

## OMF assignment 2

Zihao Song  
2018CMF

II.

- a) Formulate the problem of minimizing the expected shortfall. Let  $L = £10100$ ,  $B = £10000$ ,  $x$  belongs to  $\mathbb{R}^n$  be the investment decisions which is the fraction of budget to invest in each asset,  $r_s^T$  be the one possible scenario  $s$  of asset returns and  $y$  be the shortfall, we get:

$$\begin{aligned} \min \mathbb{E}_s[sf_s] &= \min \mathbb{E}_s[\max\{L - Br_s^T x, 0\}] \\ \text{s.t. } Br_s^T x + y &\geq L; \sum_{i=1}^8 x_i = 1; x_i \geq 0. \end{aligned}$$

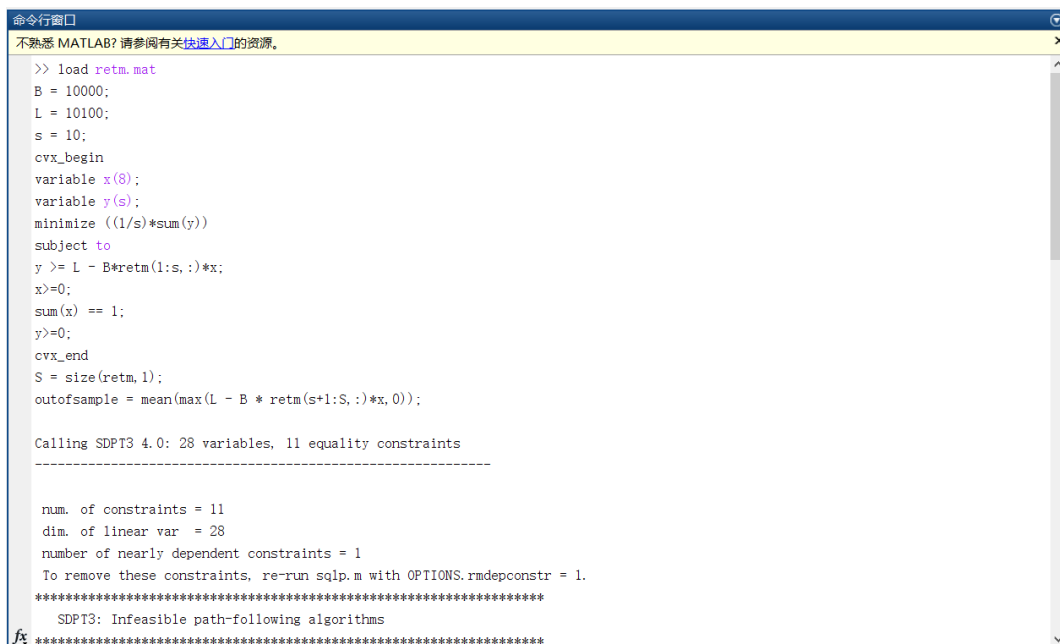
As for showing that it fits into the stochastic programming framework, let's consider: the Stochastic Recourse Problem and rewrite above as:

$$\begin{aligned} \min \mathbb{E}_s[y] &= \min \sum_{i=1}^s \frac{1}{s} * y_i \\ \text{s.t. } y_i &\geq L - Br_i^T x; \sum_{i=1}^8 x_i = 1; x_i \geq 0. \end{aligned}$$

Which contains the expectation item and the  $x$  in problem is stochastic and unknown.

Above means that it is the standard Stochastic Program with Recourse which fits into the stochastic programming framework.

