Politecnico di Torino

Corso di laurea magistrale in Ingegneria matematica

Cardinality constrained index tracking: mixed integer approaches

Studente
Graziana Mignone

Relatore
Paolo Brandimarte

I would like to dedicate this thesis to my loving parents ...

Abstract

We consider the problem of index tracking : reproducing the performance of a stock market index, but without purchasing all of the stocks that make up the index. We present two mixed-integer programming formulations of this problem that include a constraint limiting the number of stocks that can be purchased. One approach consists in minimizing the tracking error, which is the variance of the difference between tracking portfolio returns and index returns, the other approach consists in minimizing the CVaR of the negative deviation of the tracking portfolio returns from index returns. We use MATLAB and Gurobi and datasets drawn from real-life stock market indexes.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Problem statement

Index tracking is a popular form of passive fund management where the concern is to reproduce the performance of a stock index market. Tracking can be achieved by trying to hold all of the securities in the index, in the same proportions as the index(full replication). However full replication has a number of disadvantages, in particular certain stocks may be held in very small quantities; when the composition of the index is revised the holdings of all stocks will need to be changed and transaction costs associated with the subsequent sale and purchase of stocks may be high. Moreover in large indices (indices containing many stocks) the stocks associated with smaller companies may be relatively illiquid with consequent comparatively high transaction costs. Because of these disadvantages many passively managed funds,especially those that are tracking large indices, hold fewer stocks than are included in the index they are tracking and this is also our aim. We present two mixed-integer programming approaches to track a stock market index limiting the number of stocks that can be purchased. One approach consists in minimizing the tracking error, which is the variance of the difference between tracking portfolio returns and index returns, the other approach consists in minimizing the CVaR of the negative deviation of the tracking portfolio returns from index returns.

1.2 Dissertation Structure

The overall structure of the dissertation takes the form of six chapters, including this introductory chapter. Chapter 2 reviews literature and introduces a wide range of background topics. The third chapter is concerned with the problem formulation and the methodology

used for this study. The fourth chapter presents the findings of the study and discusses them. The final chapter concludes the dissertation and suggests some future work.

Chapter 2

Risk measures for active and passive portfolio management

2.1 Active and passive fund

In fund management, the wealth from numerous investors is managed. Managers have the aim of making decisions on how to achieve a gain on wealth avoiding losses.
There are two different type of fund management: active and passive.
In active fund management the aim is to try to compose a portfolio that performs better than the market. The expected return from the fund is higher than from the market at large, but also the probability of losses is relatively high. The aim of a passive fund manager instead, is to compose a portfolio that brings approximately the average market return, hence the expected return from the portfolio is lower than the one from active fund management, but passive fund management involves lower risk.
Active and passive fund managers have different attitudes to risk, with the passive manager being the most risk averse.
Passive fund management is commonly implemented by composing portfolios that simulate a chosen benchmark, typically a stock market index as in our case.
This strategy is referred to as *index tracking* and it is exactly what we want to do in this study.

2.2 Cardinality constrained index tracking : advantages

The objective of index tracking is to simulate a chosen index.
This aim can be achieved by full replication, that consists in investing in all the stocks of the index in the same proportions as in the index.

But in our study what we want to do is to limit the number of stocks to include in the tracking portfolio because of the different advantages this approach may cause.
First of all, when the number of stocks is restricted, administration costs of index tracking portfolios can be kept low. When the composition of the index is revised the holdings of all stocks will need to be changed and transaction costs associated with the subsequent sale and purchase of stocks may be high. Furthermore the stocks associated with smaller companies may be relatively illiquid with consequent high transaction costs.

2.3 Tracking error variance

Our aim is to seek to replicate an index, hence what we should do is to minimize the difference between the index return and the tracking portfolio return.
The *tracking error* is a way to measure this difference. Lots of formulations of the *tracking error* have been proposed.
We use the variance of tracking error: the variance of the difference between index return and tracking portfolio return. This is a standard approach that leads to a mixed integer quadratic problem that has a simple formulation and few variables.

2.4 CV@R

In the second model we do something different : we use, as risk-measure and as objective function to be minimized, the CV@R of the difference between the index return and the portfolio return (*loss function*).
We decided to use this measure for different reasons.
First of all because it is an asymmetric function, hence it allows to overcome the major limitation of *standard deviation* and *variance*, which is their symmetry: they measure dispersion but in both directions : positive and negative, so when we minimize them we minimize not only the probability to have high losses but also the probability to have high profits.
Secondly, it satisfies interesting properties that characterize a *coherent risk measure*.

A *single-period risk measure* is a functional $\xi(\cdot)$ mapping a random variable $X(\omega)$ to the real line.

The properties of a *coherent risk measure* are:

- **Normalization.** If a random variable is identically zero, $X \equiv 0$, then the the risk associated to this variable is zero, $\xi(0) = 0$. (If we do not hold any portfolio, we are not exposed to any risk).

- **Monotonicity.** If $X_1 \leq X_2$, then $\xi(X_1) \geq \xi(X_2)$.
  If we interpret the random variable X as a return, it is easy to understand that if the return of the portfolio 1 is always less than or equal to the return of the portfolio 2, then the risk associated to the portfolio 1 is at least as the risk associated to portfolio 2. Investing in portfolio 2 should be less risky because his return is always higher or equal to the return of portfolio 1.

