Replicating the Results of Empirical Asset pricing via Machine Learning (2020)

Kaixi Zhang
Department of Economics
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
kzhangbh@connect.ust.hk

Presentation video: https://b23.tv/XoTj9wF

1 Introduction

Gu, Kelly and Xiu (2020) provide a comprehensive analysis of machine learning methods to learn the canonical problem in finance: empirical asset pricing. Most of their work focus on using a large amount of features to forecast future asset's excess returns. Their methods include most of typical machine learning models such as OLS, partial least square (PLS), principal components regression (PCR), random forest (RF), neural networks, etc. Due to time constraint, I only replicate parts of their results.

2 Data and overarching model

Gu, Kelly and Xiu (2020) obtain their data from CRSP for all firms listed in NYSE, AMEX and NASDAQ. Their sample of monthly total individual equity returns begins in March 1957 and ends in December 2016. They build a large collection of stock-level predictive characteristics including three parts: 94 characteristics, 74 industry dummies corresponding to the first two digits of Standard Industrial Classification (SIC) codes and 8 macroeconomic predictors. To be specific, I construct the eight macroeconomic predictors following Welch and Goyal (2008), they provided data [1] and detailed explanations.[2]

The basic prediction error model of an asset's excess return can be written as

$$r_{i,t+1} = \mathbb{E}(r_{i,t+1}) + \epsilon_{i,t+1} \tag{1}$$

where $\mathbb{E}(r_{i,t+1}) = g^*(z_{i,t})$. They define the baseline set of stock-level covariates $z_{i,t}$ as

$$z_{i,t} = x_t \otimes c_{i,t} \tag{2}$$

where x_t is a $P_c \times 1$ matrix of characteristics for each stock i, and x_t is a $P_x \times 1$ vector of macroeconomic predictors (and are thus common to all stocks, including a constant). Thus, $z_{i,t}$ is a

[1] Amit Goyal's website provides original data (up to 2005, used in their paper) and updated data (up tp 2021). I use the updated monthly data set here.

[2] **Dividend Price Ratio (dp)**: the difference between the log of dividends and the log of prices. **Earnings Price Ratio (ep_m)**: the difference between the log of earnings and the log of prices. **Book-to-Market Ratio (b.m)**: the ratio of book value to market value for the Dow Jones Industrial Average. **Net Equity Expansion (ntis)**: the ratio of 12-month moving sums of net issues by NYSE listed stocks divided by the total end-of-year market capitalization of NYSE stocks. **Treasury Bills (tbl)**. **Term Spread (tms)**: the difference between the long term yield on government bonds and the Treasury-bill. **Default Yield Spread (dfy)**: the difference between BAA and AAA-rated corporate bond yields. **Stock Variance (svar)**: Stock Variance is computed as sum of squared daily returns on the S&P 500.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

$P_c P_x \times 1$ vector and the total number of covariates is $94 \times (8+1) + 74 = 920$. Combine (1) and (2), the pricing model becomes

$$g^*(z_{i,t}) = \mathbb{E}(r_{i,t+1}) = z'_{i,t}\theta \tag{3}$$

note that $g^*(z_{i,t})$ is a general format, not restricted to be a linear function.

Besides, they divide the 60 years of data into 18 years of training sample (1957–1974), 12 years of validation sample (1975–1986), and the remaining 30 years (1987– 2016) for out-of-sample testing. Besides, they use a recursive performance evaluation scheme, that is, recursively increasing the training sample, periodically refitting the entire model once per year, and making out-of-sample predictions using the same fitted model over the subsequent year. Each time refitting, they increase the training sample by a year, while maintaining a fixed size rolling sample for validation. They choose to not cross-validate in order to maintain the temporal ordering of the data for prediction.

Furthermore, to assess predictive performance, they calculate the out-of-sample R^2 as

$$R^2_{oos} = 1 - \frac{\sum_{(i,t)\in T_3}(r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{(i,t)\in T_3} r^2_{i,t+1}} \tag{4}$$

```r
R2_oos <- function(pred){
  y.test <- response[!train]
  1- sum((pred - y.test)^2, na.rm = T)/sum(y.test^2, na.rm = T)
}
```

Listing 1: Define R^2_{oos}

where T_3 indicates that fits are only assessed on the testing subsample. They also use the Diebold and Mariano (1995) test for differences in out-of-sample predictive accuracy between model (1) and model (2), which defined as

$$DM_{12} = \frac{\bar{d}_{12}}{\hat{\sigma}_{\bar{d}_{12}}} \tag{5}$$

where

$$d_{12,t+1} = \frac{1}{n_{3,t+1}} \sum_{i=1}^{n_{3,t+1}} ((\hat{e}^{(1)}_{i,t+1})^2 - (\hat{e}^{(2)}_{i,t+1})^2) \tag{6}$$

$\hat{e}^{(1)}_{i,t+1}$ and $\hat{e}^{(2)}_{i,t+1}$ denote the prediction error for stock return i at time t using each method, and $n_{3,t+1}$ is the number of stocks in the testing sample (year $t+1$). Then $\bar{d}_{12}$ and $\hat{\sigma}_{\bar{d}_{12}}$ denote the mean and Newey-West standard error of $d_{12,t}$ over the testing sample.

