterations>=10000 && the variance of loss and previous loss< 1e-6. This step has been repeated 50 times and make the average with respect to each dimension to make the obtained the fixed action close enough to the global optimum. The result for the fixed action can be found in the following table. The total loss for the optimal fixed action is -1.8761, superior to the total loss=-1.807 obtained previously.

Table 1 Fixed optimal action

0.233509688358882
0.175906930002309
0.152865769774275
0.231142279805120
0.206575332059415

The OCO regret is computed as 5.1218.

8. How does the loss of OGD compare with the loss of the best fixed action, and how does the OCO regret of OGD compare with the guaranteed OCO regret (see 3a)? Is the adversary generating 'difficult' data?

The loss of OGD is -1.8153, larger than that of best fixed action -1.8761. The OCO regret is 5.1218, smaller than the bound computed as 65.7803. The data of adversary is pretty difficult in this case, since the bound it not tight, which provides a relatively high value of OCO regret.

9. In the provided data none of the coins crashed. That is r i t > 0 for all i and all t. If there is a coin i and a time t such that r i t = 0 we run into trouble. Which assumption of Theorem 2 from the lecture notes is violated when this happens? What happens to the bound from the OCO regret from Theorem 2?

In theorem 2, the gradient of loss has been assumed to satisfy $\left\|\nabla l\left(a,z\right)\right\| \le G$. If one of the coins crashed, the gradient at that time would become infinitely large, which cannot be bounded and thus this assumption is violated. In this case the bound of the OCO regret will become infinitely large at the time when coins crashed.

Appendix

(a) OGD.m

```
% Exercise: Online Gradient Descent (OGD)

clear all;
load coin_data;
z=-log(r);

a_init = [0.2, 0.2, 0.2, 0.2, 0.2]'; % initial action

n = 213; % is the number of days
d = 5; % number of coins

% we provide you with values R and G.
alpha = sqrt(max(sum(r.^2,2)));
epsilon = min(min(r));
G = alpha/epsilon;
R = 1;

% set eta:
eta=R/(G*sqrt(n));
%%% your code here %%%

a = a_init; % initialize action. a is always a column vector

L = nan(n,1); % keep track of all incurred losses
A = nan(d,n); % keep track of all our previous actions

for t = 1:n

    % we play action a
    [l,g] = mix_loss(a,r(t,:)'); % incur loss l, compute
gradient g

    A(:,t) = a; % store played action
    L(t) = l; % store incurred loss

    % update our action, make sure it is a column vector
    %%% your code here %%%
    %calculate the prjected vector in A space
    w=a-eta*g;
    a=w;
```

```
%     v=rand(5,5);
%     for i =1:5
%        v(:,i) = project_to_simplex(v(:,i)');
%     end
%
%     S=GramSchmidt(v);
%     for i=1:5
%        a=a+(dot(w,S(:,i))/norm(S(:,i))^2)*S(:,i);
%     end
    % after the update, the action might not be anymore in the
action
    % set A (for example, we may not have sum(a) = 1 anymore).
    % therefore we should always project action back to the
action set:
    a = project_to_simplex(a')'; % project back (a = \Pi_A(w)
from lecture)

end


% compute total loss
%%% your code here %%%
L_total=sum(L);
% compute total gain in wealth
%%% your code here %%%
gain=exp(-L_total);
% compute best fixed strategy (you may make use of
loss_fixed_action.m and optimization toolbox if needed)
%%% your code here %%%
%%
A_fixed=zeros(5,50);
for j=1:50
a_fixed=rand(5,1);

a_fixed =transpose( project_to_simplex(a_fixed'));
loss=0;
alpha0=1e-5;
for k=1:10000
    loss0=loss;
    [loss, grad] = loss_fixed_action(a_fixed);
    a_fixed0=a_fixed;
    alpha=alpha0/(1+alpha0*10*k);
    a_fixed=a_fixed-alpha*grad;
    a_fixed =transpose( project_to_simplex(a_fixed'));
```

```
    dis=norm(a_fixed-a_fixed0,2);
    disloss=abs(loss-loss0);
    if disloss<1e-6&&k>1
        break
    end
end
disp(j)
A_fixed(:,j)=a_fixed;
end
a_optimal=sum(A_fixed,2)/50;
%%
% compute regret

Rn=L_total-min(sum(A-a_optimal,2));

%%% your code here %%%

%% plot of the strategy A and the coin data

% if you store the strategy in the matrix A (size d * n)
% this piece of code will visualize your strategy

figure
subplot(1,2,1);
plot(A')
legend(symbols_str)
title('rebalancing strategy OGD')
xlabel('date')
ylabel('investment action a_t')

subplot(1,2,2);
plot(s)
legend(symbols_str)
title('worth of coins')
xlabel('date')
ylabel('USD')
```

(b) mis_loss.m

```
function [l, g] = mix_loss(a, r)
%     [l, g] = MIX_LOSS(a, r)
%     Input:
%           a (column vector), the investment strategy (note a should be normalized so that
sum(a) = 1)
%           r (column vector), stock changes on day t (compared with t-1)
%     Output:
%           l (number), the mix loss
%           g (column vector), the gradient of the mix loss (with respect to action a)

     %%% your code here %%%
l=-log(a'*r);

g=-r/(a'*r);

end
```

(c) AA.m

```
% Exercise: Aggregating Algorithm (AA)

clear all;
load coin_data;

d = 5;
n = 213;

% compute adversary movez z_t
%%% your code here %%%
W=[s0;s];
r_t=zeros(n,5);
for i =1:n
    r_t(i,:)=W(i+1,:)./W(i,:);
end
z_t=-log(r_t);
% compute strategy p (see slides)
%%% your code here %%%
L_t=zeros(n,d);
C_t=zeros(n,1);
p=zeros(n,d);
for j=1:n
    L_t(j,:)=sum(z_t(1:j,:));
```

```
    C_t(j,:)=sum(exp(-L_t(j,:)),2);
    p(j,:)=exp(-L_t(j,:))./C_t(j,:);
end

% compute loss of strategy p
%%% your code here %%%
lmix_t=zeros(n,1);
for k=1:n
    lmix_t(k,1)=-log(sum(p(k,:)*transpose(exp(-z_t(k,:))),2));
end

% compute losses of experts
%%% your code here %%%
l_exp=zeros(n,d);
l_exp(1,:)=z_t(1,:);
for m=2:n
    l_exp(m,:)=sum(z_t(1:m,:));
end

% compute regret
%%% your code here %%%
Rn_AA=sum(lmix_t)-min(l_exp(213,:));
% compute total gain of investing with strategy p
%%% your code here %%%
gain_t=exp(-sum(lmix_t));
%% plot of the strategy p and the coin data

% if you store the strategy in the matrix p (size n * d)
% this piece of code will visualize your strategy

figure
subplot(1,2,1);
plot(p)
legend(symbols_str)
title('rebalancing strategy AA')
xlabel('date')
ylabel('confidence p in the experts')

subplot(1,2,2);
plot(s)
legend(symbols_str)
title('worth of coins')
xlabel('date')
ylabel('USD')
```