- b) Actually, this is c). Report in-sample and out-of-sample expected shortfall. Shown as below is the pictures of codes and results:



```
命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。

>> load retm.mat
B = 10000;
L = 10100;
s = 10;
cvx_begin
variable x(8);
variable y(s);
minimize ((1/s)*sum(y))
subject to
y >= L - B*retm(1:s,:) * x;
x >= 0;
sum(x) == 1;
y >= 0;
cvx_end
S = size(retm,1);
outofsample = mean(max(L - B * retm(s+1:S,:) * x, 0));

Calling SDPT3 4.0: 28 variables, 11 equality constraints
-----

num. of constraints = 11
dim. of linear var = 28
number of nearly dependent constraints = 1
To remove these constraints, re-run sqp.m with OPTIONS.rmdepcnstr = 1.
*****
SDPT3: Infeasible path-following algorithms
fx *****
```

Figure 1 2c) part1

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。
dual   objective value = 6.52094970e+01
gap := trace(XZ)      = 2.45e-07
relative gap          = 1.87e-09
actual relative gap   = 1.53e-09
rel. primal infeas (scaled problem) = 3.77e-11
rel. dual " " " " " = 1.02e-11
rel. primal infeas (unscaled problem) = 0.00e+00
rel. dual " " " " " = 0.00e+00
norm(X), norm(y), norm(Z) = 5.2e+02, 5.0e+03, 3.9e+02
norm(A), norm(b), norm(C) = 8.8e+04, 3.2e+04, 1.3e+00
Total CPU time (secs) = 0.10
CPU time per iteration = 0.01
termination code      = 0
DIMACS: 1.2e-10 0.0e+00 1.2e-11 0.0e+00 1.5e-09 1.9e-09
-----

Status: Solved
Optimal value (cvx_optval): +65.2095

>> outofsample

outofsample =

    184.1255

fx >>

```

Figure2 2c) part2

Thus, expected shortfall could be seen in picture: in sample is 65.2095 and out of sample is 184.1225.

c) Actually, this is d). Same as above, calculate the in sample and out of sample:

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。
>> load retm.mat
B = 10000;
L = 10100;
s = 1000;
cvx_begin
variable x(8);
variable y(s);
minimize ((1/s)*sum(y))
subject to
y >= L - B*retm(1:s,:)*x;
x>=0;
sum(x) == 1;
y>=0;
cvx_end
S = size(retm,1);
outofsample = mean(max(L - B * retm(s+1:S,:)*x,0));

Calling SDPT3 4.0: 2008 variables, 1001 equality constraints
-----

num. of constraints = 1001
dim. of linear var  = 2008
number of dense column in A = 8
checkdeconstr: AAt is not pos. def.
*****
SDPT3: Infeasible path-following algorithms
*****
fx

```

Figure 3 2d) part1

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。
dual objective value = 1.41570986e+02
gap := trace(XZ)      = 1.22e-07
relative gap         = 4.30e-10
actual relative gap  = 3.27e-10
rel. primal infeas (scaled problem) = 1.06e-11
rel. dual   " " " " = 2.13e-11
rel. primal infeas (unscaled problem) = 0.00e+00
rel. dual   " " " " = 0.00e+00
norm(X), norm(y), norm(Z) = 8.9e+03, 5.7e+03, 1.1e+02
norm(A), norm(b), norm(C) = 9.0e+05, 3.2e+05, 1.0e+00
Total CPU time (secs) = 0.50
CPU time per iteration = 0.03
termination code      = 0
DIMACS: 3.4e-10 0.0e+00 2.2e-11 0.0e+00 3.3e-10 4.3e-10
-----
Status: Solved
Optimal value (cvx_optval): +141.571

>> outofsample

outofsample =

    145.2285

fx >>

```

Figure 4 2d) part2

Thus, expected shortfall could be seen in picture: in sample is 141.571 and out of sample is 145.2285.

### III.

- a) In order to find an optimal/good 10-scenario discretization, I choose to try clustering, as question says using the kmeans2.m MATLAB script from Lab Session 6.

Firstly, by the Algorithm (k-Means Clustering):

① I need to choose the desired number of clusters  $K$ , and in this case, aim is to find the good 10-scenario, thus, we need  $K = 10$ , i.e. separates the 100000 scenarios into 10 clusters.

② Then, start the k-mean and cvx for 100 times: Choose initial cluster centers randomly. I use: (from kmeans2.m)

