智能投顾：理论与实践

Denis Frederic
Email: 78112407@qq.com

November 16, 2016

Abstract

本文简单回顾智能投顾（Robo-Advisor）的海内外发展情况，主要分析该投资方案的算法，并以 Python 模拟实现。

Contents

Chapter 1

介绍

所谓的智能投顾（Robo-Advisor），是指根据现代资产组合管理理论，利用机器学习等大数据处理技术，结合投资者的风险承受力与投资偏好，为其提供定制的资产投资方案。

由于管理费较低（0.20% 至 0.50% 左右），资产标的范围广泛，同时为客户提供避税服务，该业务在美国发展较为迅速。但该业务在国内基本没有实质发展。

1.1 海外发展情况

2010 年，第一家智能投顾公司 Betterment 成立于纽约。截至 2016 年 10 月末，该公司管理的资产规模约 60 亿美元。

目前，规模最大的智能投顾公司是 Vanguard 基金公司，智能投顾管理规模约 410 亿美元。但该公司管理的全部基金规模超过 35,000 亿元，其智能投顾的规模占比仅为 1.17% 。

结论：

（1）发展速度较快。从 2010 年至 2016 年，智能投顾公司管理的资产规模从无到有，到达了 3,000 亿（参见图 1.1）。

（2）占比较低。全美的基金管理规模是（TODO 查询），智能投顾占比

是?

1.2 国内情况

在目前的法律法规框架下，国内该类业务已被叫停。此外，国内资产标的范围较小，费用较高，并且没有相关税收优惠政策，都制约了该类业务的发展。

1.2.1 弥财

国内的弥财（https://micaiapp.com）从设计到理念，都比较接近美国的 WealthFront 公司。该公司的资产标的均为海外资产，并且需要投资者自行换汇。