- **Translation invariance.** $\xi(X+a) = \xi(X) - a$, where a is a constant.
  This means that if we add a fixed amount of money to the portfolio, the risk measure is reduced (if $a > 0$) or increased (if $a < 0$) by the same amount.

- **Positive homogeneity.** $\xi(bX) = b\xi(X)$
  If we increase the amount of money invested in a portfolio, then also the risk associated to this portfolio increases in the same proportion.

- **Subadditivity.** $\xi(X+Y) \leq \xi(X) + \xi(Y)$.
  The risk associated to the sum of two random variables is not higher than the risk associated to the sum of respective risks. (We know that diversification is expected to decrease the risk).

These are all interesting properties, but what makes the CV@R more attractive from a computational point of view are the *positive omogeneity* and *subadditivity* that can be combined into a **convexity condition**:

$$\xi(\lambda X + (1-\lambda)Y) \leq \lambda\xi(X) + (1-\lambda)\xi(Y), \qquad \forall\lambda \in [0,1].$$

In standard mean-risk models we use variance or standard deviation as risk measures, but they are not coherent and not convex.

They satisfy the property of normalization, but they are not translation invariant, in fact:

$$Var(X+a) = Var(x)$$

They don't satisfy the property of monotonicity.

The positive homogeneity condition also fails because:

$$Var(bX) = b^2 Var(X)$$

but

$$\sqrt{Var(bX)} = b\sqrt{Var(X)},$$

hence this condition is met by standard deviation.

For what concerns subadditivity, we have that variance is subbaditive only in some cases, when the covariance between the random variables is less than or equal to zero :

$$Var(X_1 + X_2) = Var(X_1) + Var(X_2) + 2Cov(X_1, X_2)$$

Standard deviation instead, is subbadditive:

$$\begin{aligned}\sigma_{X_1+X_2} &= \sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2 + 2\rho_{12}\sigma_{X_1}\sigma_{X_2}} \\ &\leq \sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2 + 2\sigma_{X_1}\sigma_{X_2}} \\ &= \sigma_{X_1} + \sigma_{X_2}\end{aligned}$$

.

The standard deviation looks a bit better than variance as risk measure.

The CV@R is a quantile-based risk measure. Another well-known quantile-based risk measure is the V@R, which is a quantile of the probability distribution of loss:

$$V@R_{1-\alpha} = min\{x : F_X(x) \leq 1 - \alpha\}$$

where F is the CDF: cumulative distribution function of X.

But V@R suffers from different limitations: for example it cannot distinguish between different tail shapes and it cannot tell us anything about what append on the right side of the value of the quantile. Furthermore it is not a coherent measure because it is not subbadditive.

CV@R let us to overcome all these limitations.

CV@R, conditional value-at-risk, is an asymmetryc and coherent risk measure related to tail expectations, and it is defined as the expected value of losses over a time horizon T exceeding the V@R value:

$$CV@R_{1-\alpha,T} = E[L_T | L_T > V@R_{1-\alpha,T}].$$

Chapter 3

Our aim is to track an index using a limited number of assets. The cardinality-constrained tracking portfolio may include at most C_{max} assets. In order to express the cardinality constraint, we need to introduce binary variables y_i, one for each asset, modeling the inclusion of asset i in the tracking portfolio:

$$y_i = \begin{cases} 1 \text{ if the stock i is included in the portfolio} \\ 0 \text{ otherwise} \end{cases}$$

Binary variables y_i and continuous variables x_i should be linked by a so-called "big-M" constraint such as

$$x_i \leq M y_i,$$

where M is a suitably large constant, such that when $y_i = 1$, the constraint is non-binding for x_i. In our case, when x_i is a portfolio weight for a portfolio where short-selling is forbidden, we can set $M = 1$.

Let us consider the following notation:

n is the number of the components of the index
x_i is the fractions of wealth allocated to the asset i
y_i is a boolean variable that is equal to 1 if we invest in the i-th stock, 0 if we don't invest in it
C_{max} is the maximum number of stocks that we want to include in the tracking portfolio
t is the number of observations
τ_t is the rate of return of the index at time t
r_{it} is the rate of return of asset i at time t

3.1 Formulation of the Tracking Error Model

A tracking error minimization problem subject to a constraint on the total number of assets can be formulated as following:

$$\begin{aligned} &min\ TE \\ &s.t.\ \sum_{i=1}^{n} x_i = 1 \\ &\sum_{i=1}^{n} y_i \leq C_{max} \\ &x_i \leq y_i \quad \forall i = 1..., n \\ &x_i \geq 0 \quad \forall i = 1..., n \\ &y_i \in \{0,1\} \quad \forall i = 1..., n \end{aligned}$$

As we said before, there are different formulations of the tracking error, but in our study we use the following one:

$$(Bx - \tau)'(Bx - \tau)$$

where B is the matrix containing the returns of index components, hence the number of its rows is equal to the number of observations: t, and the number of columns is equal to the total number of stocks: n;

τ is the vector containing the index returns , and x is the vector of the tracking error portfolio weights .

