智能投顾：理论与实践

Email: 78112407@qq.com

October 28, 2016

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。目前最大的智能投顾公司是著名的 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。2010 年至 2016 年的六年时间内，目前全部的智能投顾公司管理的资产规模约 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?

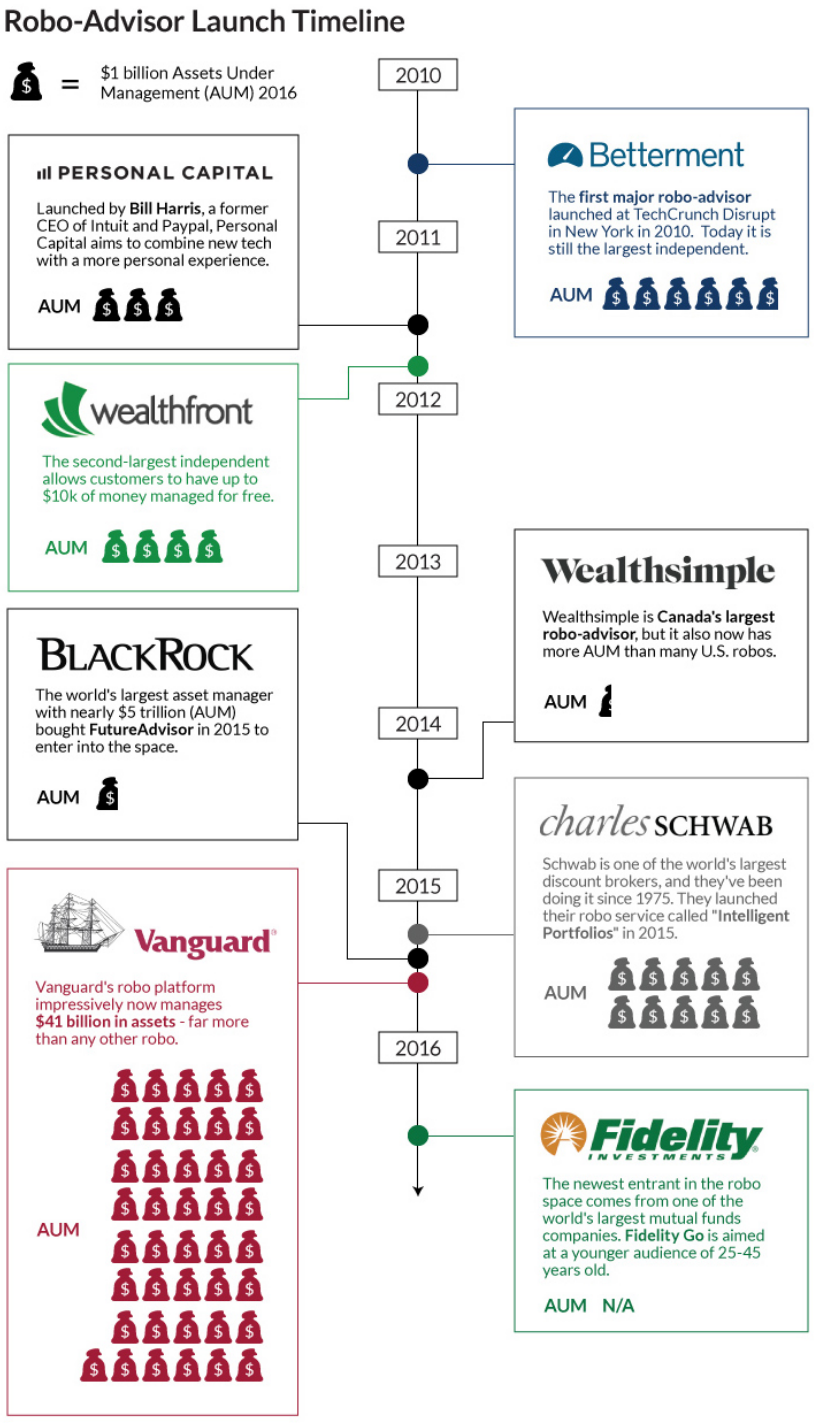

Figure 1.1: 智能投顾发展一览

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

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。

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。

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 （TODO 链接）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[1] 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 普遍比较坑，QDII 组合费用（管理费 + 托管费）在 1.00% 左右，海外的约在 0.10% 左右。

[1]此处不是广告！该券商未以任何形式提供赞助 -_-

2.2 算法分析

假设资产池中有 N 个备选资产。对于资产 i $(1 \leq i \leq N)$，其预期收益率为 μ_i，波动率 σ_i。资产的协方差矩阵为 $V \in \mathbb{R}^{N \times N}$，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：给定一个组合的预期收益率 μ $(\min\{\mu_i\} \leq \mu \leq \max\{\mu_j\})$，我们计算该组合对应的最小波动率的组合 $w(\mu)$ [2]。

