Project: Playing against market

1.1 Objectives of the first stage

1. Learn how to download stock prices from different data sources using Python library pandas-datareader

2. Learn how to compute daily stock returns using Pandas library

3. Learn how to analyze stock returns –

a. check for normality,

b. compute KL-divergence between some normal distribution and actual distribution of returns of a stock. Solve minimization problem (minimize KL-divergence) in order to find parameters of the normal distribution that your data follows.

4.Learn how to find a long-only portfolio with maximum return under worst possible market scenario. Portfolio is defined by weights of each stock.

5.Learn how to compute sensitivity of the portfolio risk to a change in certain stock weight. (for example, if additional restrictions for purchasing certain stocks are applied).

1.2 Objectives of the second stage

1. Learn how to compute daily drawdown for the portfolio

2. Learn how to formulate a problem of portfolio selection with minimum drawdown constraints

3. Learn how to solve a portfolio selection problem with drawdown constraints

2 Background

Imagine that we want to invest in financial securities. In addition, we want to investigate the possibilities and outcomes of investing into different financial products. In other words, we want to analyze possible investment strategies. There are several mathematical formulations of this problem that are easy to interpret as games with market.

We may have the following information on daily returns of market instruments given by *market experts*

|  |  |  |  |
| --- | --- | --- | --- |
| Table 2.6 Annual returns on stocks under different market conditions | | | |
| Market | Bonds | Stocks | MM | |
| Low | 0.0500 | -0.03 | 0.0067 | |
| Neutral | 0.04 | 0.003 | 0.067 | |
| Medium High | 0.03 | 0.05 | 0.067 | |
| High | 0.0250 | 0.082 | 0.067 | |

or we may only have historical scenarios:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table 2.6 Daily returns for portfolio of stocks (from Dow Jones Index) | | | | | |
| Date | V | PG | CSCO | MCD |
| 1/5/2019 | 0.032 | 0.012 | -0.004 | 0.007 |
| 1/6/2019 | 0.029 | 0.003 | -0.007 | 0 |
| 1/7/2019 | 0.039 | 0.004 | 0.004 | -0.004 |
| 1/8/2019 | -0.01 | 0.011 | 0.005 | 0.002 |
| 1/11/2019 | -0.012 | 0.061 | -0.003 | 0.018 |
| 1/12/2019 | -0.008 | -0.03 | -0.016 | -0.006 |
| 1/13/2019 | 0.012 | 0.001 | 0.018 | -0.008 |

For this example we take *historical scenarios* as inputs and find the combination of weights in the stock portfolio that is expected to deliver the maximum return under the worst market scenario.

This is a *data-driven* (or *model-free*) approach to portfolio selection - no need to imply a distribution of returns as in , say, Markovitz model of portfolio selection. This may be considered advantageous in the area of machine learning and artificial intelligence applications.

A modern portfolio theory (a problem of portfolio selection) was developed by Harry Markowitz (1952, Nobel Prize in Economics, 1990). It is called mean-variance analysis and it is a quadratic model that is very sensitive to changes in data. The main idea of this model is selecting a portfolio that maximizes expected return under given level of risk. Many improvements to this model were suggested that address major problems of the model. One type of improvement is made by considering Value at Risk (VaR) of the portfolio as a measure of risk as opposed to variance as in the classical model. This allows for model linearization and solves the problem of excessive data sensitivity. Toward this end, Expected Shortfall (or Conditional Value at Risk (CVAR) ) may be a better risk measure because it is coherent as opposed to VaR. It is noted however that solutions under VaR, ES, CVAR and variance result in the same optimal portfolio if returns are normally distributed, see, for example [1].

3 Data

On day the drawdown of a portfolio over 1 year history (252 business days) is computed as , where is a peak value of a portfolio over the same period

Drawdowns on each day in the past correspond to *the historical scenarios.*

4 Mathematical Model

The mathematical model includes the following:

n – number of stocks in the portfolio;

m – number of *historical scenarios*;

w – – weight of stock i

– value of the portfolio under scenario j;

Portfolio drawdown under scenario j:

is peak value of the portfolio under scenario j.

Objective: Find the best strategy (combination of weights) under the worst market scenario seen so far. Best strategy correspond to the *minimum of worst(maximum) drawdowns over all scenarios*

Such portfolio must have large drawdown and serves as upper bound of possible portfolio drawdowns under more careful selection.