但经本人测试，9 月初该公司申请了账户，到目前还没有开户成功，并且官方提供的资产电话也无人接听。可能是受到监管的影响，无法开展业务。

1.2.2 蛋卷基金

国内的雪球网（TODO 查询）

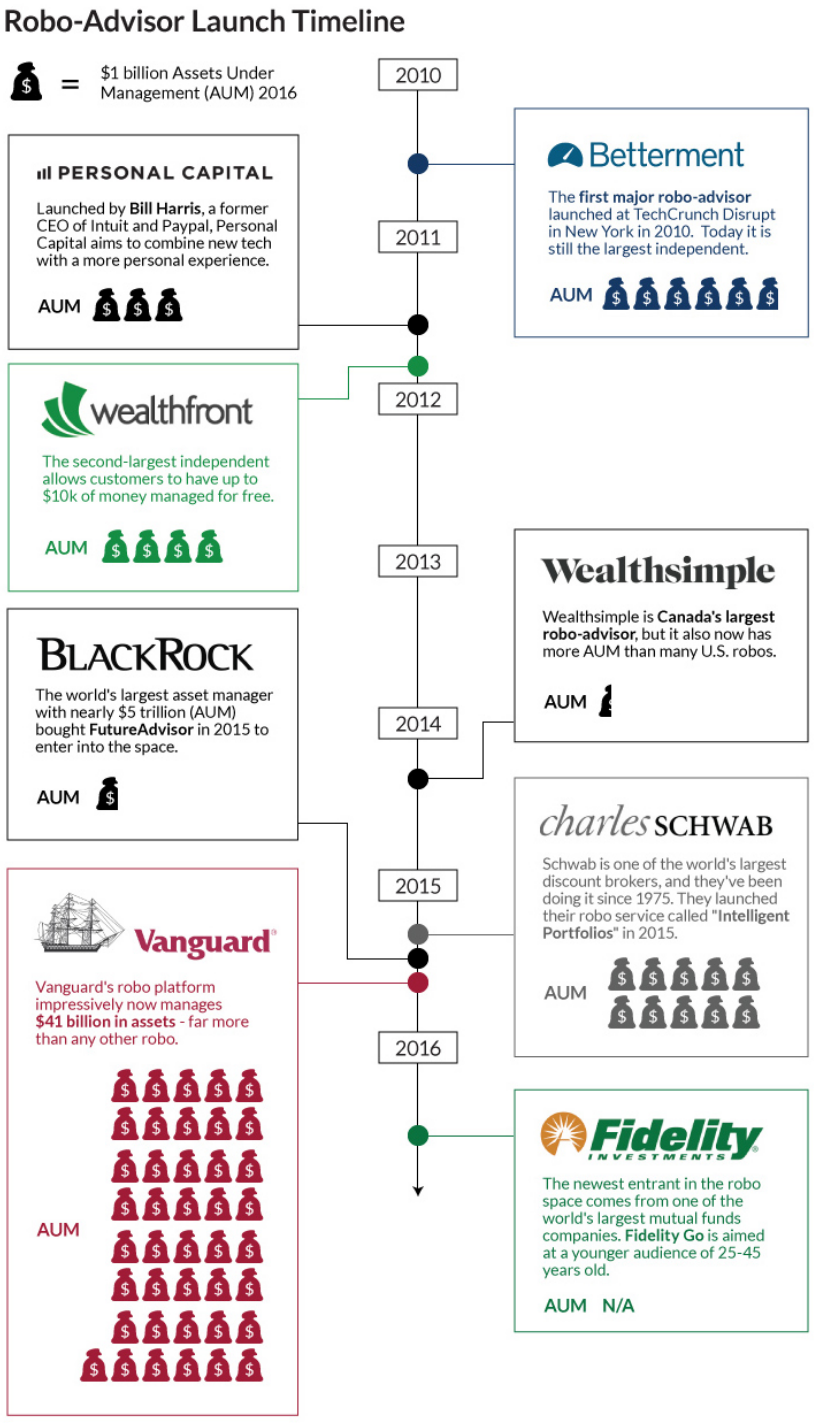

Figure 1.1: 智能投顾发展一览

(图片来源：www.visualcapitalist.com/robo-advisor-arms-race)

Chapter 2

实践出真知

本节构建资产组合。算法的理论基础是资本资产定价模型 CAPM（TODO 链接），同时参考 wealthfront [1] 的介绍。算法主要通过 Python（TODO 链接）实现，代码部分参考了通联数据（TODO 链接），海外资产的数据由 Yahoo Finance （TODO 链接）提供，国内资产的数据由 TuShare（TODO 链接）提供。

改进的方向：（1）选取资产（2）评估资产的相关性（3）预测资产收益率（4）模拟不同组合

本文的资产标的主要涵盖股票、黄金、房地产、原油及货币基金。

主要的资产组合都是 ETF。（TODO 解释为什么要用 etf）手续费较低，流通性好，可以日内交易。顺便提一下，华泰证券 [2] ETF 交易费最低 0.1 元！如果用来少量的搭建组合，还是比较划算。其他券商还是最低 5 元。

2.1 ETF 费用一览

（TODO 各 etf 的管理费、规模、成立日期，管理人）

国内的 ETF 收费太高，QDII 组合费用（管理费 + 托管费）在 1.00%

[1] https://research.wealthfront.com/whitepapers/investment-methodology/

[2] 此处不是广告！该券商未以任何形式提供赞助 -_-

左右，海外的约在 0.10% 左右。

2.2 算法分析

假设资产池中有 N 个备选资产。对于资产 i $(1 \leq i \leq N)$, 其年化预期收益率为 μ_i，波动率 σ_i。全部资产的协方差矩阵为 $V \in \mathbb{R}^{N \times N}$，其中，$V_{i,j}$ 表示资产 i 与资产 j 的相关性系数。特别地 $V^\top = V$，并且 $V_{k,k} = 1$ $(k = 1, \ldots, N)$ 。

给定一个资产组合的权重集:$\{w = (w_1, \ldots, w_N)^\top \in \mathbb{R}^{N \times 1} : \sum_{i=1}^{N} w_i = 1, w_i \geq 0\}$，我们下面计算该集合中对应“有效边界”(Efficient frontier) 对应的子集合。即：任意给定一个预期收益率 μ $(\min\{\mu_i\} \leq \mu \leq \max\{\mu_j\})$，我们计算 μ 对应的全部组合中，波动率最小的组合 $w(\mu)$ [3]。

该问题等价于如下的一个二次优化问题:

$$
\begin{aligned}
\min \quad & \frac{1}{2} w^\top V w \\
s.t. \quad & \sum_{i=1}^{N} w_i \cdot \mu_i = \mu \\
& \sum_{i=1}^{N} w_i = 1 \\
& w_i \geq 0 \,, \quad (1 \leq i \leq N)
\end{aligned}
$$

