**Hegde Project Final Report**

**Group member: Weijia Deng,**

**Kai Yang,**

**Jianda Bi**

1. **Introduction**

Hedging is an important aspect of portfolio management. Portfolio managers need to select the hedging securities and associated hedge ratios to minimize a given risk metric of interest, such as the volatility of the portfolio, the VAR, CVAR, etc. It is not an easy task to choose the appropriate hedging securities and hedging method because there are so many choices in the market and different portfolio need different management. In this project, we are going to explore the best hedging method of our constructed portfolio and compare the hedging performance of different hedges (equities, bonds and gold).

1. **Summary of the process**

**First Step:** In this hedging project, we start by implementing the minimum variance hedge for a single stock (we choose IBM) and a market ETF (we choose the SPY ETF). After designing an initial hedging strategy, we examine how significantly the hedged portfolio reduces or increases several standard risk and performance metrics of the original stock and the hedged portfolio including the volatility, return, Sharpe Ratio, maximum drawdown, value-at-risk, beta of the portfolio to the market. Then, we investigate other three hedging strategies——minimization the variance of the return distribution, VAR, CVAR, to find the most appropriate method for our portfolio. We only do one hedge in this step using one-year price data from 2016-01-01 to 2016-12-31.

**Second Step:** After hedging for a single stock and single hedging security, we decide to hedge for a portfolio and a set of hedging securities. To do this, we construct a portfolio formed by stocks in a given sector (We choose finance sector), and hedge it with three hedging securities relative to the sector, that is SPY, IYF, CSJ, to form the hedged portfolio. We choose one-year price data of those securities from 2016-01-01 to 2016-12-31 and hedge every five days using the last three hedging methods that we discussed above. (The first minimum variance hedge method is too coarse to get accurate outcome) We then check some key risk metrics to find the best method for this portfolio.

**Third Step:** we consider a fixed income portfolio that consists of corporate bonds and hedge it with the appropriate hedging security which we think is TLO (SPDR Barclays Long Term Treasury ETF), to find that if it can get as significant risk reductions as we saw in the equity case above. For this part, we use an existing corporate bond ETF ——LQD (iShares iBoxx $ Investment Grade Corporate Bond ETF). Because the bonds are less liquid than equities and are traded in different days, it’s too difficult for us to form a corporate bonds portfolio and find the last price data without the help of any bond valuation systems.

**Final Step**: we explore the problem of hedging inflation. Gold is widely believed to be a good hedge against inflation. We assume that inflation (year on year CPI) is tradeable and try to hedge it with GOLDAMGBD228NLBM (Gold Fixing Price 10:30 A.M. (London time) in London Bullion Market, based in U.S. Dollars) to test the hedging performance of gold and compare the result with TIPS (hedging with iShares Barclays TIPS Treasury Inflation Protected Securities Bond Fund ETF) and SPY to see which one is a better hedge for inflation.

1. **Description of Data and Constructed Portfolios**

In the project, we use the pandas datareader to get all end of day data from google finance. And we construct 3 hedged portfolios.

The first one is Ct = St + h\* Ft, where S is IBM stock, and F is SPY futures.

The second one is Ct =Pt + h1\*F1 + h2\*F2 + h3\*F3

Where, P is the portfolio which is equally weighted formed among 10 stocks in financial sector. The 10 stocks are MS, PFK, BLK, CS, GS, ICE, MET, KCG, APO, FIG. F1 is the hedging security SPY, F2 is the hedging security IYF (iShares Dow Jones US Financial ETF) and F3 is the hedging security CSJ（iShares Barclays 1-3 Year Credit Bond ETF)

The third one is also Ct = St + h\* Ft, where S is LQD (iShares iBoxx $ Investment Grade Corporate Bond ETF) and F is TLO (SPDR Barclays Long Term Treasury ETF).

1. **Description of the models**

For models we use, first, we try the easiest hedging strategy ——minimum variance hedge.

Our model is:

**This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program.** , This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program.

This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program. is the combined hedged position——long IBM stock (This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program.) and long h\*SPY futures (This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program.), where h =hedge ratio. And we need to minimize the variance of Var (This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program.) to get the optimal hedge ratio.

Let, This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program.

For,This is the rendered form of the equation. You can not edit this directly. Right click will give you the option to save the image, and in most browsers you can drag the image onto your desktop or another program.