3 Algorithms in details

3.1 Simple linear

The simple linear predictive regression model estimated through ordinary least squares (OLS) with all covariates and OLS-3 which only contains size **(mvel1)**, book-to-market **(bm)** and momentum **(mom12m, mom1m, mom36m, mom6m)** as covariates. The objective function is

$$\mathcal{L}(\theta) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} (r_{i,t+1} - g(z_{i,t};\theta))^2 \tag{7}$$

where $g(z_{i,t};\theta) = z'_{i,t}\theta$. They also use Huber robust objective function to improve the stability of forecasts (to deal with heavy tails of the distribution of financial returns)

$$\mathcal{L}_H(\theta) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T} H(r_{i,t+1} - g(z_{i,t};\theta), \xi) \tag{8}$$

where

$$H(x;\xi) = \begin{cases} x^2, & if \quad |x| \le \xi \\ 2\xi|x| - \xi^2, & if \quad |x| > \xi \end{cases} \tag{9}$$

tuning parameter ξ can be optimized adaptively from the data.

```
# using Huber loss instead of the l2 loss
library(MASS)
ols_Huber <- rlm(X.train, y.train)
ols3_Huber <- rlm(X3.train, y.train)
## Default S3 method:
##rlm(x, y, weights, ..., w = rep(1, nrow(x)),
    #init = "ls", psi = psi.huber,
    #scale.est = c("MAD", "Huber", "proposal 2"), k2 = 1.345,
    #method = c("M", "MM"), wt.method = c("inv.var", "case"),
    #maxit = 20, acc = 1e-4, test.vec = "resid", lqs.control = NULL)
pred.ols_Huber <- as.matrix(X.test) %*% as.matrix(ols_Huber$
    coefficients)
pred.ols3_Huber <- as.matrix(X3.test) %*% as.matrix(ols3_Huber$
    coefficients)
```

Listing 2: OLS with Huber loss

3.2 Penalized linear

Penalized methods adds a penalty to the original loss function

$$\mathcal{L}(\theta;.) = \mathcal{L}(\theta) + \phi(\theta;.) \tag{10}$$

where $\phi(\theta;.)$ "elastic net" penalty takes the form

$$\phi(\theta;\lambda,\rho) = \lambda(1-\rho)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\rho\sum_{j=1}^{P}\theta_j^2 \tag{11}$$

Some special cases as following

$$\phi(\theta;.) = \begin{cases} \frac{1}{2}\lambda\sum_{j=1}^{P}\theta_j^2, & Ridge \\ \lambda\sum_{j=1}^{P}|\theta_j|, & Lasso \\ \lambda(1-\rho)\sum_{j=1}^{P}|\theta_j| + \frac{1}{2}\lambda\rho\sum_{j=1}^{P}\theta_j^2, & ElasticNet \\ \lambda\sum_{j=1}^{P}||\theta_j||, & GroupLasso \end{cases} \tag{12}$$

```
library(hqreg)
X = as.matrix(X.train)
y = as.vector(y.train)
ENet.fit <- hqreg(X, y, method = "huber")
pred.ENet <- predict(ENet.fit, X.test)
X.test <- as.matrix(X.test)
pred.ENet <- predict(ENet.fit, X.test, type = "response")
```

Listing 3: Elastic net (ENet)

3.3 PCR and PLS

Both PCR and PLS are dimension reduction methods, and consist of a two-step procedure.

```
##Partial least square (PLS)
library(pls)
set.seed(1)
pls.fit <- plsr(y.train ~ ., data = pred.train, scale = T, validation
    = "CV")
summary(pls.fit)
pred.pls <- predict(pls.fit, X.test, ncomp = 10)
##Principal components regression (PCR)
set.seed(1)
pcr.fit <- pcr(y.train ~ ., data = pred.train, scale = T, validation =
     "CV")
summary(pcr.fit)
validationplot(pcr.fit, val.type = "MSEP")
pred.pcr <- predict(pcr.fit, X.test, ncomp = 84)
```

Listing 4: PLS and PCR

3.4 Generalized linear: moving beyond linearity

They still use the least squares objective function but with a K-term spline series [3] expansion of the predictors

$$g(z;\theta,p(\cdot)) = \sum_{j=1}^{P} p(z_j)'\theta_j \tag{13}$$

where $p(\cdot) = (p_1(\cdot), p_2(\cdot), ..., p_K(\cdot))$ is a vector of basis functions, and the parameters are now a $K \times N$ matrix $\theta = (\theta_1, \theta_2, ..., \theta_N)$. They adopt a spline series of order two: $(1, z, (z-c_1)^2, (z-c_2)^2, ..., (z-c_{K-2})^2)$, where $c_1, c_2, ...c_{K-2}$ are knots. They also use Huber robustness modification which is specialized for the spline expansion setting and is known as the group lasso

$$\phi(\theta;\lambda,K) = \lambda \sum_{j=1}^{P} (\sum_{k=1}^{K} \theta_{j,k}^2)^{1/2} \tag{14}$$

```
library(glmnet)
library(grpreg)
# lasso
cv.lasso.fit <- cv.glmnet(X.train, y.train, alpha = 1)
bestlam <- cv.lasso.fit$lambda.min
lasso.fit <- glmnet(X.train, y.train, alpha = 1)
pred.lasso <- predict(lasso.fit, s = bestlam, newx = X.test)
# group lasso
cv.glm.fit <- cv.grpreg(X, y, group = 1:ncol(X), penalty = "grLasso")
bestlam <- cv.glm.fit$lambda.min
glm.fit <- grpreg(X, y, group = 1:ncol(X), penalty = "grLasso")
pred.glm <- predict(glm.fit, X.test, s = bestlam)
```