The decision variables are x_i and y_i and they are subject to the following constraints:

- $\sum_{i=1}^{n} x_i = 1$: the sum of our portfolio weights is 1;
- $\sum_{i=1}^{n} y_i \leq C_{max}$: the total number of stocks in our tracking portfolio is at most C_{max};
- $x_i \leq y_i$: this constraint relates x_i with y_i
- $x_i \geq 0$: we rule out short selling , hence portfolio weights are set to be non negative;
- $y_i \in \{0,1\}$: we set yi to be a boolean variable

Since the TE is a quadratic function, this problem is a MIQP problem (Mixed Integer Quadratic Programming Problem), hence it is computationally complex, but there is an optimizer software that let us solve this kind of problems : Gurobi.

3.2 Formulation of Mean-CvaR model

(Inserire la matematica relativa al cvar)

Let us consider the following *loss function* :

$$L(x,t) = \tau_t - r_t$$

where $r_t = \sum_{i=1}^{n} r_{it} x_i$ is the tracking portfolio return at time t, hence $L(x,t)$ is the difference between the index return and the portfolio return at time t.

Let us define:

$$z_t = max\{0, L(x,t)\}$$

So, the formulation of the problem is:

$$min \quad \zeta + \frac{1}{\alpha} \sum_{t=1}^{n} p_t z_t$$

$$z_t \geq -\sum_{i=1}^{n} r_{it} x_i + \tau_t - \zeta \quad \forall t = 1..., T$$

$$z_t \geq 0 \quad \forall t = 1..., T$$

$$s.t. \sum_{i=1}^{n} x_i = 1$$

$$\sum_{i=1}^{n} y_i \leq C_{max}$$

$$x_i \leq y_i \qquad \forall i = 1...,n$$

$$x_i \geq 0 \qquad \forall i = 1...,n$$

$$y_i \in \{0,1\} \qquad \forall i = 1...,n$$

The decision variables are :

x_1, y_i for $i = 1....,n$, z_t for $t = 1...,T$ and ζ, where ζ is the VaR.

Furthermore we assume p_t, which is the probability of z_t, equal to $\frac{1}{T}$, where T is the total amount of observations.

The first and the second constraints are the linear form of $z_t = max\{0, L(x,t)\}$.

3.3 Methodology

The methodology used is the following: first of all we drew data from internet. This data contain information about the FTSE MIB index values and its component prices from 2012 until 2018. The dataset has been divided into smaller ones, containing each one information about two years, so we ended up with five datasets. Then, each dataset has been further splitted in two parts : the training set and the test set. The former has been used to train the model and to get the optimal tracking portfolio and the latter has been used to evaluate the performance of the optimal portfolio against the benchmark (the index). The same approach has been used for the other model and then the results of the two models have been compared.

3.4 Tracking Error Model Code

As we can see at the following link : $http://www.gurobi.com/documentation/7.5/refman/matlab\_solving\_models\_with.html\#matlab:solving$, Gurobi notation is based on Matlab matrices and it allows us to solve problems of this form:

$$\begin{aligned} &minimize && x^t Qx + c^t x + alpha \\ &s.t. && Ax = b \end{aligned}$$

$$l \leq x \geq u$$

$$some\ x_j\ integral$$

$$some\ x_k\ lie\ within\ second\ order\ cones$$

$$x^t Qcx\ +\ q^t x\ \leq beta \qquad \text{(second order constraint)}$$

$$some\ x_i\ in\ SOS \qquad \text{(special oder set constrains)}$$

Our *objective function* is :

$$(Bx-\tau)'(Bx-\tau) =$$
$$= (x^t B^t - \tau^t)(Bx-\tau) =$$
$$= x^t B^t Bx - x^t B^t \tau - \tau^t Bx + \tau^t \tau$$

Since $x^t B^t \tau = (Bx)^t \tau = \tau^t (Bx)$ because it is a scalar vector product, we get :

$$x^t B^t Bx - 2\tau^t Bx + \tau^t \tau$$

Hence, according to Gurobi notation, we have :

$$Q = B^t B$$
$$c = -2\tau^t B$$
$$alpha = \tau^t \tau$$

First of all we build this matrices in Matlab:

```
rng(1);

load Assets_values;    %Assets_values is the matrix containing stock prices from
```

```
                        %2012 untill 2018
load Index_values;      %Index_values is the vector containg index values
                        %from 2012 to 2018
load('time.mat')        % time.mat is the vector containg dates




ri=price2ret(Index_values);   % ri is the vectore of index returns


A=Assets_values(1:505,:);       % From Assets_values we select
                                % only data regarding two years
ri=ri(1:504);                   % We do the same for index values

Ar1=price2ret(A);               % Ar1 = matrix of stock returns
```