Python 中 `Cvxopt` 包可以解决该优化问题。

2.3 Python Code

本部分，我们实现上述算法。代码分为两部分：（1）根据我们选定的资产，下载其历史数据；（2）针对每一个资产，估算其年化预期收益率 μ_i、波动率 σ_i，以及全部资产间的协方差矩阵 V。最后计算全部资产的有效边界（`Efficient frontier`）。

[3]表示 $w(\cdot)$ 是一个关于 μ 的函数

2.3.1 选定资产

我们选取的国内资产拟定为货币基金 ETF，上证 50ETF，中证 500ETF，汇深 300ETF，恒生 ETF，债券 ETF，黄金 ETF；海外资产选取标普 500ETF（SPY），纳斯达克指数（QQQ），USO，XOP，黄金 ETF（GLD）。

详见代码：

Listing 2.1: Download the data

```python
# -*- coding: utf-8 -*-
'''
Modified on Fri, Sep 23, 2016
Author: Haifeng XU
Email:  78112407@qq.com

df.shape
df.describe()
df.info()
'''

import numpy   as np
import tushare as ts
import pandas  as pd
import pandas_datareader.data as web
import datetime

_start_date = '2007-01-01'
_end_date   = '2016-10-01'

## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
def _initial_index_cn() :
    index_list = []
    index_name = []

    index_list.append( '150151' )    ## HS300A
    index_name.append( 'HS300A' )

    index_list.append( '150152' )    ## HS300B
    index_name.append( 'HS300B' )
```

```
31
32      index_list.append( '159920' )   ## 恒生ETF
33      index_name.append( '恒生ETF' )
34
35      index_list.append( '160125' )   ## 南方香港
36      index_name.append( '南方香港' )
37
38      index_list.append( '160416' )   ## 石油黄金
39      index_name.append( '石油黄金' )
40
41      index_list.append( '160717' )   ## 恒生H股
42      index_name.append( '恒生H股' )
43
44      index_list.append( '161116' )   ## 易基黄金
45      index_name.append( '易基黄金' )
46
47      index_list.append( '161210' )   ## 国投新兴
48      index_name.append( '国投新兴' )
49
50      index_list.append( '161714' )   ## 招商金砖
51      index_name.append( '招商金砖' )
52
53      index_list.append( '161815' )   ## 银华通胀
54      index_name.append( '银华通胀' )
55
56      index_list.append( '162411' )   ## 华宝油气
57      index_name.append( '华宝油气' )
58
59      index_list.append( '164701' )   ## 添富贵金
60      index_name.append( '添富贵金' )
61
62      index_list.append( '164815' )   ## 工银资源
63      index_name.append( '工银资源' )
64
65      index_list.append( '165510' )   ## 信诚四国
66      index_name.append( '信诚四国' )
67
68      index_list.append( '165513' )   ## 信诚商品
```

```
    index_name.append( '信诚商品' )

    index_list.append( '510300' )    ## HS300 ETF
    index_name.append( '沪深300 ETF')

    index_list.append( '510500' )    ## 500 ETF
    index_name.append( '中证500 ETF')

    index_list.append( '510900' )    ## H股ETF
    index_name.append( 'H股ETF' )

    index_list.append( '511860' )    ## MoneyFund
    index_name.append( 'MoneyFund' )

    index_list.append( '513030' )    ## 德国30
    index_name.append( '德国30' )

    index_list.append( '513100' )    ## 纳指ETF
    index_name.append( '纳指ETF' )

    index_list.append( '513500' )    ## 标普500
    index_name.append( '标普500' )

    return index_list, index_name
## ........................................................ ##


## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
def _initial_index_us() :
    '''
    Initialization of US assets
    '''

    index_list = []
    index_name = []

    index_list.append( '' )
    index_name.append( 'Apple' )
```

```
107
108     index_list.append( 'GOOG' )
109     index_name.append( 'Google' )
110
111     index_list.append( 'GLD' )
112     index_name.append( 'Gold' )
113
114     index_list.append( 'SPY' )
115     index_name.append( 'S&P 500' )
116
117     index_list.append( 'USO' )
118     index_name.append( 'USO' )
119
120     index_list.append( 'XOP' )
121     index_name.append( 'XOP' )
122
123     return index_list, index_name
124 ## ..................................................... ##
125
126
127 ## ******************************************************* ##
128 def _initial_index( country_code ):
129     '''
130     ---------------------------------------------
131     This function just insert the asset code and name
132     ---------------------------------------------
133     '''
134
135     index_list = []
136     index_list_name = []
137
138     if country_code == 'cn' :
139         index_list, index_list_name = _initial_index_cn()
140
141     if country_code == 'us' :
142         index_list, index_list_name = _initial_index_us()
143
144     return index_list, index_list_name
```

```
145 ## ..................................................... ##
146
147
148 ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
149 ## 'my_ptf_cn' records the close price of the assets
150 ## in 'index_list_cn[]'
151 ## ********************************************************* ##
152 def _download_data( country_code ) :
153     '''
154     ----------------------------------------------------
155         Parameters
156     ----------------------------------------------------
157     index: refers to the code of assets.
158     index_name: refers to the name of assets.
159
160     ----------------------------------------------------
161         Return
162     ----------------------------------------------------