```
rnd_idx = randsample(m, k);
```

```
mu = retm(rnd_idx, :);
```

Thus, mu contains all the 10 cluster centers I get at the beginning.

③ Using k-mean 20 times to find the approx. 10 centers and assign each point  $x_i$  to the nearest center, as written in code, firstly I need to cycle through points  $i$  from 1 to 100000 which at the beginning is initially assigned into cluster 1. Next is calculating the distance of point  $i$  from center of cluster 1 and cycle the number of center when computing the distance of point  $i$  from  $j$ -th center. If  $r < \min$ , i.e. if  $i$ -th point is closer  $j$ -th center to than all previous centers, then assign it to the cluster defined by center  $j$ , and new closest distance is  $r$ .

④ Generate new means as centers of the new clusters. Firstly, find indices of all points belonging to cluster  $j$  and calculate the length of indices, if it is zero, then keep above centers; else, then average all points belonging to cluster  $j$ , which could get the new centers.

⑤ At the end of ④, we get a  $10 \times 8$  matrix contained in  $\text{muex}$ , thus, we need to use cvx to solve the optimal program:

$$\min \mathbb{E}_s[y] = \min c./100000 * y$$

$$s.t. y_i \geq L - Br_i^T x; y_i \geq 0; \sum_{i=1}^8 x_i = 1; x_i \geq 0.$$

Where c contains the sizes of the clusters.

at the end of each k-mean, and store the problem results in a set. Thus, running the codes, finally get the results of all 20  $\min \mathbb{E}_s[y]$  and select the smallest one.

But in real case, I find that the results run quite slowly. That may be the k-mean, i.e. the high numbers of calculation (cycle by cycle). Eventually, I get the result as below:

**Thus, the smallest is the 17-th: 152.4184 and the investment decision is the 17th column in xx and the good 10-scenario is the 129 to 136 columns in muex, which is the figure 11 shows (8\*10 matric).**

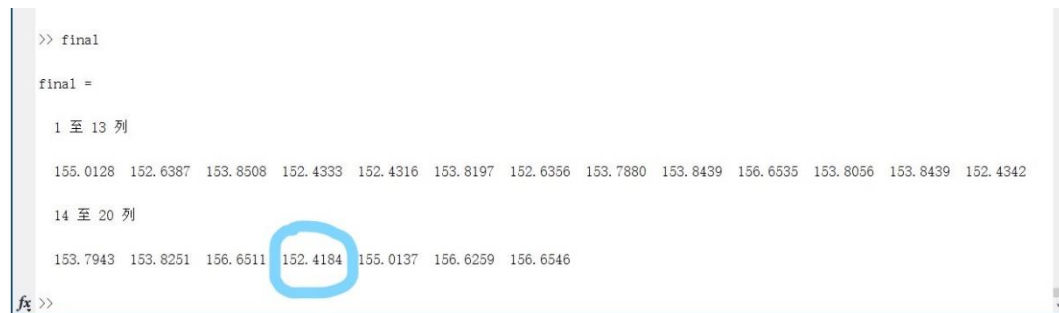


