**Index Tracking**

ETF select a small number or subset of constituents of stock or bond index to mimic BM index. Since ETF does not contain all constituents of BM index (full replication), tracking error (TE) take places. Furthermore, optimal subset is not fixed but variable according to the market development so that frequently rebalancing is required.   
  
The number of constituents of BM index is so large that the full replication is impossible due to the transaction costs and liquidity problem. Therefore, Index tracking is finding the optimal combination of subset securities for minimizing tracking errors and its objective function is formulated as follows.   
  
\[\begin{align} TE = \frac{1}{T} \sum\_{t=1}^{T} \left( \sum\_{i=1}^{N} \left( w\_i r\_{it} – R\_t^I \right)^2 \right) \end{align}\] Here, \(R\_t^I\) adn \(r\_{it}\) are time \(t \) returns of BM index and its constituents respectively and \(w\_i\) is the weight of \(i\) constituent.   
  
Using vector-matrix notation, the above problem is reformulated with its constraints as follows. \[\begin{align} &\min\_{w} \frac{1}{T} || Rw – R^I ||\_2^2 \\ \text{subject to}& \\ &e^T w = 1 \\ &\eta\_i Z\_i \leq w\_i < Z\_i \delta\_i \\ &\sum\_{t=1}^{N} Z\_i = K \\ &Z\_i = 0 \quad or \quad 1, \quad i=1,2,...,N \end{align}\] Here, \(N\) is the number of constituents of BM index and \(K\) is the number of constituents of ETF. \(R^I=(R\_1^I,R\_2^I,…,R\_T^I )^T\) is a \(T×1\) vector of BM index return and \(R=(R\_1,R\_2,…,R\_T)\) is a \(T×N\) matrix which is concatenated with all \(T×1\) vector of \(R\_i=(r\_i1,r\_i2,…,r\_iT )^T\) horizontally. \(w=(w\_1,w\_2,…,w\_N )^T\) is a \(T×1\) vector of allocation weights.   
  
Seeing the above constraints, first condition is so called budget constraint which means all capital is invested into ETF portfolio. Second condition denote the lower and upper bound for allocation weights. Third condition is a cardinality constraints that \(Z\_i\) may take on 0 or 1 and sum of it is \(K\). This constraints means only \(K\) securities from all \(N\) are invested.   
  
But this problem is considered a difficult problem because cardinality constraints make this NP hard problem, in other words, \(\sum\_{t=1}^{N} Z\_i = K\) make this problem highly dimensional discrete problem.. This means only when we calculate all combinations by using mixed integer programming, we can select the optimal combination. But the number of combination is too large to calculate it. For this reason, this problem is also called the sparse index tracking problem. Of course, recently Fengmin, Xu, and Xue (2015) suggest \(L\_{1/2}\) Regularization for this problem.   
  
For this post, we use sparseIndexTracking R package for sparse index tracking and also use ROI.plugin.ecos R package for index tracking and finally compare these two results.

**Second-order conic programming (SOCP)**

For index tracking, we use ROI and ROI.plugin.ecos. In particular, ROI.plugin.ecos provide the solver for the second-order conic programming (SOCP).   
  
**What is a SOCP and what is the relationship between SOCP and index tracking?**   
  
Second-order cone programming (SOCP) problems are convex optimization problems in which a linear function is minimized over the intersection of an affine linear manifold with the Cartesian product of second-order cones.   
  
Index tracking problem is typically rewritten into SOCP format and ROI.plugin.ecos or other index tracking solver need SOCP format as input format. Therefore we need to transform our index tracking errors minimization problem into second-order conic programming problem.   
  
We present the original and transform problem. You can easily find the concept of SOCP in the context of index tracking problem.   
  
For example, we try to mimic the benchmark index by minimizing tracking error. TE problem is as follows.   
  
\[\begin{align} &\min\_{w} \sqrt{\sum\_{t=1}^{T} \left( \sum\_{i=1}^{N} \left( R\_t^I – w\_i r\_{it} \right)^2 \right)} \\ \text{subject to}& \\ &e^T w = 1 \\ &w > 0 \\ \end{align}\]   
  
Here, \(w = (w\_1 , w\_2 , …, w\_N) \) and \(r = (r\_1, r\_2, …, r\_N) \).   
  
\[\begin{align} &\min\_{w} t \\ \text{subject to}& \\ &\sqrt{\sum\_{t=1}^{T} \left( \sum\_{i=1}^{N} \left( R\_t^I – w\_i r\_{it} \right)^2 \right)} \ge t \\ &e^T w = 1+t \\ &w > 0 \\ \end{align}\]   
  
Here, \(w = (w\_1 , w\_2 , …, w\_N, t) \) and \(r = (r\_1, r\_2, …, r\_N, 1) \).   
  
It is worth noting that definitions of \(w\) and \(r\) are different between two equations. The second equation also include \(t\) as a control variable. Second equation treats the first equation’s objective function as an additional constraint. For convenience, two equations omit \(\frac{1}{T}\) since it is a constant and use a square root for formal expression.   
  
Although the definition of SOCP seems somewhat difficult, we can easily observe the characteristics of SOCP from the above two formulation. The bottom line is that convex objective function can be transformed into a constraint and an objective function is replaced by a linear function.