Once we have prepared the dataset to work with, we split it in two subdatasets: the training set, used for our model, and the test set.

```
[t,n]= size(Ar1);   % n = number of index components
                        = size of vector x;
                    % t = number of observations
time1=time(1:t+1);

time2=time(1:t);
m=t;
t=2*round( m/3);   % We set the size of training set to be
                   % the two third of the whole dataset
l=m-t;
rint=ri;
ri=rint(1:t);
```

```
rl=rint(t+1:m);
timer=time2;
time2=timer(1:t);
timel=timer(t+1:m);

Ar=Ar1(1:t,:);   % We select only the first two third                    %


Q=Ar'*Ar;       % Q=quadratic term of objective function;
                 % since our decision variables include not
                 % only the vector x but also the vector y
                 % we need to adjust the matrix Q
```

Observation: in our problem, the decision variables are not only the stock weights of the tracking portfolio, but also the boolean variables y_i. Hence we have to adjust the matrix Q we defined before and also the vector c in such a way that, if we substitute the vector x in (*) with the vector of decision variables containing both x and y, we get the same result as in (*) .

$$\overline{Q} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}$$

```
Q=[Q,zeros(n,n);zeros(n,2*n)];

time1=time(1:t+1);

time2=time(1:t);


alpha=ri'*ri;    % alpha=constant term of objective function;

c=-2*ri'*Ar;     % c=linear term of objective function;


c=[c,zeros(1,n)];   % We adjust c;
```

Now we need to build the matrix A of linear constraints.

```
r1= [ones(1,n),zeros(1,n)]; % r1 = first row of A
```

The first row of A corresponds to the constraint on x components that limits the sum of portfolio weights to be 1 ($\sum_{i=1}^{n} x_i = 1$) .

```
r2= [zeros(1,n), ones(1,n)];
```

The second row of A corresponds to the constraint limiting the number of stocks ($\sum_{i=1}^{n} y_i \leq K$).

Now we need to write the constraint : $x_i \leq y_i$ that is equal to : $x_i - y_i \leq 0$. This corresponds to a matrix made up by two adjacent diagonal matrices. The former is equal to the identity matrix, the latter is equal to negative identity matrix.

```
r3= [eye(n),-eye(n)];

A=[r1; r2; r3];



k=15;   % k = maximum number of stocks in the portfolio

b=[1;k;zeros(n,1)];
```

Now we can use Gurobi notation.

```
 clear model;
```

```
model.Q=sparse(Q);      % In Gurobi all the matrices
                        % must be sparse
model.A = sparse(A);
model.obj = c';         % Linear term of objective function
model.rhs =b ;
model.sense(1) = '=';   % We set the sense of the first constraint
model.sense(2:25)='<';  % The sense of the other constraints
model.vtype(1:n)='C';   % The first n variables are
                         % continuous
model.vtype(n+1:2*n)='B'; % The other variables are
                           % boolean
model.alpha=alpha;        % Constant term of objective function




result = gurobi(model);
```

Let's show some results.

```
fprintf('Obj: %e\n', result.objval); % result.objval  is the minimum
                                     % value of the objective function

for v=1:2*n
fprintf('%e\n', result.x(v));
end



xsol=result.x(1:n);          % xsol = solution for portfolio weights
```

```
    ysol=result.x(n+1:2*n);     % solution for boolean vector y


    num_gurobi_2012_1=sum(xsol>0); % total number of stocks in the
                                   % optimal portfolio

    save('num_gurobi_2012_1.mat','num_gurobi_2012_1');

    port=zeros(m,1);

for i=1:m
    port(i)= Ar1(i,:)*xsol;   % port = vector containing
end                           % optimal portfolio returns
                              % for the whole period of two years
```