Figure 5 3a) result part1

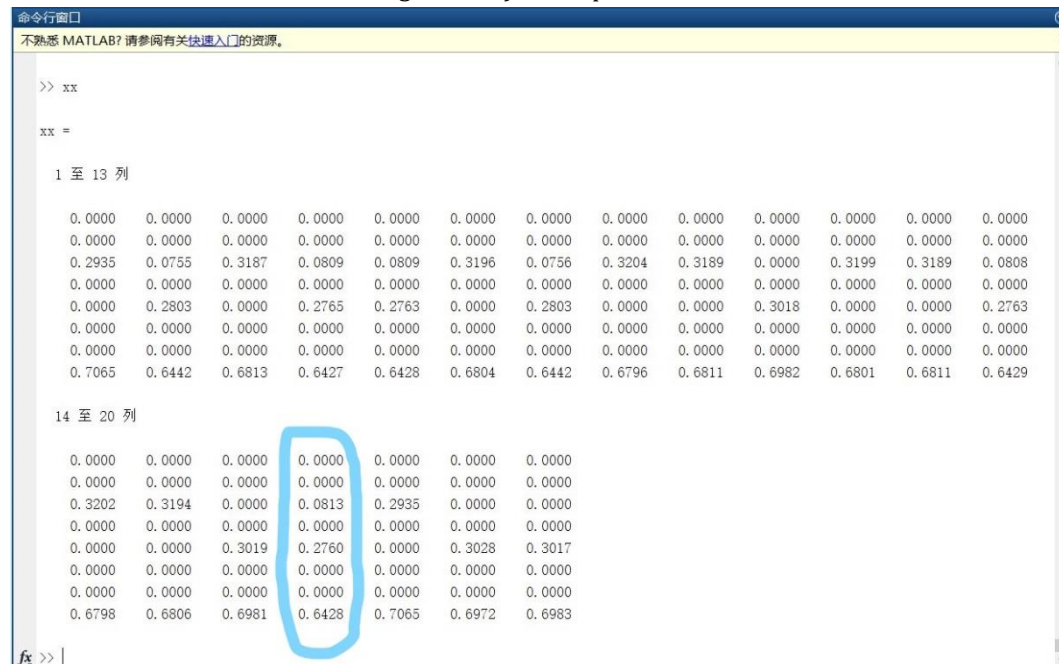


Figure 6 3a) result part2

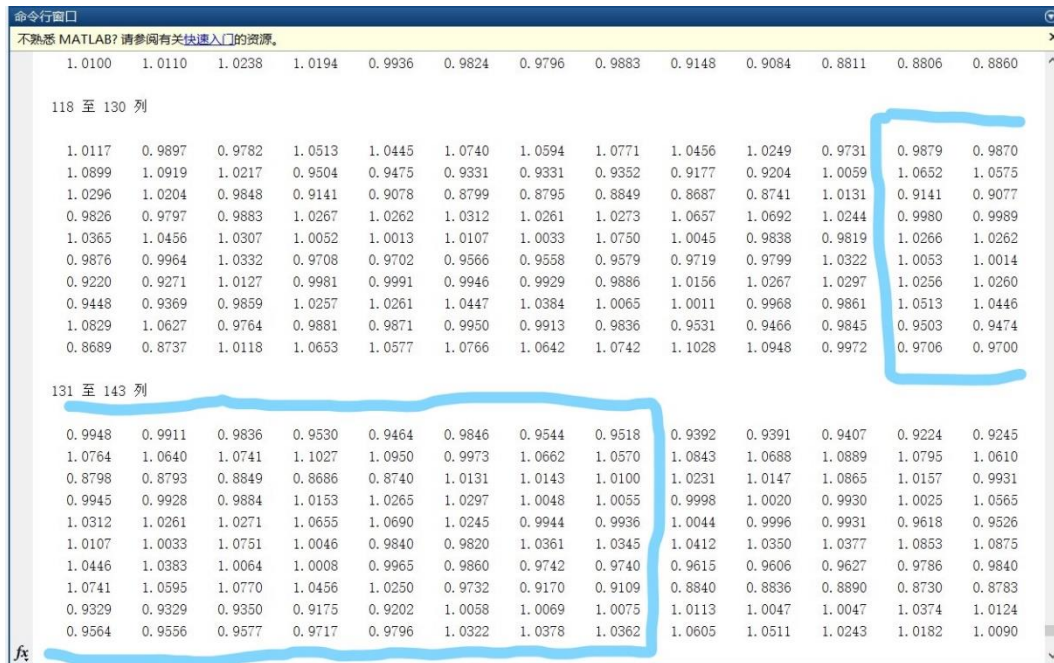


Figure 7 3a) result part3

At last, comment on the effort required in coming up with the scenarios, i.e. how much work would it be in general to apply in my method. Actually, when running the k-mean for 20 times, we need to calculate the steps during each k-mean. Set up the original conditions: 8 steps. During each k-mean: firstly, the randomly choose: 2 steps; then, k-mean cycle 200 times: calculate the min distance: 100000\*9 steps cycle contains 5 steps; next is changing the new centers: 10 steps cycle contains 6 steps; calculating the q matrix and running cvx 1 time, fulfilling the xx, muex and final 1 time. This is how much work I need to do in 3a).

- b) Now, using the SGM, I randomly choose the initial investment decision  $x$  and change the MATLAB scripts.

In RecCost, firstly define the cost function and the derivate of the cost function:

$y = L - B * rs * x$ ;  $dydx = - B * rs$ ; and in fact, I decide to do a complete sweep (i.e. use all scenarios in order), the cost and dcost are directly the same as  $y$  and  $dydx$ .

About the SGM, doing a complete sweep means cycling for 100000\*3 times and choose every line in retm to calculate the reccost, and also do the projection. Then, the final  $x$  could be the choice of our decision (as below), and then calculate the out of sample expected shortfall.

For reason why I choose to do 3 complete sweeps within retm is I have used 1, 2, 3, 4 and 5 complete sweep and when time is 3, I get the lowest out of sample shortfall. That may because when time is 3, the method gets the point result in the lowest shortfall during calculating the stochastic gradient.

Below is under the condition of calculating dcost using  $L = 1.01$ ,  $B = 1$ :