163     DataFrame: which contains the close price of the assets
164         in "index"
165     '''
166
167     ## create an empty pd, which would be returned
168     df = pd.DataFrame()
169
170     ## assets to be added
171     index, index_name = _initial_index( country_code )
172
173     '''
174     The data from CN is supplied by TuShare.
175     Please refer to
176     http://tushare.org/index.html for more details.
177     注意: CN 与 US 的价格日期是相反的, 我把 CN 的顺序调整了,
178         与 US 保持一致。即 tail() 是最新的数据。
179     '''
180     ## append the close price
181     if country_code == 'cn' :
182         for i in xrange( len( index ) ) :
```

```
183             df[ index[i] ] = ts.get_hist_data( index[ i ] ,
184                                                start = _start_date
    ,
185                                                end   = _end_date
    )[ 'close' ]
186
187     '''
188     The data from US is supplied by Yahoo-finance. Please refer to
189     http://pandas-datareader.readthedocs.io/en/latest/remote_data.
    html#yahoo-finance
190     for more details.
191
192     Notice that 'Adj Close' price is what we want.
193     '''
194     ## append the close price
195     if country_code == 'us' :
196         for i in xrange( len( index ) ) :
197             df[ index[i] ] = web.DataReader( index[ i ] ,
198                                              'yahoo',
199                                              start = _start_date ,
200                                              end   = _end_date )[
    'Adj Close' ]
201
202     ## replace inf and NA with zero
203     df[ df == np.inf ] = 0
204     df.fillna( 0, inplace = True )
205
206     ## rename the columns
207     df.rename( columns = dict( zip( index, index_name ) ),
208                inplace = True )
209
210     ## rename the index name
211     df.index.name = 'date'
212
213     ## make sure the tail() is the latest data
214     df = df.sort_index( ascending = True )
215
216     ## make sure the index is of type datetime
```

```
    df.index = pd.to_datetime( df.index )

    ## test
    print df.shape

    return df
## ..................................................... ##


## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
## calculate the correlation of the assets
## first of all, we calculate the changed ratio of price
## the input needs to be a DataFrame containing the close price
## ******************************************************* ##
def _price_change( tmp ) :
    '''
    ------------------------------------------------
        Parameters
    ------------------------------------------------
    tmp: a dataframe containing the price.
        Notice that the tail is the latest date

    ------------------------------------------------
        Return
    ------------------------------------------------
    DataFrame: a dataframe containing the change of price
    '''

    df = tmp.copy()
#   df[1:] = 1.0 * df[1:].values / df[:-1].values - 1
    df = df / df.shift(1) * 1.0 - 1.0

    ## the first row should be zero
    df[:1] = 0

    ## replace inf and NA with zero
    df[ df == np.inf ] = 0
    df.fillna( 0, inplace = True )
```

```
255
256     return df
257 ## ..................................................... ##
258
259
260 ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
261 ## We construst my portfolio according to 'index_list_cn'.
262 ## Each entry records the close price of the assets.
263 ## ********************************************************* ##
264 def _get_my_ptf():
265     '''
266     return my portfolio
267     '''
268
269     df    = pd.DataFrame()
270     df_cn = pd.DataFrame()
271     df_us = pd.DataFrame()
272
273     df_cn = _download_data( country_code = 'cn' )
274     df_us = _download_data( country_code = 'us' )
275
276     '''
277     Caution: join = 'outer' or 'inner'
278     http://pandas.pydata.org/pandas-docs/stable/merging.html
279     '''
280     df = pd.concat( [df_cn, df_us],
281                     axis = 1,
282                     join = 'inner' )
283
284     ## replace inf and NA with zero
285     df[ df == np.inf ] = 0
286     df.fillna( 0, inplace = True )
287
288     return df
289 ## ..................................................... ##
290
291 my_ptf = _get_my_ptf()
292 my_ptf.to_csv( 'Data/etf_close_price.csv' )
```

```python

my_ptf_rtn = _price_change( my_ptf )
my_ptf_rtn.to_csv( 'Data/etf_rtn.csv' )

## well, this is a bonus for users...
print('well, good job!')

'''
import matplotlib.pyplot as plt
fig = plt.figure()
ax  = fig.add_subplot(1,1,1)
ax.plot( randn( 1000 ).cumsum() )
ax.set_xticks( [0, 50, 100] )
ax.set_xticklabels( )
ax.set_title( )
ax.set_xlabel( )
ax.legend(loc='best')
'''

## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
## END of the code
## ******************************************************** ##
## ..................................................... ##
```