Some statistics.

```
meanport=mean(port)  % We compute the mean and the standard deviation
                     % of our portfolio and of the index
stdport=std(port)

meanri=mean(rint)
stdri=std(rint)

figure(1)
hist(port)    % Portfolio return has a normal distribution


cdfplot(port(1:t))    % We compute the empirical
                      % comulative distribution function
cumsum_port=cumsum(port(1:t));

figure(2)
```

```
hist(ri)
cdfplot(ri)
cumsum_ri=cumsum(ri);



figure(3)                       % We plot portfolio returns
                                % first for the period of
                                % the training set and then
                                % for the period of the
                                % test set.

plot(time2,port(1:t),'blue')

hold on

plot(time2,ri, 'green' )

figure(4)                 % We do the same considering
                          % cumulative returns

plot(time2,cumsum_port,'blue')

hold on

plot(time2,cumsum_ri, 'green' )

figure(5)

plot(timel,port(t+1:m),'blue')

hold on

plot(timel,rl, 'green' )
```

```
cumsum_rl=cumsum(rl);
cumsum_portl=cumsum(port(t+1:m));

figure(6)

plot(timel,cumsum_portl,'blue')

hold on

plot(timel,cumsum_rl, 'green' )



numassetsy=sum(ysol==1);
numassetsx=sum(xsol~=0);

cumsum_port_train_gurobi_2012_1=cumsum_port;
cumsum_ri_train_gurobi_2012_1=cumsum_ri;

cumsum_port_test_gurobi_2012_1=cumsum_portl;
cumsum_ri_test_gurobi_2012_1=cumsum_rl;

save('cumsum_port_train_gurobi_2012_1','cumsum_port_train_gurobi_2012_1');
save('cumsum_port_test_gurobi_2012_1','cumsum_port_test_gurobi_2012_1');

save('cumsum_ri_train_gurobi_2012_1','cumsum_ri_train_gurobi_2012_1');
save('cumsum_ri_test_gurobi_2012_1','cumsum_ri_test_gurobi_2012_1');


port_train=port(1:t);   % optimal portfolio returns for training set data
port_test=port(t+1:m);  % optimal portfolio returns for test set data

%We compute the mean and the standard deviation of portfolio
%returns with training set data and test set data

meanport_trainingset=mean(port(1:t));
```

```
stdport_trainingset=std(port(1:t));

meanri_trainingset=mean(ri);
stdri_trainingset=std(ri);


meanport_testgset=mean(port(t+1:m));
stdport_testset=std(port(t+1:m));

meanri_testset=mean(rl);
stdri_testset=std(rl);

Gurobi_indiciAzioni2012_parte1=find(xsol~=0); %indexes of
                                              %stocks in the optimal portfolio

t_esecuzione_Gurobi_2012_1=result.runtime;  % execution time
TE_trainingset_Gurobi_2012_1=result.objval;

sumtetrain=0;
sumtetest=0;

% We compute the tracking error for both the portfolio
%with training data and the portfolio with test data

for i=1:t

   sumtetrain=sumtetrain+ (ri(i)-port(i))^2;

end

tetrain=sumtetrain/(t-1); % Tracking error computation

for i=t+1:m

   sumtetest=sumtetrain+ (rint(i)-port(i))^2;
```

```
end

tetest=sumtetest/(m-t);


port_train_gurobi_2012_1=port_train;
port_test_gurobi_2012_1=port_test;

save('port_train_gurobi_2012_1.mat', 'port_train_gurobi_2012_1');

save('port_test_gurobi_2012_1.mat', 'port_test_gurobi_2012_1');

pesi_gurobi2012_1=xsol;

save('pesi_gurobi2012_1','pesi_gurobi2012_1');

TE_gurobi_train2012_1=tetrain;
TE_gurobi_test2012_1=tetest;

save('TE_gurobi_train2012_1','TE_gurobi_train2012_1');
save('TE_gurobi_test2012_1','TE_gurobi_test2012_1');
```

3.5 CVaR Model Code

First of all we set the name of the problem and the decision variables.

```
cvarprob = optimproblem; % name of the problem

xvars = optimvar('xvars',t,1,'Type','continuous','LowerBound',0,'UpperBound',1 );
% portfolio weights
yvars = optimvar('yvars',t,1,'Type','integer','LowerBound',0,'UpperBound',1);
% boolean variables
zvars = optimvar('zvars',n,1,'Type','continuous','LowerBound',0);
%  variable z
eps=optimvar('eps','Type','continuous','LowerBound',0); %eps is the VaR
```

Secondly, we set the constraints :

```
Sommauno= sum(xvars)==1;   % constraint on x

NumAzioni=sum(yvars)<=k;   % constraint on y

Xy= optimconstr(t,1);


for i= 1:t

    Xy(i)= xvars(i)<=yvars(i);

end


Zetan = optimconstr(n,1);   % constraint on z

for i= 1:n

    s=0;
    for h= 1:t
        s=s+Assetsr(i,h)*xvars(h);
    end        %uso il ciclo perchè facendo il prodotto tra vettori ho errore

    Zetan(i) = zvars(i) >= ( ri(i) - eps - s );

end



cvarprob.Constraints.Sommauno = Sommauno;

cvarprob.Constraints.NumAzioni= NumAzioni;
```

```
cvarprob.Constraints.Xy= Xy;

cvarprob.Constraints.Zetan=Zetan;



alpha= 0.05;

% we set the objective function
cvarprob.Objective = eps + 1/alpha* 1/n * sum(zvars);

sol= solve(cvarprob);
```

Now we do the same analysis we have done before with the results of the previous model.

```
xsol=sol.xvars
ysol=sol.yvars

sum(sol.xvars>0) % number of stocks in the optimal portfolio
sum(sol.yvars)

port=zeros(m,1);

for i=1:m

    port(i)= Assetsr(i,:)*xsol;

end

meanport=mean(port(1:n))
stdport=std(port(1:n))
```

```
meanri=mean(ri(1:n))
stdri=std(ri(1:n))

port_train=port(1:n);
port_test=port(n+1:m);

figure(6)
hist(port(1:n))

cdfplot(port)
cumsum_port=cumsum(port(1:n));

figure(7)

hist(ri)
cdfplot(ri)
cumsum_ri=cumsum(ri);

figure(8)

plot(time2,port(1:n),'blue')

hold on

plot(time2,ri, 'green' )


figure(9)

plot(time2,cumsum_port,'blue')

hold on

plot(time2,cumsum_ri, 'green' )
```

```
figure(10)

plot(timel,port(n+1:m),'blue')

hold on

plot(timel,rl, 'green' )

cumsum_rl=cumsum(rl);
cumsum_portl=cumsum(port(n+1:m));

figure(11)

plot(timel,cumsum_portl,'blue')

hold on

plot(timel,cumsum_rl, 'green' )


indiciAzioni2012_1=find(xsol~=0); % indexes of the stocks
                                  % in the optimal
                                  % portfolio

% Tracking error computation

sumtetrain=0;
sumtetest=0;

for i=1:t

   sumtetrain=sumtetrain+ (ri(i)-port(i))^2;

end

tetrain=sumtetrain/(t-1);
```

```
for i=t+1:m

   sumtetest=sumtetrain+ (rint(i)-port(i))^2;

end

tetest=sumtetest/(m-t);


port_train_cvar_2012_1=port_train;
port_test_cvar_2012_1=port_test;

save('port_train_cvar_2012_1.mat', 'port_train_cvar_2012_1');
save('port_test_cvar_2012_1.mat', 'port_test_cvar_2012_1');


pesi_cvar2012_1=xsol;

save('pesi_cvar2012_1','pesi_cvar2012_1');

TE_cvar_train2012_1=tetrain;
TE_cvar_test2012_1=tetest;

save('TE_cvar_train2012_1','TE_cvar_train2012_1');
save('TE_cvar_test2012_1','TE_cvar_test2012_1');
```