**R package**

Using **ROI** and **ROI.plugin.ecos**, we can perform the index tracking minimization. But this case, since there is no cardinality constraints, we need to select the subset of securities. But **sparseIndexTracking** R package implements this cardinality constraints by adjusting the regularization parameter (\(\lambda\)). The higher the \(\lambda\), the more the coefficients are shrinked towards zero.

**R code**

The following R code implements two index tracking problems. We use data which is embedded in sparseIndexTracking R package. For expositional purpose, we assume the universe of stock consisted of 30 because it is difficult to demonstrate the results as a table or figure when using all 386 stocks. But after understanding the main contents, we also deal the 386 case.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95 | #==============================================================  # Financial Econometrics & Derivatives, ML/DL  # using R, Python, Keras, Tensorflow  # by Sang-Heon Lee  #  # <https://kiandlee.blogspot.com>  #————————————————————–  # Index Tracking Error Minimization  # using ROI.ecos and sparseIndexTracking  #==============================================================    graphics.off()  # clear all graphs  rm(list = ls()) # remove all files from your workplace    library(sparseIndexTracking)  library(ROI)  library(ROI.plugin.ecos)    #————————————————  # Data  #————————————————        # load stock index data      data(INDEX\_2010)      y = as.vector(INDEX\_2010$SP500)      X = as.matrix(INDEX\_2010$X)        # comment it when full data is used      X <– X[,1:30]        nobs = length(y); nX = ncol(X)    #————————————————  # 1) Using ROI and ROI.ecos  #————————————————        # w  = c( w1,  w2,  w3, t)’      # Xn = c(Xn1, Xn2, Xn3, 1)      #      # min sqrt( (y1 – X1’\*w)^2 + (y2 – X2’\*w)^2      #         + (y3 – X3’\*w)^2 + (y4 – X4’\*w)^2      #         + (y5 – X5’\*w)^2      # )      # s.t.      #      w1 + w2 + w3 = 1      #      w1, w2, w3 > 0        # –> Rewritten into the standard form      #      # minimize t      # s.t.      #      sqrt( (y1 – X1’\*w)^2 + (y2 – X2’\*w)^2      #          + (y3 – X3’\*w)^2 + (y4 – X4’\*w)^2      #          + (y5 – X5’\*w)^2      #      ) <= t      #      w1 + w2 + w3 = 1      #      w1, w2, w3 > 0        #————————————————      # Index tracking error minimization      # using second order cone programming      #————————————————        A <– rbind(c( rep(0,nX), –1), cbind(X,0))        soc <– OP(objective   = L\_objective(c(rep(0,nX), 1)),                constraints = c(                    C\_constraint(A, K\_soc(nobs+1), c(0,y)),                    L\_constraint(c(rep(1,nX), 0), “==”, 1))      )        soc\_sol <– ROI\_solve(soc, solver = “ecos”)      wgt\_roi <– soc\_sol$solution[1:nX]    #————————————————  # 2) Using sparseIndexTracking  #————————————————        # fit portfolio under error measure ETE      # (Empirical Tracking Error)        # Unconstrained      wgt\_sps <– spIndexTrack(X, y, lambda = 1e–180, u = 1,                              measure = ‘ete’, thres = 1e–180)        # Constrained      # wgt\_sps <- spIndexTrack(X, y, lambda = 1e-7,      #                         u = 1, measure = ‘ete’)    #————————————————  # 3) Comparison for allocation weights  #————————————————        round(cbind(wgt\_roi, wgt\_sps),4)    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

With arguments for unconstrained parameters (\(\lambda=1e-180\) and subset of stocks \(n=30\), Running the above R code results in the following weight allocations of two R package: ROI with ROI.plugin.ecos and sparseIndexTracking.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | > #————————————————  > # 3) Comparison for allocation weights  > #————————————————  >  >     round(cbind(wgt\_roi, wgt\_sps),4)                     wgt\_roi wgt\_sps  1436513D UN Equity  0.0270  0.0270  1500785D UN Equity  0.0220  0.0220  1518855D US Equity  0.0319  0.0319  9876566D UN Equity  0.0607  0.0607  A UN Equity         0.0149  0.0149  AA UN Equity        0.0426  0.0426  AAPL UW Equity      0.0444  0.0444  ABC UN Equity       0.0151  0.0151  ABT UN Equity       0.1330  0.1330  ADBE UW Equity      0.0114  0.0114  ADM UN Equity       0.0127  0.0127  ADP UW Equity       0.1440  0.1440  ADSK UW Equity      0.0113  0.0113  AEE UN Equity       0.0453  0.0453  AEP UN Equity       0.0158  0.0159  AES UN Equity       0.0074  0.0074  AET UN Equity       0.0132  0.0132  AFL UN Equity       0.0413  0.0413  AGN UN Equity       0.0145  0.0146  AIG UN Equity       0.0002  0.0002  AIV UN Equity       0.0452  0.0452  AIZ UN Equity       0.0202  0.0202  AKAM UW Equity      0.0000  0.0000  ALL UN Equity       0.0348  0.0348  ALTR UW Equity      0.0172  0.0172  AMAT UW Equity      0.0336  0.0336  AMGN UW Equity      0.0411  0.0411  AMP UN Equity       0.0503  0.0503  AMT UN Equity       0.0437  0.0437  AMZN UW Equity      0.0051  0.0051    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

For the sparse index tracking, with arguments for unconstrained parameters (\(\lambda=1e-6\) and subset of stocks \(n=30\), Running the above R code results in the following weight allocations of two R package: ROI with ROI.plugin.ecos and sparseIndexTracking.