```
命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。
>> load retm.mat;
x = [0.1;
     0.15;
     0.2;
     0.05;
     0.09;
     0.16;
     0.11;
     0.14];
dcc = [];
xp = [];
L = 10100;
B = 10000;
for w = 1:3
    for p = 1:100000
        [cost, y, dc] = RecCost(x, retm(p,:));
        %dcc = [dcc dc];
        gamma = 1/p;
        x = x - gamma*dc;
        x = proj_unit_simplex(x);
        %xp = [xp x];
    end
end
sf = mean(max((L-B*retm*x),0));
```

Figure 8 3b) part1

The expected shortfall is 144.7868.

```
>> sf

sf =

    144.7868

>> x

x =

    0.2087
    0.1573
    0.0870
    0.0000
    0.0347
    0.0000
    0.0000
    0.5123
```

Figure 9 3b) part2

This is the best investment decision obtained in this way.

With experimenting the SGM for 1000 times and write down each terms' result, I could figure out that when w above from 1 to 1000, the expected shortfall run from 146 (1) to 144 (3) and then up to 145 (4-1000), floating during about 145.65.

Besides, I have noticed there are another two parts could slightly change the answer. One is during the reccost function, whether we use  $L = 10100$ ,  $B = 10000$  to calculate the dcost or use the  $L = 1.01$ ,  $B = 1$  and another is the initial point x.

For the first one, I check the dcost and the investment decision x I get when using each line of retm:

Below is under the condition of calculating dcost using  $L = 1.01$ ,  $B = 1$ : (I just choose several columns to show the law)

dc														
8x100000 double														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-0.9983	0	-0.9506	0	-0.9045	0	0	-0.9830	-0.9803	-1.0108	-1.0034	0	0	-0.980
2	-0.9737	0	-0.9288	0	-0.8951	0	0	-0.9788	-1.0143	-0.9661	-1.0045	0	0	-0.957
3	-0.9414	0	-0.9046	0	-0.9400	0	0	-0.9579	-0.9702	-0.9871	-1.0224	0	0	-0.945
4	-0.9548	0	-0.9005	0	-0.8921	0	0	-0.9563	-0.9950	-0.9387	-1.0346	0	0	-0.963
5	-0.9637	0	-0.9154	0	-0.8970	0	0	-0.9310	-0.9743	-0.9524	-1.0897	0	0	-0.976
6	-0.9287	0	-0.9123	0	-0.8896	0	0	-1.0576	-0.9988	-0.9743	-1.0183	0	0	-0.935
7	-0.9675	0	-0.9244	0	-0.9147	0	0	-1.0394	-1.0122	-0.9340	-1.0072	0	0	-1.002
8	-0.9824	0	-1.0340	0	-0.9830	0	0	-1.0160	-1.0118	-1.0074	-0.9534	0	0	-0.980
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														

Figure 10 3b)  $L = 1.01$ ,  $B = 1$  dcost