Chapter 4

Results and discussion of experiments

4.1 TE-model results and CVaR-model results comparison

We want to choose a tracking portfolio that performs well in-sample, but we also hope that the chosen portfolio will perform well out-of-sample. For this reason we decided to split each dataset in two parts: the training set (in-sample) to fit the model and the test set (out-of-sample) to evaluate the future performance of the chosen portfolio.

Table 4.1 In-sample k=15.

	2012-2014		**2013-2015**	
	CVaR	*TE*	*CVaR*	*TE*
TE	$3.0049e^{-04}$	$1.5679e^{-04}$	$2.3792e^{-04}$	$2.9328e^{-04}$
Std	0.0087	0.0078	0.0128	0.0099
μ	$9.0807e^{-04}$	$9.2340e^{-04}$	-0.0023	$2.1019e^{-04}$
skewness	-0.0761	-0.2764	-0.0789	-0.2840
kurtosis	3.0663	3.3721	3.0963	3.2457
# stocks	7	9	10	9

Table 4.2 In-sample k=15.

	2014-2016		**2015-2017**	
	CVaR	*TE*	*CVaR*	*TE*
TE	$4.1943e^{-04}$	$5.5305e^{-04}$	$8.5289e^{-04}$	$3.6548e^{-04}$
Std	$0.0197e^{-04}$	0.0132	0.0105	0.0121
μ	$6.7777e^{-04}$	$5.8207e^{-04}$	$-6.0401e^{-04}$	$2.9513e^{-04}$
skewness	-0.3846	-0.3170	-0.3782	-0.3952
kurtosis	4.0488	4.1576	4.5677	5.0806
# stocks	7	6	7	9

Table 4.3 In-sample k=15.

	2016-2018	
	CVaR	*TE*
TE	$1.1699e^{-04}$	$2.0484e^{-04}$
Std	0.0088	0.0080
μ	-0.0041	$2.3240e^{-04}$
skewness	0.1421	-0.1231
kurtosis	4.7175	4.1043
# stocks	8	10

Table 4.4 Out-of-sample k=15.

	2012-2014		**2013-2015**	
	CVaR	*TE*	*CVaR*	*TE*
TE	$1.3860e^{-05}$	$3.1280e^{-04}$	$1.2843e^{-05}$	$5.8581e^{-04}$
Std	0.0102	0.0097	0.0135	0.013
μ	$-2.5457e^{-04}$	$-1.5902e^{-04}$	$5.9222e^{-04}$	$9.8137e^{-04}$
skewness	-0.3312	-0.1786	-0.3421	-0.3394
kurtosis	3.8250	3.3041	3.6025	4.5887
# stocks	7	9	10	9

Table 4.5 Out-of-sample k=15.

	2014-2016		**2015-2017**	
	CVaR	*TE*	*CVaR*	*TE*
TE	$1.9057e^{05}$	0.0011	$3.8832e^{-05}$	$7.2981e^{-04}$
Std	0.0155	0.0120	0.0118	0.0075
μ	$4.3660e^{-04}$	$4.1266e^{-04}$	$5.0366e^{-04}$	$6.7536e^{-04}$
skewness	-0.5904	-0.5473	-0.0942	-0.0587
kurtosis	6.6678	7.5618	3.7950	4.5209
# stocks	7	6	7	9

Table 4.6 Out-of-sample k=15.

	2016-2018	
	CVaR	*TE*
TE	$6.1955e^{-06}$	$4.0938e^{-04}$
Std	0.0091	0.0096
μ	$2.9780e^{-04}$	$-4.3669e^{-04}$
skewness	-0.2259	-0.2424
kurtosis	3.8539	3.6447
# stocks	8	10

The tables show, for each dataset regarding a time period of 2 years, for each model (CVaR model and TrackingError model), and for In-sample data and Out-of-sample data, the information about the optimal portfolio : the tracking error (TE), standard deviation (Std), mean (μ), skewness (nota), kurtosis (nota) and number of stocks selected (# stocks).