So, Var [ ΔS + h ΔF ] = σ2S  + h2σ2F - 2h(ρ σS σF)

Solving for h\*:  **h\* = - ρ(σS/σF)**

Where σS=standard deviation of ΔS, σF=standard deviation of ΔF, ρ=correlation between ΔS and ΔF.

Second, we use the other hedging strategy——minimum the variance of the return distribution

Ct = St + h\*Ft, Rt = (Ct-Ct-1)/Ct-1

And we need to minimize the variance of Rt to get the optimal hedge ratio h\*.

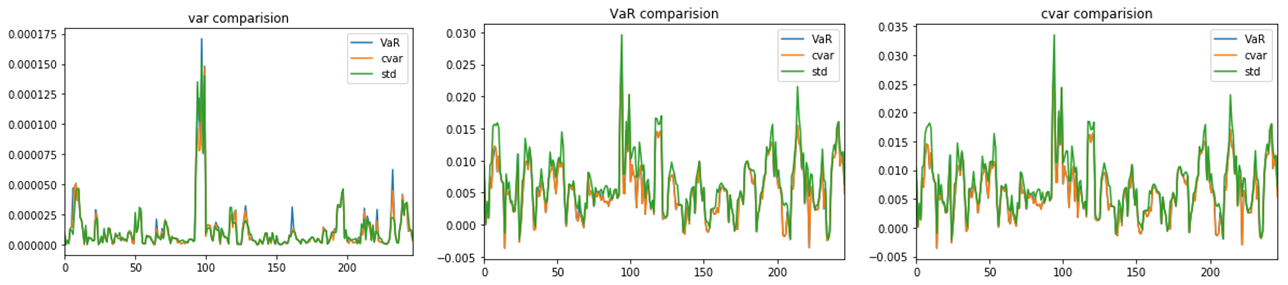
Third, we try the minimum VAR and CVAR method.