2.3.2 有效边界

估算资产的年化预期收益率以、波动率以及相关性系数。

Listing 2.2: Draw the captial allocation line

```
# -*- coding: utf-8 -*-

import numpy  as np
import pandas as pd
import datetime
from matplotlib import pyplot as plt
from cvxopt     import matrix, solvers

df_return = pd.DataFrame()
df_return = pd.read_csv( 'Data/etf_rtn.csv' )
df_return = df_return.sort_values( by = 'date', ascending = True )
df_return.fillna( 0, inplace = True )
print( df_return.head() )
## 注意:
## 这里每一列的有效数据长度不一致
## 即: 有的资产可能从2013年开始, 有的是从1998年开始

## print ( df_return.describe() )
print ( df_return.head() )


## 国内的 ETF QDII
portfolio1 = [2,3,4]
## 海外的 ETF
portfolio2 = [23,24,25,26]
## TODO: 如何快速的表达 2:23 ?

## 协方差矩阵
cov_mat = df_return.cov()
## 标的预期收益
exp_rtn = df_return.mean()


## 这个计算的代码需要优化一下
```

```
35 def cal_efficient_frontier( portfolio ) :
36     '''
37     We will compute the risk and its corresponding return.
38     '''
39
40     cov_mat1 = cov_mat.iloc[ portfolio ][ portfolio ]
41     exp_rtn1 = exp_rtn.iloc[ portfolio ]
42     max_rtn  = max( exp_rtn1 )
43     min_rtn  = min( exp_rtn1 )
44     risks    = []
45     returns  = []
46
47     '''
48     20个点作图
49     http://cvxopt.org/examples/tutorial/qp.html
50
51     min 1/2 * x^T * Q * x + p * x
52     s.t. G * x <= h
53          A * x = b
54     '''
55     for level_rtn in np.linspace( min_rtn, max_rtn, 20) :
56         sec_num = len( portfolio )
57
58         _Q = 2 * matrix( cov_mat1.values )
59         _p = matrix ( np.zeros( sec_num ) )
60         _G = matrix ( np.diag( -1 * np.ones( sec_num ) ) )
61         _h = matrix ( 0.0 , ( sec_num, 1 ) )
62         _A = matrix ( np.matrix( [ np.ones( sec_num ), exp_rtn1.
    values ] ) )
63         _b = matrix ( [ 1.0 , level_rtn ] )
64
65         solvers.options['show_progress'] = False
66         sol = solvers.qp( _Q, _p, _G, _h, _A, _b )
67
68         '''
69         ## 查看最优权重
70         if level_rtn == max_rtn :
71             print ('sol is here: \n')
```

```
            print ( sol['x'] )
        '''

        risks.append( sol[ 'primal objective' ] )
        returns.append( level_rtn )

    return np.sqrt( risks ), returns


risk1, return1 = cal_efficient_frontier( portfolio1 )
risk2, return2 = cal_efficient_frontier( portfolio2 )

fig = plt.figure ( figsize = (14, 8 ))
ax1 = fig.add_subplot( 111 )
ax1.plot ( risk1, return1 )
ax1.plot ( risk2, return2 )
ax1.set_title  ( 'Efficient Frontier', fontsize = 14 )
ax1.set_xlabel ( 'Standard Deviation', fontsize = 12 )
ax1.set_ylabel ( 'Expected Return'   , fontsize = 12 )
ax1.tick_params ( labelsize = 12 )
ax1.legend( [ 'portfolio1' , 'portfolio2' ] , loc = 'best',
    fontsize = 14 )
fig.savefig( 'Figure/Efficient_Frontier.png' )









print ('well, good job!')


## -------------------- END ----------------------- ##
```