xp															
8x100000 double															
	651	99652	99653	99654	99655	99656	99657	99658	99659	99660	99661	99662	99663	99664	9
1	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	0.1530	
2	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	0.1687	
3	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	0.1644	
4	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	0.0064	
5	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	0.0725	
6	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	0.0480	
7	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	0.0561	
8	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	0.3309	
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															

Figure 11 3b)  $L = 1.01$ ,  $B = 1$  x

Below is under the condition of calculating dcost using  $L = 10100$ ,  $B = 10000$  (with the same initial point):

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。
x
dcc = [dcc dc];
gamma = 1/p;
x = x - gamma*dc;
x = proj_unit_simplex(x);
xp = [xp x];
end
sf = mean(max((L-B*retm*x), 0));
>> sf

sf =

    145.4840

>> x

x =

    0.2846
    0.0433
    0.0920
    0.0256
    0.0454
    0.0074
    0.0009
    0.5007

```

Figure 12 3b) L = 10100, B = 10000 shortfall and decision

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-9.9825e+03	0	-9.5057e...	0	-9.0454e...	0	-1.0144e...	-9.8300e...	-9.8032e...	-1.0108e...	-1.0034e...	0	0	-9.80...
2	-9.7371e+03	0	-9.2881e...	0	-8.9514e...	0	-1.0500e...	-9.7878e...	-1.0143e...	-9.6605e...	-1.0045e...	0	0	-9.57...
3	-9.4137e+03	0	-9.0460e...	0	-9.3996e...	0	-1.0139e...	-9.5793e...	-9.7024e...	-9.8706e...	-1.0224e...	0	0	-9.45...
4	-9.5480e+03	0	-9.0049e...	0	-8.9208e...	0	-1.0273e...	-9.5626e...	-9.9503e...	-9.3870e...	-1.0346e...	0	0	-9.63...
5	-9.6368e+03	0	-9.1542e...	0	-8.9698e...	0	-9.7709e...	-9.3096e...	-9.7432e...	-9.5243e...	-1.0897e...	0	0	-9.76...
6	-9.2870e+03	0	-9.1228e...	0	-8.8962e...	0	-1.0135e...	-1.0576e...	-9.9875e...	-9.7425e...	-1.0183e...	0	0	-9.35...
7	-9.6745e+03	0	-9.2444e...	0	-9.1470e...	0	-9.9851e...	-1.0394e...	-1.0122e...	-9.3395e...	-1.0072e...	0	0	-1.00...
8	-9.8238e+03	0	-1.0340e...	0	-9.8297e...	0	-1.0088e...	-1.0160e...	-1.0118e...	-1.0074e...	-9.5338e...	0	0	-9.80...
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														