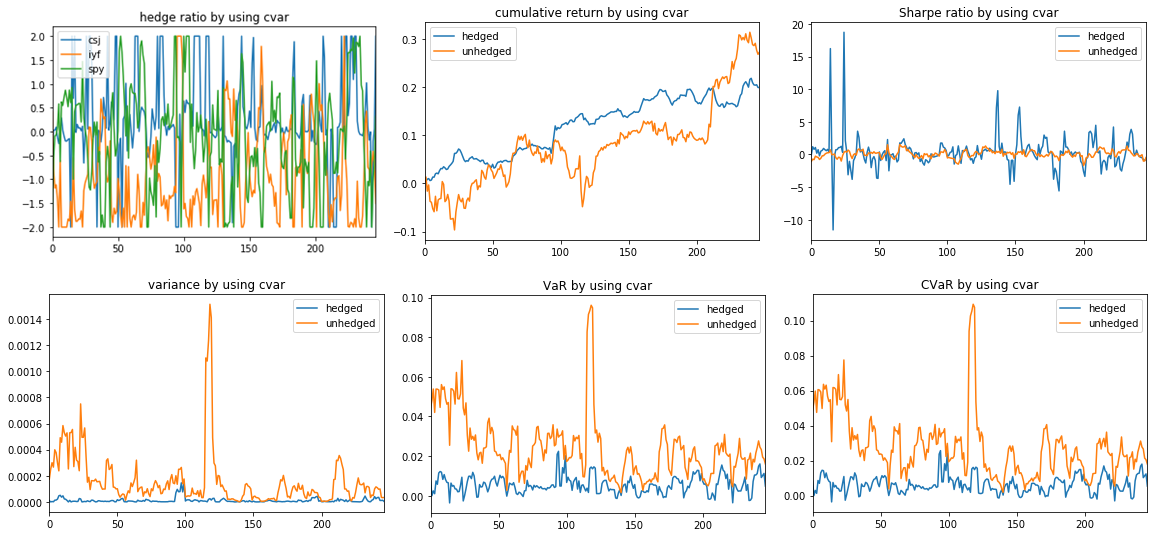
Ct = St + h Ft

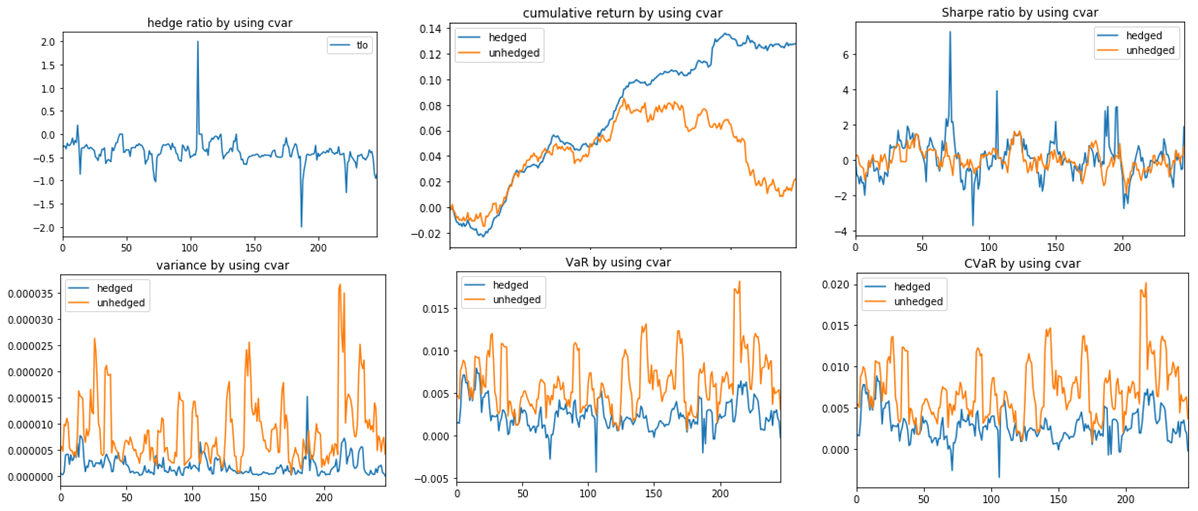
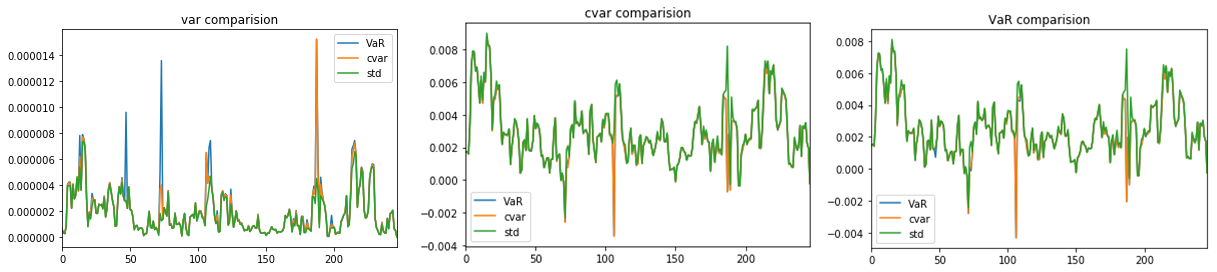
Separately minimize VAR(Ct) and CVAR(Ct) to get the h\*.

1. **Plots and Conclusions**

For first step, the hedge ratio for minimum variance hedge method is -0.66758009473738822, and the performance of hedged portfolio is better than the unhedged stock IBM. For minimum variance of return method, the hedge ratio is -0.92992296, the Var method is -0.90441702, and the CVAR method is -0.90509052.

For second step, we hedged for the stock portfolio. As mentioned before, minimizing variance of return, minimizing VAR and minimizing CVAR are three methods we conducted. All three methods worked well in terms of reducing standard risk and improving performance metrics. However, cumulative return ended up lower. And we decided to choose the method of minimizing CVAR for further analysis for hedged portfolio of this method performances best. Here are the performance metrics. Though return was lowered, risk was reduced.



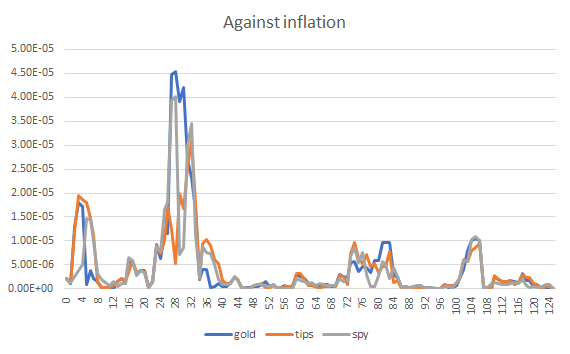
For third step, we hedged fixed income portfolio. For this step, we chose the method of minimizing CVAR. This method worked well. Not only performance metrics were improved, but also cumulative return ended up higher, which was surprising. Concluded directly from the graph, risk reduction here is not as significant as that in equity case. 