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

Python 中 `Cvxopt` 包可以解决该优化问题。

2.3 Python Code

代码：

Listing 2.1: Download the data

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

[2] 表示 $w(\cdot)$ 是一个关于 μ 的函数

```
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' )

    index_list.append( '159920' )   ## 恒生ETF
    index_name.append( '恒生ETF' )

    index_list.append( '160125' )   ## 南方香港
    index_name.append( '南方香港' )

    index_list.append( '160416' )   ## 石油黄金
    index_name.append( '石油黄金' )

    index_list.append( '160717' )   ## 恒生H股
    index_name.append( '恒生H股' )

```

```
    index_list.append( '161116' )   ## 易基黄金
    index_name.append( '易基黄金' )

    index_list.append( '161210' )   ## 国投新兴
    index_name.append( '国投新兴' )

    index_list.append( '161714' )   ## 招商金砖
    index_name.append( '招商金砖' )

    index_list.append( '161815' )   ## 银华通胀
    index_name.append( '银华通胀' )

    index_list.append( '162411' )   ## 华宝油气
    index_name.append( '华宝油气' )
    ## 标普美国行业指数系列之油气开采及生产行业指数
    ## (S&P Select Industry Oil & Gas Exploration & Production)
    ## SPSIOP is the index being traced
    ## XOP is an ETF tracing SPSIOP
    ## 指数成分股的入选必须满足以下条件:
    ## 1、成份股是标普美国全市场指数的成员;
    ## 2、成份股属于GICS定义的油气二级行业分类;
    ## 3、成份股市值大于5亿美金，或市值大于4亿美金;
    ## 4、且交易量年换手率大于150%。

    index_list.append( '164701' )   ## 添富贵金
    index_name.append( '添富贵金' )

    index_list.append( '164815' )   ## 工银资源
    index_name.append( '工银资源' )

    index_list.append( '165510' )   ## 信诚四国
    index_name.append( '信诚四国' )

    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')
```

```
82
83      index_list.append( '510500' )   ## 500 ETF
84      index_name.append( '中证500 ETF')
85
86      index_list.append( '510900' )   ## H股ETF
87      index_name.append( 'H股ETF' )
88
89      index_list.append( '511860' )   ## MoneyFund
90      index_name.append( 'MoneyFund' )
91
92      index_list.append( '513030' )   ## 德国30
93      index_name.append( '德国30' )
94
95      index_list.append( '513100' )   ## 纳指ETF
96      index_name.append( '纳指ETF' )
97
98      index_list.append( '513500' )   ## 标普500
99      index_name.append( '标普500' )
100
101     return index_list, index_name
102 ## ..................................................... ##
103
104
105 ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
106 def _initial_index_us() :
107     '''
108     Initialization of US assets
109     '''
110
111     index_list = []
112     index_name = []
113
114     index_list.append( 'AAPL' )
115     index_name.append( 'Apple' )
116
117     index_list.append( 'GOOG' )
118     index_name.append( 'Google' )
119
```

```
120     index_list.append( 'GLD' )
121     index_name.append( 'Gold' )
122 
123     index_list.append( 'SPY' )
124     index_name.append( 'S&P 500' )
125 
126     index_list.append( 'USO' )
127     index_name.append( 'USO' )
128 
129     index_list.append( 'XOP' )
130     index_name.append( 'XOP' )
131 
132     return index_list, index_name
133 ## ................................................... ##
134 
135 
136 ## ****************************************************** ##
137 def _initial_index( country_code ):
138     '''
139     ---------------------------------------------
140     This function just insert the asset code and name
141     ---------------------------------------------
142     '''
143 
144     index_list = []
145     index_list_name = []
146 
147     if country_code == 'cn' :
148         index_list, index_list_name = _initial_index_cn()
149 
150     if country_code == 'us' :
151         index_list, index_list_name = _initial_index_us()
152 
153     return index_list, index_list_name
154 ## ................................................... ##
155 
156 
157 ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
```

```
158 ## 'my_ptf_cn' records the close price of the assets
159 ## in 'index_list_cn[]'
160 ## ********************************************************* ##
161 def _download_data( country_code ) :
162     '''
163     ------------------------------------------------------
164         Parameters
165     ------------------------------------------------------
166     index: refers to the code of assets.
167     index_name: refers to the name of assets.
168 
169     ------------------------------------------------------
170         Return
171     ------------------------------------------------------
172     DataFrame: which contains the close price of the assets
173         in "index"
174     '''
175 
176     ## create an empty pd, which would be returned
177     df = pd.DataFrame()
178 
179     ## assets to be added
180     index, index_name = _initial_index( country_code )
181 
182     '''
183     The data from CN is supplied by TuShare.
184     Please refer to
185     http://tushare.org/index.html for more details.
186     注意: CN 与 US 的价格日期是相反的, 我把 CN 的顺序调整了,
187         与 US 保持一致。即 tail() 是最新的数据。
188     '''
189     ## append the close price
190     if country_code == 'cn' :
191         for i in xrange( len( index ) ) :
192             df[ index[i] ] = ts.get_hist_data( index[ i ] ,
193                                                start = _start_date
    ,
194                                                end   = _end_date
```

```
    )[ 'close' ]

    '''
    The data from US is supplied by Yahoo-finance. Please refer to
    http://pandas-datareader.readthedocs.io/en/latest/remote_data.
    html#yahoo-finance
    for more details.

    Notice that 'Adj Close' price is what we want.
    '''
    ## append the close price
    if country_code == 'us' :
        for i in xrange( len( index ) ) :
            df[ index[i] ] = web.DataReader( index[ i ] ,
                                             'yahoo',
                                             start = _start_date ,
                                             end   = _end_date )[
    'Adj Close' ]

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

    ## rename the columns
    df.rename( columns = dict( zip( index, index_name ) ),
               inplace = True )

    ## rename the index name
    df.index.name = 'date'

    ## make sure the tail() is the latest data
    df = df.sort_index( ascending = True )

    ## make sure the index is of type datetime
    df.index = pd.to_datetime( df.index )

    ## test
    print df.shape
```

```
230
231     return df
232 ## ..................................................... ##
233
234
235 ## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
236 ## calculate the correlation of the assets
237 ## first of all, we calculate the changed ratio of price
238 ## the input needs to be a DataFrame containing the close price
239 ## ********************************************************* ##
240 def _price_change( tmp ) :
241     '''
242     ----------------------------------------------------
243         Parameters
244     ----------------------------------------------------
245     tmp: a dataframe containing the price.
246         Notice that the tail is the latest date
247
248     ----------------------------------------------------
249         Return
250     ----------------------------------------------------
251     DataFrame: a dataframe containing the change of price
252     '''
253
254     df = tmp.copy()
255 #   df[1:] = 1.0 * df[1:].values / df[:-1].values - 1
256     df = df / df.shift(1) * 1.0 - 1.0
257
258     ## the first row should be zero
259     df[:1] = 0
260
261     ## replace inf and NA with zero
262     df[ df == np.inf ] = 0
263     df.fillna( 0, inplace = True )
264
265     return df
266 ## ..................................................... ##
267
```

```python

## ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ##
## We construst my portfolio according to 'index_list_cn'.
## Each entry records the close price of the assets.
## ******************************************************** ##
def _get_my_ptf():
    '''
    return my portfolio
    '''

    df    = pd.DataFrame()
    df_cn = pd.DataFrame()
    df_us = pd.DataFrame()

    df_cn = _download_data( country_code = 'cn' )
    df_us = _download_data( country_code = 'us' )

    '''
    Caution: join = 'outer' or 'inner'
    http://pandas.pydata.org/pandas-docs/stable/merging.html
    '''
    df = pd.concat( [df_cn, df_us],
                    axis = 1,
                    join = 'inner' )

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

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

my_ptf = _get_my_ptf()
my_ptf.to_csv( 'Data/etf_close_price.csv' )

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
## ******************************************************** ##
## ..................................................... ##
```