Since the aim of the TE-model is to minimize the tracking error, we expect that, at least for in-sample data, the tracking error of the tracking portfolio we get with TE-model is less than the TE of the portfolio we obtain with the CVaR model, but we can notice from the tables that, for as concerns the in-sample data, the tracking error for the two models are more or less the same.
For as concern out-of-sample data, instead, the TE of TE optimal portfolio is always higher than the TE of CVaR portfolio. The two portfolios (TE-portfolio and CVaR portfolio) have more or less the same TE and tend to follow the index movements in the same way even if the composition is different. This is quite evident in the figures.

The figures show the cumulative returns of tracking portfolios and of the index for In-sample data and Out-of-sample data.

The *red* curve represents the cumulative return of the the TE-model portfolio, the *blue* curve represents the cumulative return of CVaR-Model portfolio and the *green* curve represents the cumulative return of the index.
From the figures is quite evident that tracking portfolios perform always better than the index, not only with the In-sample data, where we expect the portfolio to perform well because it is the solution of our optimal model, but also with out-of-sample data and also when the index show very low and decreasing returns such as in Fig. 4.6

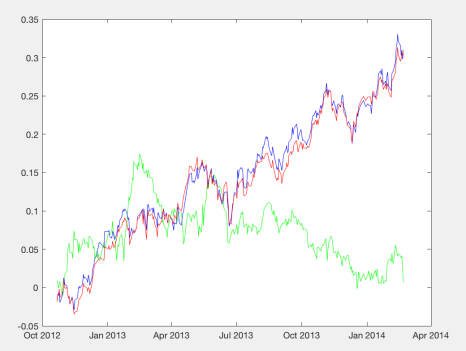

Fig. 4.1 In-sample dataset 1

Fig. 4.2 Out-of-sample dataset 1

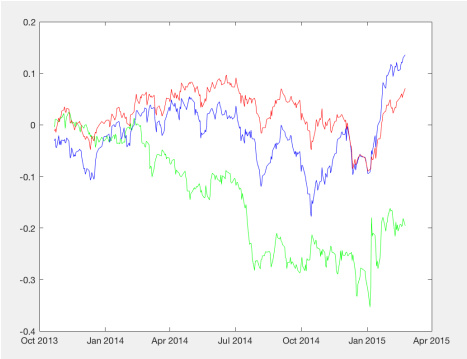

Fig. 4.3 In-sample dataset 2

Fig. 4.4 Out-of-sample dataset 2

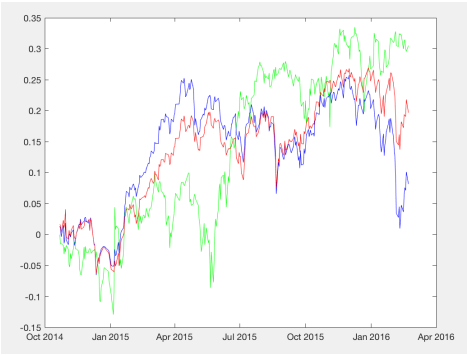

Fig. 4.5 In-sample dataset 3

Fig. 4.6 Out-of-sample dataset 3

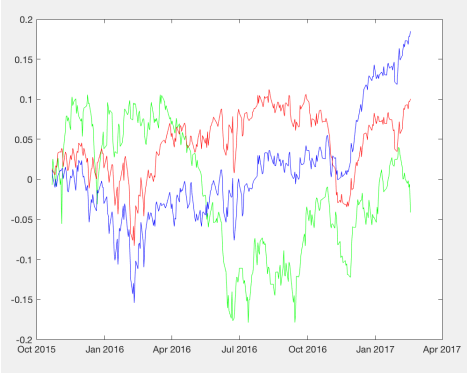

Fig. 4.7 In-sample dataset 4

Fig. 4.8 Out-of-sample dataset 4

Fig. 4.9 In-sample dataset 5

Fig. 4.10 Out-of-sample dataset 5

Another evident observation is that our optimal portfolios are *less volatile* than the index.

While the CVaR portfolio has a lower tracking error, the TE-portfolio has a higher return (on average) and the same level of risk (they have in all cases similar standard deviation). So TE portfolio is more attractive than CVaR portfolio. Another advantage of TE-model is the *execution time* which is much lower than the execution time of CVaR model. (aggiungi tabella con tempi di esecuzione)

Table 4.7 2012-2014

	Index		**CVaR-model**		**TE-model**	
	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
μ	$2.0032e^{-05}$	-0.0011	$9.0807e^{-04}$	$-2.5457e^{-04}$	$9.2340e^{-04}$	$-1.5902e^{-04}$
std	0.0095	0.0114	0.0087	0.0102	0.0078	0.0097

Table 4.8 2013-2015

	Index		**CVaR-model**		**TE-model**	
	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
μ	$-5.8170e^{-04}$	0.0012	-0.0023	$5.9222e^{-04}$	$2.1019e^{-04}$	$9.8137e^{-04}$
std	0.0143	0.0187	0.0128	0.0135	0.0099	0.0130

Table 4.9 Stocks included in TE-portfolios

2012-2014	**2013-2015**	**2014-2016**	**2015-2017**	**2016-2018**
2	2	2	1	1
3	3	3	2	2
5	4	4	3	8
8	5	5	4	10
10	8	8	8	11
11	11	12	10	12
12	12		11	13
18	18		12	17
19	19		20	19
				22

Table 4.10 Stocks included in CVaR-portfolios

2012-2014	**2013-2015**	**2014-2016**	**2015-2017**	**2016-2018**
2	1	2	2	2
3	2	4	3	3
8	4	5	4	8
10	5	8	8	11
12	8	9	10	12
13	11	12	11	14
20	12	23	12	17
	14			21
	18			
	20			

We can see from the tables that the number of stocks that compose the TE portfolio and CVaR portfolio differs always by one unit or two, in the worst case. Furthermore, if we investigate on this stocks, we can see that also the stocks chosen for the two portfolios are more or less the same with little variations.