For final step, gold, TIPS and SPY were used to hedge the year-on-year CPI. Since CPI is not actually tradable and not profitable, we only focus on the variance of return, which measures the risk directly. All of the three hedges reduced risk in some level. The graph showed the variance of return of hedged portfolio after hedging in the three methods. Gold is not the best to hedge inflation. It was hard to determine TIPS or SPY is better merely from the graph.

Correlation between cpi and gold among the whole period: 7.551126740194910592e-03

Correlation between cpi and tips among the whole period: 2.424547369076580916e-03

Correlation between cpi and spy among the whole period: 8.346819022432366841e-03



1. **Appendix (code)**

**First Step,**

import datetime as dt  
import numpy as np   
import pandas as pd  
import matplotlib.pyplot as plt  
from scipy.optimize import minimize  
from pandas\_datareader import data, wb# Download data of ibm and spy from google #start\_date = dt.datetime(2016,1,1)   
end\_date = dt.datetime(2016,12,31)  
ibmp = pd.DataFrame(data.DataReader('ibm','google',start=start\_date,end=end\_date)['Close'])  
ibmpc = (ibmp-ibmp.shift(1)).dropna()  
  
spyp = pd.DataFrame(data.DataReader('spy','google',start=start\_date,end=end\_date)['Close'])  
spypc = (spyp-spyp.shift(1)).dropna()

# Calculate h\*#var1=np.var(spypc)var2=np.var(ibmpc)import mathstdspy=math.sqrt(var1)stdibm=math.sqrt(var2)covariance=np.cov(spypc['Close'],ibmpc['Close'])[0][1]correlation=covariance/(stdspy\*stdibm)hstar=-correlation\*(stdibm/stdspy)

price\_ibm = data.DataReader('ibm','google',start=start\_date,end=end\_date)['Close']price\_spy = data.DataReader('spy','google',start=start\_date,end=end\_date)['Close']price\_portfolio=price\_ibm-hstar\*price\_spyrtnibm = pd.DataFrame(price\_ibm.pct\_change().dropna())rtnport = pd.DataFrame(price\_portfolio.pct\_change().dropna()) # Performance metrics of ibm and hedged portfolio#rf=0.004from scipy.stats import norm#IBM#returns=rtnibmmu = np.mean(returns)std = np.std(returns)valueAtRisk\_ibm = norm.ppf(0.05, mu, std)volatility\_ibm=returns.std()\*np.sqrt(252)sharpe\_ratio\_ibm = (returns.mean() - rf) / volatility\_ibmcovariance1=np.cov(price\_spy,price\_ibm)[0][1]variance1=np.var(price\_spy)beta\_ibm=covariance1/variance1Roll\_Max = pd.rolling\_max(price\_spy, 252, min\_periods=1)Daily\_Drawdown = price\_spy/Roll\_Max - 1.0Max\_Daily\_Drawdown = pd.rolling\_min(Daily\_Drawdown, 252, min\_periods=1)Daily\_Drawdown.plot()Max\_Daily\_Drawdown.plot()#Portfolio#returns=rtnportmu = np.mean(returns)std = np.std(returns)valueAtRisk\_portfolio = norm.ppf(0.05, mu, std)volatility\_portfolio=returns.std()\*np.sqrt(252)sharpe\_ratio\_portfolio = (returns.mean() - rf) / volatility\_portfoliocovariance2=np.cov(price\_spy,price\_portfolio)[0][1]variance1=np.var(price\_spy)beta\_portfolio=covariance2/variance1Roll\_Max = pd.rolling\_max(price\_portfolio, 252, min\_periods=1)Daily\_Drawdown = price\_portfolio/Roll\_Max - 1.0Max\_Daily\_Drawdown = pd.rolling\_min(Daily\_Drawdown, 252, min\_periods=1)Daily\_Drawdown.plot()Max\_Daily\_Drawdown.plot()#Functions to get stock return data#def get\_stock\_rtn(start\_date,end\_date,ticker):  
 ''' ticker: 'SPY'  
 start\_date: dt.datetime(2011,1,4)   
 end\_date: dt.datetime(2017,5,1)  
 '''  
 price\_df = data.DataReader(ticker,'google',start=start\_date,end=end\_date)['Close']  
 rtndf = pd.DataFrame(price\_df.pct\_change().dropna())  
 rtndf.columns = ['return']  
 return rtndf