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
def cal_efficient_frontier( portfolio ) :
    '''
    We will compute the risk and its corresponding return.
```

```
    '''

    cov_mat1 = cov_mat.iloc[ portfolio ][ portfolio ]
    exp_rtn1 = exp_rtn.iloc[ portfolio ]
    max_rtn  = max( exp_rtn1 )
    min_rtn  = min( exp_rtn1 )
    risks    = []
    returns  = []

    '''
    20个点作图
    http://cvxopt.org/examples/tutorial/qp.html

    min 1/2 * x^T * Q * x + p * x
    s.t. G * x <= h
         A * x = b
    '''
    for level_rtn in np.linspace( min_rtn, max_rtn, 20) :
        sec_num = len( portfolio )

        _Q = 2 * matrix( cov_mat1.values )
        _p = matrix ( np.zeros( sec_num ) )
        _G = matrix ( np.diag( -1 * np.ones( sec_num ) ) )
        _h = matrix ( 0.0 , ( sec_num, 1 ) )
        _A = matrix ( np.matrix( [ np.ones( sec_num ), exp_rtn1.
    values ] ) )
        _b = matrix ( [ 1.0 , level_rtn ] )

        solvers.options['show_progress'] = False
        sol = solvers.qp( _Q, _p, _G, _h, _A, _b )

        '''
        ## 查看最优权重
        if level_rtn == max_rtn :
            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 ---------------------- ##
```