4.2 Varying k

The number of stocks k in the tracking portfolio is a user defined parameter. One of the advantage of this approach is that the user can easily vary k, to see the effect of changing the number of stocks that they choose to hold. For the first experiments we use $k = 15$, which is more or less equal to half of the total number of stocks in the index. We can easily notice from tables that all the optimal portfolios contain a number of stocks that is always less than or equal to 10, more precisely the number of selected stocks is always between 7 and 10.
In order to understand what happen if we restrict drastically the number of stocks, we do some experiments setting k=5.

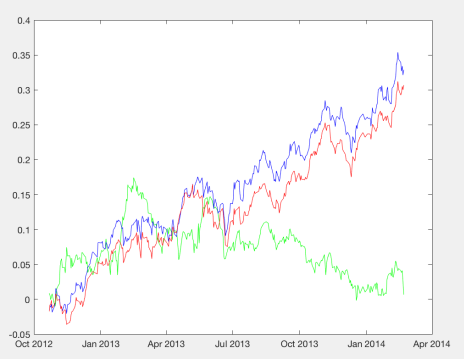

Fig. 4.11 In-sample dataset 1

Fig. 4.12 Out-of-sample dataset 1

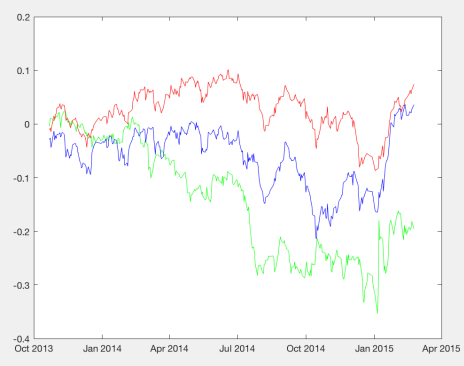

Fig. 4.13 In-sample dataset 2

Fig. 4.14 Out-of-sample dataset 2

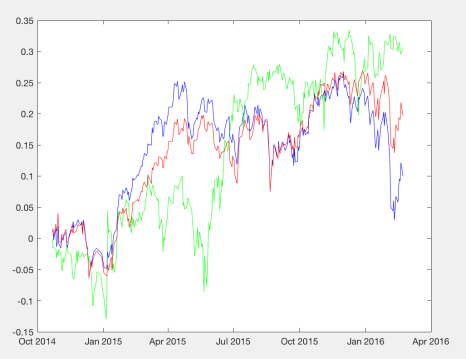

Fig. 4.15 In-sample dataset 3

Fig. 4.16 Out-of-sample dataset 3

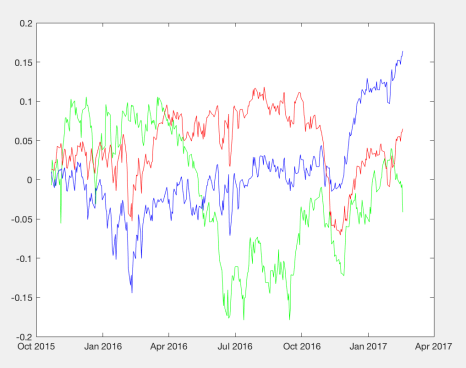

Fig. 4.17 In-sample dataset 4

Fig. 4.18 Out-of-sample dataset 4

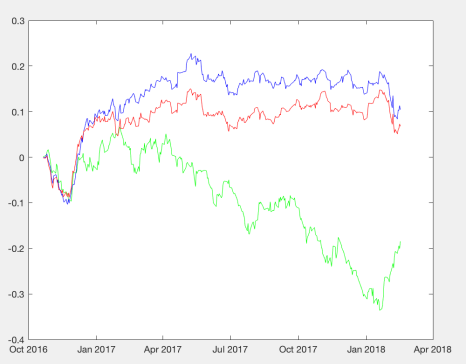

Fig. 4.19 In-sample dataset 5

Fig. 4.20 Out-of-sample dataset 5

Chapter 5

Conclusions and future work

5.1 Future work

The results of our models have proved to be quite positive with portfolio performances, in the most cases, good and better than the index performance.
But we know that actually it is quite difficult to have this results because in real life there are lots of factors that come into play such as transaction costs and feeds.
Since the aim of the models we have shown is to restrict the number of stocks to include in a portfolio, which is itself something that implicitly goes in the direction of reducing the transaction costs, it could be interesting to introduce in the previous models a constraint limiting transaction costs.

References