Figure 13 3b) L=10100, B = 10000 dcost (same column as figure 10)



xp														变量 - xp
8x100000 double														
	99651	99652	99653	99654	99655	99656	99657	99658	99659	99660	99661	99662	99663	99664
1	0.2639	0.2655	0.2680	0.2680	0.2680	0.2670	0.2670	0.2665	0.2670	0.2670	0.2665	0.2708	0.2708	0.2698
2	0.0211	0.0202	0.0225	0.0225	0.0225	0.0222	0.0222	0.0211	0.0225	0.0225	0.0219	0.0255	0.0255	0.0247
3	0.0119	0.0105	0.0104	0.0104	0.0104	0.0101	0.0101	0.0122	0.0097	0.0097	0.0087	0.0119	0.0119	0.0103
4	0.0076	0.0047	0.0031	0.0031	0.0031	0.0037	0.0037	0.0043	0.0019	0.0019	9.5395e-04	0.0043	0.0043	8.6304e-04
5	0.0660	0.0667	0.0643	0.0643	0.0643	0.0658	0.0658	0.0658	0.0660	0.0660	0.0646	0.0651	0.0651	0.0773
6	0.0096	0.0051	0.0025	0.0025	0.0025	0.0058	0.0058	0.0068	0.0074	0.0074	0.0035	0.0027	0.0027	0.0014
7	0.0255	0.0241	0.0211	0.0211	0.0211	0.0213	0.0213	0.0231	0.0202	0.0202	0.0164	0.0114	0.0114	0.0049
8	0.5945	0.6032	0.6079	0.6079	0.6079	0.6041	0.6041	0.6000	0.6053	0.6053	0.6175	0.6084	0.6084	0.6109
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														

Figure 14 3b)  $L = 10100$ ,  $B = 10000$  x (same column as figure 11)

Easily we can figure out that during calculating dcost using  $L = 10100$ ,  $B = 10000$ , the stochastic gradient converges much slower than under the condition of  $L = 1.01$  and  $B = 1$ . For reason why, it may somehow because of the magnitude, as  $x$  starts from a vector each element below 1 and then calculate the gradient dcost (which is  $10^3$ , much higher than  $10^{-1}$ ), thus, for figure 15, it stays much stable than figure 18. As just change the  $L$  and  $B$  both in the same way, it could not change the answer, which could be figured out in figure 13 and 16. Also could be figured in the pictures is that about the decimal points: in the first column, for  $L = 1.01$  is  $-0.9983$  and  $L = 10100$  is  $-9982.5$ , and that may somehow change the speed we find the optimal point.

Another is the initial point, I change the initial point to below and under the same SGM  $L = 1.01$  and  $B = 1$ :

```

命令窗口
不熟悉 MATLAB? 请参阅有关快速入门的资源。
>> load retm.mat;
x = [0.1;
     0.15;
     0.2;
     0.05;
     0.1;
     0.15;
     0;
     0.25];
dcc = [];
xp = [];
L = 10100;
B = 10000;
for w = 1:3
    for p = 1:100000
        [cost, y, dc] = RecCost(x, retm(p,:));
        %dcc = [dcc dc];
        gamma = 1/p;
        x = x - gamma*dc;
        x = proj_unit_simplex(x);
        %xp = [xp x];
    end
end
sf = mean(max((L-B*retm*x), 0));
>> sf
fx sf =

```

Figure 15 3b) change initial part1

```

sf =
    144.8519

>> x

x =
    0.1974
    0.1436
    0.0821
    0.0000
    0.0359
    0.0000
    0.0000
    0.5410

```

Figure 16 3b) change initial part2

Similarly, quite small difference means that the stochastic gradient somehow finds the global optimal solution.

As for commenting on the computational effort compared to the scenarios-based approaches under 3a). Actually, when running the stochastic gradient for 3 times, we need to calculate the steps during each SGM. Set up the original conditions: 5 steps. During each stochastic gradient: firstly, cycling the scenarios in order in retm need 100000 steps; then, do the reccost function: 8 steps; next is calculating the new x (1 step) and project it into simplex: 8 steps; fulfilling the dc, xp and calculating the out of sample mean 1 time. This is how much work I need to do in 3b), which is much faster than 3a), may because it does not contain calculation cycle by cycle.