# Get IBM and SPY return#  
ticker = 'ibm'  
start\_date = dt.datetime(2016,1,1)   
end\_date = dt.datetime(2016,12,31)  
ibmdf = get\_stock\_rtn(start\_date,end\_date,ticker)  
ticker = 'spy'  
spydf = get\_stock\_rtn(start\_date,end\_date,ticker)#minimize the variance of the return distribution #from scipy.optimize import minimize  
def min\_var(h):  
 return np.var((ibmdf + h\*spydf)['return'])res = minimize(min\_var, 1, method='nelder-mead', options={'xtol':1e-8,'disp': True})res.x

# minimize the VAR #

from scipy.stats import normfrom scipy.optimize import minimize  
def min\_VAR(h):returns=ibmdf + h\*spydfmu = np.mean(returns)std = np.std(returns)alpha=0.01VAR = norm.ppf(1-alpha)\*std - mu return VARres = minimize(min\_VAR, 1, method='SLSQP', options={'disp': True})res.x

# minimize the CVAR #

from scipy.stats import normfrom scipy.optimize import minimizedef min\_CVAR(h): returns=ibmdf + h\*spydf mu = np.mean(returns) std = np.std(returns) alpha=0.01 CVAR=alpha\*\*-1\*norm.pdf(norm.ppf(alpha))\*std-mu return CVARres = minimize(min\_CVAR, 1, method='SLSQP', options={'disp': True})res.x

**Second Step**

import datetime as dt

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

from pandas\_datareader import data, wb

from scipy.optimize import minimize

from scipy.stats import norm

class Portfolio:

"""the hedge portoflio """

def \_\_init\_\_(self, pf\_sec, hedge\_sec, start, end , p\_weight):

"""Return a portfolio object which contain securities \*pf\_sec\* and hedge securities \*hedge\_sec\*."""

self.pf\_sec = pf\_sec

self.hedge\_sec = hedge\_sec

self.start = start

self.end = end

self.p\_weight = p\_weight

def port\_rtn(self):

"""get portfolio returns"""

data\_p = data.DataReader(self.pf\_sec,'google',start=self.start,end=self.end)['Close']

#portfolio price

pf\_p = np.sum(data\_p\*self.p\_weight,axis=1)

#portfolio returns

pf\_rtn= pd.DataFrame(pf\_p.pct\_change().dropna() ,columns=['return'])

return pf\_rtn

def hedge\_rtn(self):

"""get portfolio returns"""

#hedge securities price

hedge\_p = data.DataReader(self.hedge\_sec,'google',start=self.start,end=self.end)['Close']

#hedge securities returns

hedge\_rtn = pd.DataFrame(hedge\_p.pct\_change().dropna())

return hedge\_rtn

class Hedge:

"""

used for hedge purpose

"""

def \_\_init\_\_(self,port\_rtn, hedge\_rtn, n, lambd):

"""optimize portfolio and generating hedge ratio for specified time period

the formula for hedge is as following:

P\_final = Ret(portfolio) + lambda(hedge\_ratio\*hedge\_securities)

#securieties to be hedged

self.port\_rtn = port\_rtn

# hedge securieties

self.hedge\_rtn = hedge\_rtn

# rolling base(hedge every n day)

self.n = n

# lambda for hedge securities

self.lambd = lambd

self.num = self.hedge\_rtn.shape[1]

def by\_VaR(self):

'''optimize based on VaR'''

alpha=0.01

# minimize fuction

def min\_VaR(h):

p = self.port\_rtn['return']+np.dot(self.hedge\_rtn,np.array(h))\* self.lambd

return norm.ppf(1-alpha)\* np.std(p)-np.mean(p)

# bondary for hedge ratio

bon = tuple((-2, 2) for i in range(self.num))

# optimizarion

res = minimize(min\_VaR, tuple(0 for i in range(self.num)),

method='SLSQP',bounds=bon, options={'disp': True})

return res.x

def by\_cvar(self):

'''optimize based on cvar'''

alpha=0.01

# minimize fuction

def min\_cvar(h):

p = self.port\_rtn['return']+ np.dot(self.hedge\_rtn,np.array(h))\* self.lambd

return alpha\*\*-1\*norm.pdf(norm.ppf(alpha))\*np.std(p)-np.mean(p)