Introducing new variable = , the objective is rewritten as follows

The expression means that for any row j in the data matrix (which represents the drawdowns of the stocks on day j) the weighted some of stock drawdowns is greater or equal . This is because we introduced as a minimum value of weighted sum of stock returns on any given day, the weighted sum of returns on day j is always greater or equal to the minimum

= 1, m (2.8)

, (2.9)

, = 1, n (2.10)

Constraint 2.9 means that the weights of stocks in the portfolio should sum up to 1. Constraint 2.10 means that each weight should be greater or equal to 0.

5 Mathematical Model in Matrix Form

*linprog* function from scikit-learn was already discussed in the previous section. Let us give more attention to input vectors and matrices:

objective: ,

s.t.

,

Let us look at the input data in matrix form. For our example the following parameters must be specified:

* c – vector of objective function coefficients
* – matrix of inequality constraints
* - RHS for inequality constraints
* – matrix of equality constraints
* – RHS of equality constraints
* bounds – bounds for each decision variable
* x is a decision vector – it will be computed as a result of optimization

In our example the decision vector is comprised of four stock weights and additional variable .

Let us rewrite the inequality constraint (2.8) from the previous section

[

The objective is rewritten as follows:

Rewrite RHS in matrix form:

Decision variables:

b\_eq

6 Python Code Sample for computing weights of optimal portfolio

This Python code reads stock data from “LogReturns1000v1.csv” file (Table 2.6) and computes stock weights that correspond to the best portfolio under worst market scenario.

import scipy.optimize as sopt

from numpy import genfromtxt

import numpy as np

data\_file = 'LogReturns1000v1.csv'

my\_data = genfromtxt(data\_file, delimiter = ',')

n=30

m=1000

A=np.ones((m,n+1))

A[0:m,0:n] = -my\_data[1:m+1,1:n+1]-2.0

c=np.zeros(n+1)

c[-1]=-1.0

b=np.zeros(m)

A\_eq=np.zeros((m,n+1))

A\_eq[0:1,0:n]=1.0

b\_eq=np.zeros(m)

b\_eq[0]=1.0

x0\_bounds = (0, None)

v\_bounds=(None,None)

bounds = [x0\_bounds]\*n

bounds.append(v\_bounds)

res=sopt.linprog(c,A,b,A\_eq,b\_eq,bounds)

print ('portfolio return', res.fun)

print ('optimal weights', res.x[0:n])

7 Analysis of the Solution

Table 2.7 shows how many shares of each stock we should keep in the portfolio.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 2.7 Optimal weights for each stock | | | | |
| CSCO | MCD | PG | V |
| 0.571 | 0.239 | 0.00 | 0.188 |

Figure 2.8 shows the optimal weights in graphical form.

|  |
| --- |
|  |
| Figure 2.8 Graph of optimal portfolio weights |

This is the Python code to create the graph:

#plot optimal portfolio weights

weights = list(res.x[:-1])

stock\_names = ['CSCO','MCD','PG','V']

y\_pos = np.arange(len(stock\_names))

print(weights )

plt.bar(y\_pos, weights , align='center', alpha=0.6)

plt.xticks(y\_pos, stock\_names,rotation='vertical')

plt.ylabel('Usage')

plt.title('Portfolio weights')

plt.show()

8 Questions and Further Analysis

1.We would like for stock 3 to have a minimum weight of 50% in the optimal portfolio. How do we add this constraint to the mathematical model and to a Python program?

2.We would like to analyze stock prices for certain historical period and decide if their distribution is close to a normal distribution with certain mean and variance. We formulate a minimization problem:

Find such mean and variance of normal distribution that minimize KL-divergence between given distribution of stock returns (distribution is given by its mean and variance) and normal distribution in question.

References

[1] R. Tyrrel Rockafellar, Stanislav Uryasev, Optimization of Conditional-Value-At-Risk

<http://uryasev.ams.stonybrook.edu/wp-content/uploads/2011/11/CVaR1_JOR.pdf>

Appendix A. Download daily closing prices for 5 Dow Jones stocks from yahoo, 4.5 year history

import pandas as pd

import pandas\_datareader as web

symbols = ['AAPL', 'MSFT', 'IBM', 'AABA', 'GLDI','V']

filename='DowJonesdata.csv'

noa = len(symbols)

data=pd.DataFrame()

for sym in symbols:

data[sym] = web.DataReader(sym, data\_source='yahoo',start='2015-01-19', end='2019-09-12')['Close']

data.columns = symbols

data.to\_csv(filename)