Replication for 'Bond Risk Premiums with Machine Learning'

YoungHae Kim

March 9, 2021

1. Data

- Monthly yield-data : Liu, Wu(2020), Reconstructuring the yield curve
  - Annualized continuously-compounded zero coupon yield in percentage points
  - Extract → date : 1971.09 ~ 2019.12, maturity : 1month ~ 120month
  - `https://sites.google.com/view/jingcynthiawu/yield-data`
- Monthly macro-data : McCracken, Ng(2015)
  - Use 'current.csv' data → extract 1971.09 ~ 2019.12
  - `https://research.stlouisfed.org/econ/mccracken/fred-databases/`
  - In above site, 135 variables are described in 'Appendix_table_update' file. But now 7 variables are omitted → 128 variables
  - Omitted variables : NAMPI (Group1), NAPMEI (Group2), NAPM, NAPMNOI, NAPMSDI, NAPMII (Group4), NAPMPRI (Group7)
- Construct forward-rate, excess-return from yield-data
  - Define the zero-coupon yield at t with a maturity of n as $y_t^{(n)}$ $(t = \frac{1}{12}, \frac{2}{12}, ...48\frac{3}{12})$ $(n = 1, 2, ...10)$
  - The price of the n-year discount bond at time t relates to the zero-coupon yield : $log(P_t^{(n)}) = -ny_t^{(n)}$
  - The forward rate with maturity n at time t is dened as the return for a loan starting at $t+n-1$ and maturing at $t+n$: $f_t^{(n)} = log(P_t^{(n-1)}) - log(P_t^{(n)})$
  - The excess return : $rx_{t+1}^{(n)} = log(P_{t+1}^{(n-1)}) - log(P_t^{(n)}) - y_t^{(1)}$

2. Estimation method

- Using expanding windows.
  - Using 'Xexog' (fwd rate), 'X' (macro) : 1971.8~1988.12 / 'Y' (xr-rate) : 1972.8 ~1989.12 → Estimate parameters, By using 'Xexog' , 'X' (1989.01) → predict 'Y' (1990.01)
  - Using 'Xexog' (fwd rate), 'X' (macro) : 1971.8~1989.01 / 'Y' (xr-rate) : 1972.8 ~1990.01 → Estimate parameters, By using 'Xexog' , 'X' (1989.02) → predict 'Y' (1990.02)
  - Continue in this fasion... ... Using 'Xexog' (fwd rate), 'X' (macro) : 1971.8~2018.11 / 'Y' (xr-rate) : 1972.8 ~2019.11 → Estimate parameters, By using 'Xexog' , 'X' (2018.12) → predict 'Y' (2019.12)
  - So OOS : 360 months
- All sample period can be adjusted in part 4 in 'main' file. (Line 120 ~ 138)
  - Should input the end of month for the start / end of sample period
- maturity (n) : the maturity left when buying the bond
  - (Line 151 in 'main' file) maturity = [1,2,3,4,6,9] #(n) = 2,3,4,5,7,10

3. ML method

- PCR (fwd only) / PCR (fwd + macro)
  - No hyper-parameter
  - num_pca = [3,5,10] : # of principal component, So the size of predicted outcome : 3(num_pca) * 360(OOS) * 6(maturity)
- PLS (fwd only) / PLS (fwd + macro)
  - No hyper-parameter
  - num_pls = [3,5,10] : # of pls component, So the size of predicted outcome : 3(num_pls) * 360(OOS) * 6(maturity)
- Ridge-regression (fwd only)
  - hyper-paramter tuning α=[.01, .05, .1, .5, 1, 2.5, 5, 10] (Gridsearch)
  - the size of predicted outcome : 360(OOS) * 6(maturity)
- Lasso (fwd + macro)
  - hyper-parameter tuning in α automatically (Not gridsearch)
  - the size of predicted outcome : 360(OOS) * 6(maturity)
- Elastic net (fwd + macro)
  - 2 hyper-parameter tuning
  - l_1 ratio = [.1, .3, .5, .7, .9] / α automatically
  - the size of predicted outcome : 360(OOS) * 6(maturity)
- Gradient Boosting Regression Tree (fwd only)
  - No hyper-parametre tuning
  - Loss ftn , # of boosting stage, # initial estimator, max_features, ... etc are all set up
  - the size of predicted outcome : 360(OOS) * 6(maturity)
- Random-forest (fwd-rate + macro)
  - No hyper-parametre tuning
  - n_estimators, max_depth, bootstrap, max_features , max_samples ... ... etc are all set up
  - the size of predicted outcome : 360(OOS) * 6(maturity)
- Random-forest (fwd-rate + macro), (hyper-parameter tuning)
  - hyper-paramter tuning : 1(n_estimators), 2(max_depth), 3(max_features)
  - the other hyper-parameters are all set up
  - the size of predicted outcome : 360(OOS) * 6(maturity)
- Neural-Net (fwd only) (Figure2, page 12)
  - No hyper-parameter tuning
  - archi is the # of neurons in hidden layers for fwd variables (list) ex) [5,5]
  - drop-out is prob for fast training ex)0.25
  - use mini-batch / early-stopping
  - the size of predicted outcome : 360(OOS) * 6(maturity)
- Neural-Net (fwd+macro) (Figure3-(a)) (macro + fwd direct in the last layer)
  - No hyper-parameter tuning
  - archi is the # of neurons in hidden layers for macro variables (list) ex) [32,16]
  - drop-out is for fast training ex)0.25
  - use mini-batch / early-stopping
  - After the hidden layers(archi), the (macro) outcome and fwd-rate are linearly combined to output layer. If archi is [32, 16], 128(macro variables)$\rightarrow$ 32$\rightarrow$ 16 + 10(fwd-rate) (=26) $\rightarrow$ 6(excess return)
  - the size of predicted outcome : 360(OOS) * 6(maturity)