# bondary for hedge ratio

bon = tuple((-2, 2) for i in range(self.num))

# optimizarion

res = minimize(min\_cvar, tuple(0 for i in range(self.num)),

method='SLSQP',bounds=bon, options={'disp': True})

return res.x

def by\_std(self):

'''optimize based on std'''

# minimize fuction

def min\_std(h):

p = self.port\_rtn['return']+ np.dot(self.hedge\_rtn,np.array(h))\* self.lambd

return np.std(p)

# bondary for hedge ratio

bon = tuple((-2, 2) for i in range(self.num))

# optimizarion

res = minimize(min\_std, tuple(0 for i in range(self.num)),

method='SLSQP',bounds=bon, options={'disp': True})

return res.x

def roll(self,method,n, rf,pl):

ratio\_data = []

return\_data\_h = []

return\_data\_uh = []

sharpe\_ratio\_h = []

sharpe\_ratio\_uh = []

var\_data\_h = []

var\_data\_uh = []

VaR\_data\_h = []

VaR\_data\_uh = []

CVaR\_data\_h = []

CVaR\_data\_uh = []

print 'Processing....'

for i in xrange (0,n):

r\_pf=self.port\_rtn['return'][i:i+5]

r\_hedge=self.hedge\_rtn[i:i+5]

if method == 'VaR':

alpha=0.01

def VaR(h):

p = r\_pf+ np.dot(r\_hedge,np.array(h))\* self.lambd

return norm.ppf(1-alpha)\* np.std(p)-np.mean(p)

# bondary for hedge ratio

bon = tuple((-2, 2) for i in range(self.num))

res = minimize(VaR, tuple(0 for i in range(self.num)),

method='SLSQP',bounds=bon, options={'disp': False})

elif method == 'cvar':

alpha=0.01

def cvar(h):

p = r\_pf+ np.dot(r\_hedge,np.array(h)) \* self.lambd

return alpha\*\*-1\*norm.pdf(norm.ppf(alpha))\*np.std(p)-np.mean(p)

# bondary for hedge ratio

bon = tuple((-2, 2) for i in range(self.num))

res = minimize(cvar, tuple(0 for i in range(self.num)),

method='SLSQP',bounds=bon, options={'disp': False})

elif method == 'std':

def std(h):

return np.std(r\_pf+np.dot(r\_hedge,np.array(h))\* self.lambd)

# bondary for hedge ratio

bon = tuple((-2, 2) for i in range(self.num))

res = minimize(std, tuple(0 for i in range(self.num)),

method='SLSQP',bounds=bon, options={'disp': False})

else:

break

# return using last day data

ret\_h = (r\_pf+ np.dot(r\_hedge,np.array(res.x)) \* self.lambd)[4]

ret\_uh = r\_pf[4]

portfolio\_info = r\_pf+ np.dot(r\_hedge,np.array(res.x)) \* self.lambd

# sharpe ratio

sp\_h= (np.mean(portfolio\_info)-rf)/np.std(portfolio\_info)

sp\_uh= (np.mean(r\_pf)-rf)/np.std(r\_pf)

# variance

var\_h= np.var(portfolio\_info)

var\_uh= np.var(r\_pf)

# VaR

alpha=0.01

VaR\_h = norm.ppf(1-alpha)\* np.std(portfolio\_info)-np.mean(portfolio\_info)

VaR\_uh = norm.ppf(1-alpha)\* np.std(r\_pf)-np.mean(r\_pf)

# CVaR

CVaR\_h = alpha\*\*-1\*norm.pdf(norm.ppf(alpha))\*np.std(portfolio\_info)-np.mean(portfolio\_info)

CVaR\_uh = alpha\*\*-1\*norm.pdf(norm.ppf(alpha))\*np.std(r\_pf)-np.mean(r\_pf)

return\_data\_h.append(ret\_h)

return\_data\_uh.append(ret\_uh)

ratio\_data.append(res.x)

sharpe\_ratio\_h.append(sp\_h)

sharpe\_ratio\_uh.append(sp\_uh)

var\_data\_h.append(var\_h)

var\_data\_uh.append(var\_uh)

VaR\_data\_h.append(VaR\_h)

VaR\_data\_uh.append(VaR\_uh)

CVaR\_data\_h.append(CVaR\_h)