Listing 5: Generalized linear model with group lasso

3.5 Boosted regression trees and random forests

```
## Random forest (RF)
library(randomForest)
set.seed(1)
rf.fit <- randomForest(y.train ~ ., data = pred.train, mtry = 30,
    importance = T)
```

[3] *R package: splines*

Table 1: Monthly out-of-sample stock-level prediction performance (percentage R^2_{oos})

	OLS	OLS-3	PLS	PCR	ENet	GLM	RF	GBRT
GKX(2020)'s results	-3.46 +H	0.16 +H	0.27	0.26	0.11 +H	0.19 +H	0.33	0.34 +H
Replicate results	-697.5 +H	-2.09 +H	1.69	1.66	-18.4 +H	-21.06	-6.08	-67.05

```
importance(rf.fit)
varImpPlot(rf.fit)
pred.rf <- predict(rf.fit, newdata = X.test)
## gradient boosted regression trees (GBRT)
library(gbm)
set.seed(0)
pred_train <- as.data.frame(pred_train)
boost.fit <- gbm(y.train ~ ., data = pred.train,
                 distribution = "gaussian", n.trees = 500,
                 interaction.depth = 4) # l2 norm
summary(boost.fit)
plot(boost.fit, i = "mom1m")
plot(boost.fit, i = "dy")
X.test <- as.data.frame(X.test)
pred.boost <- predict(boost.fit, newdata = X.test, n.trees = 500)
```

Listing 6: random forests and gradient boosted regression trees

4 Comparative results

I present the comparison of different machine learning models in terms of their out-of-sample predictive R^2, which includes OLS with all covariates, OLS-3, PLS, PCR, elastic net (ENet), generalized linear model with group lasso (GLM), random forest (RF), gradient boosted regression trees (GBRT).

```
> R2_oos(pred.ols3_Huber)
[1] -0.02088182
> R2_oos(pred.ols_Huber)
[1] -6.974626
> R2_oos(pred.pls)
[1] 0.0169284
> R2_oos(pred.pcr)
[1] 0.01657843
> R2_oos(pred.ENet)
[1] -0.1840032
> R2_oos(pred.lasso)
[1] -0.2106485
> R2_oos(pred.glm)
[1] -0.670718
> R2_oos(pred.rf)
[1] -0.06084936
> R2_oos(pred.boost)
[1] -0.6705164
```

Listing 7: Results of R^2_{oos}

5 Variable importance analysis

They calculate the reduction in R^2 from setting all values of a given predictor to zero within each training sample, and average these into a single importance measure for each predictor. but I just use the results of random forest to show that

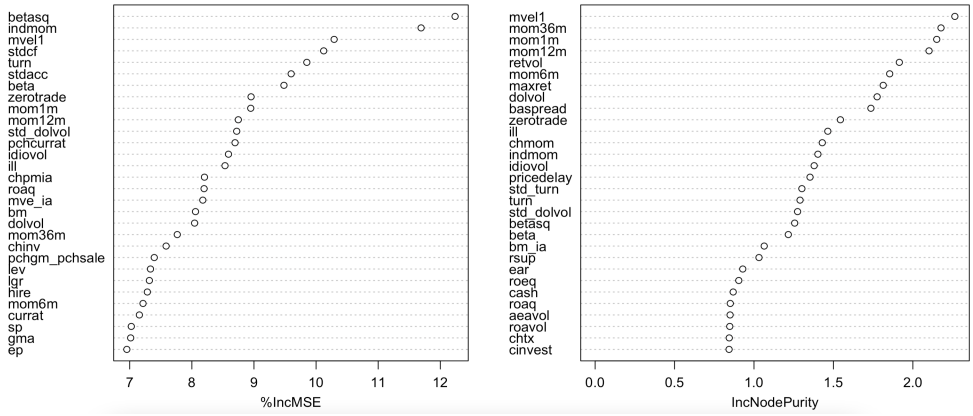

Figure 1: Variable importance by random forest

```
importance(rf.fit)
varImpPlot(rf.fit)
```

Listing 8: using random forests to show which covariates matter

References

[1] Gareth, J. & Trevor, H. (2021). An introduction to Statistical Learning. *Springer, Second Edition*.

[2] Gu, S. & Kelly, B. & and Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies* **33**(5): 5249-5262.

[3] Welch, I., and A. Goyal. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies* **21**: 1455–1508.