CVaR\_data\_uh.append(CVaR\_uh)

i=i+1

ratio\_data = pd.DataFrame(ratio\_data,columns=list(self.hedge\_rtn.columns))

return\_data\_uh = pd.DataFrame(return\_data\_uh,columns=['unhedged'])

return\_data\_h = pd.DataFrame(return\_data\_h,columns=['hedged'])

return\_data = pd.concat([return\_data\_h, return\_data\_uh], axis=1)

sharpe\_ratio\_uh = pd.DataFrame(sharpe\_ratio\_uh,columns=['unhedged'])

sharpe\_ratio\_h = pd.DataFrame(sharpe\_ratio\_h,columns=['hedged'])

sharpe\_ratio = pd.concat([sharpe\_ratio\_h, sharpe\_ratio\_uh], axis=1)

var\_data\_uh = pd.DataFrame(var\_data\_uh,columns=['unhedged'])

var\_data\_h = pd.DataFrame(var\_data\_h,columns=['hedged'])

var\_data = pd.concat([var\_data\_h, var\_data\_uh], axis=1)

VaR\_data\_uh = pd.DataFrame(VaR\_data\_uh,columns=['unhedged'])

VaR\_data\_h = pd.DataFrame(VaR\_data\_h,columns=['hedged'])

VaR\_data = pd.concat([VaR\_data\_h, VaR\_data\_uh], axis=1)

CVaR\_data\_uh = pd.DataFrame(CVaR\_data\_uh,columns=['unhedged'])

CVaR\_data\_h = pd.DataFrame(CVaR\_data\_h,columns=['hedged'])

CVaR\_data = pd.concat([CVaR\_data\_h, CVaR\_data\_uh], axis=1)

cumulative\_return\_uh = (return\_data\_uh+ 1).cumprod() - 1

cumulative\_return\_h = (return\_data\_h+ 1).cumprod() - 1

cumulative\_return\_uh.columns = ['unhedged']

cumulative\_return\_h.columns = ['hedged']

cumulative\_return = pd.concat([cumulative\_return\_h, cumulative\_return\_uh], axis=1)

if pl == 'all':

ratio\_data.plot(title = 'hedge ratio'+' by using '+str(method))

#return\_data\_h.plot(title = str(method)+' return')

#return\_data\_uh.plot(title = str(method)+' return')

return\_data.plot(title = 'return'+' by using '+str(method))

#cumulative\_return\_uh.plot(title = str(method)+' cumulative return')

#cumulative\_return\_h.plot(title = str(method)+' cumulative return')

cumulative\_return.plot(title = 'cumulative return'+' by using '+str(method))

#sharpe\_ratio\_h.plot(title = str(method)+' Sharpe ratio')

#sharpe\_ratio\_uh.plot(title = str(method)+' Sharpe ratio')

sharpe\_ratio.plot(title = 'Sharpe ratio'+' by using '+str(method))

var\_data.plot(title = 'variance'+' by using '+str(method))

VaR\_data.plot(title = 'VaR'+' by using '+str(method))

CVaR\_data.plot(title = 'CVaR'+' by using '+str(method))

return cumulative\_return\_h

elif pl == 'VaR':

VaR\_data.plot(title = 'VaR'+' by using '+str(method))

VaR\_data\_h.columns=[str(method)]

return VaR\_data\_h

elif pl == 'var':

var\_data.plot(title = 'variance'+' by using '+str(method))

var\_data\_h.columns=[str(method)]

return var\_data\_h

elif pl == 'cvar':

CVaR\_data.plot(title = 'CVaR'+' by using '+str(method))

CVaR\_data\_h.columns=[str(method)]

return CVaR\_data\_h

elif pl == 'cumulative':

cumulative\_return.plot(title = 'cumulative return'+' by using '+str(method))

cumulative\_return\_h.columns=[str(method)]

return cumulative\_return\_h

#Construct the portfolio#

sec = ['ms','pfk','blk','cs','gs','ice','met','kcg','apo','fig']

h\_sec = ['spy','iyf','csj']

start = dt.datetime(2016,1,1)

end = dt.datetime(2016,12,31)

portfolio\_weight = [1. /len(sec) for i in sec]

# set up data gathering class and gather data

default\_portfolio = Portfolio(sec,h\_sec,start,end,portfolio\_weight)

# return for hedge securities

hedge\_rtn = default\_portfolio.hedge\_rtn()

# return for securities to be hedged

port\_rtn = default\_portfolio.port\_rtn()

op = Hedge (port\_rtn,hedge\_rtn,n=1,lambd = 1)

#n = 5 # number of day or number of month

n = port\_rtn.shape[0]-5+1 # number of day or number of month (all data)

print 'period',n

rf = 0.1/365

pl='var' # 'all','var', 'cvar', 'VaR', 'cumulative'

total = pd.concat([op.roll('VaR',n,rf,pl),op.roll('cvar',n,rf,pl),op.roll('std',n,rf,pl)], axis=1)

**Third Step**

sec = ['lqd']

h\_sec = ['tlo']

start = dt.datetime(2016,1,1)

end = dt.datetime(2016,12,31)

portfolio\_weight = [1./len(sec) for i in sec]

default\_portfolio = Portfolio(sec,h\_sec,start,end,portfolio\_weight)

# return for hedge securities

hedge\_rtn = default\_portfolio.hedge\_rtn()

# return for securities to be hedged

port\_rtn = default\_portfolio.port\_rtn()

op = Hedge(port\_rtn,hedge\_rtn,n=1,lambd = 1)

**Final Step**

#gold#

start = dt.datetime(2006,1,1)

end = dt.datetime(2016,12,31)

cpi=data.DataReader('CPIAUCSL','fred', start, end )

gold=data.DataReader('GOLDAMGBD228NLBM','fred', start , end )

df = pd.concat([cpi,gold],axis=1)

df['GOLDAMGBD228NLBM']=df['GOLDAMGBD228NLBM'].fillna(method='pad')

df=df.dropna()

df=df.pct\_change().dropna()

df = pd.concat([cpi,gold],axis=1).pct\_change().dropna()

df.columns = ['return','gold']

# return for hedge securities

hedge\_rtn = pd.DataFrame(df['gold'])

# return for securities to be hedged

port\_rtn = pd.DataFrame(df['return'])

op = Hedge(port\_rtn,hedge\_rtn,n=1,lambd = 1)

# TIPS#

start = dt.datetime(2006,1,1)

end = dt.datetime(2016,12,31)

cpi=data.DataReader('CPIAUCSL','fred', start, end )

tip=data.DataReader('tip','google', start , end )['Close']

df = pd.concat([cpi,tip],axis=1)

df['Close']=df['Close'].fillna(method='pad')

df=df.dropna()

df=df.pct\_change().dropna()

##df = pd.concat([cpi,tip],axis=1).pct\_change().dropna()

df.columns = ['return','tip']

# return for hedge securities

hedge\_rtn = pd.DataFrame(df['tip'])

# return for securities to be hedged

port\_rtn = pd.DataFrame(df['return'])

op = Hedge(port\_rtn,hedge\_rtn,n=1,lambd = 1)

#SPY#

start = dt.datetime(2006,1,1)

end = dt.datetime(2016,12,31)

cpi=data.DataReader('CPIAUCSL','fred', start, end )

spy=data.DataReader('spy','google', start , end )['Close']

df = pd.concat([cpi,spy],axis=1)

df['Close']=df['Close'].fillna(method='pad')

df=df.dropna()

df=df.pct\_change().dropna()

#df = pd.concat([cpi,spy],axis=1).pct\_change().dropna()

df.columns = ['return','spy']

# return for hedge securities

hedge\_rtn = pd.DataFrame(df['spy'])

# return for securities to be hedged

port\_rtn = pd.DataFrame(df['return'])

op = Hedge(port\_rtn,hedge\_rtn,n=1,lambd = 1)

#Port\_rtn.shape#

op.by\_VaR()

op.by\_cvar()

op.by\_std()

# graph for hedge ratio change over month#

#n = 5 # number of day or number of month (test)

n = port\_rtn.shape[0]-5+1 # number of day or number of month (all data)

print 'period',n

rf = 0.1/365

pl='var' # 'all','var', 'cvar', 'VaR', 'cumulative'

total = pd.concat([op.roll('VaR',n,rf,pl),op.roll('cvar',n,rf,pl),op.roll('std',n,rf,pl)], axis=1)

# change return value if you want to see graph rather than culmulative return#

total.plot(title = str(pl)+' comparision')

corgold=np.correlate(df['return'],df['gold'])

cortips=np.correlate(df['return'],df['tip'])

corspy=np.correlate(df['return'],df['spy'])