

Evaluation of Machine Learning in Finance

Ze Yu Zhong

Supervisor: David Frazier

Monash University

Wednesday 8th May, 2019

- 1 Problems in Empirical Finance
- 2 What is Machine Learning?
- 3 Why apply Machine Learning in Finance?
- 4 Model Specification
- 5 Simulation
 - Real World Observations
 - Simulation Design
- 6 Real Data
- 7 Model Evaluation
- 8 Results
- 9 References
- 10 Questions and Answers

Background of Problems in Empirical Finance

- Regressors can be:
 - ▶ Non-stationary - information now does not contain information about the future
 - ▶ Persistent - shocks in a series have effects that last for a long time
 - ▶ Cross sectionally correlated - regressors may seem important but are actually the result of a different underlying regressor
 - ▶ Endogeneous - omitted variable bias, etc

Background Problems in Empirical Finance

- Data is not robust - structural breaks are evident in returns data, and many regressors that once performed well do not anymore
- Extremely large number of potential factors (regressors) that is still increasing: over 600 documented in the literature Harvey and Liu (2019)

What is Machine Learning?

- Statistical/Machine Learning refers to a vast set of tools for understanding data
- Building statistical models for predicting outputs based on inputs
- Find patterns in datasets
- Examples of procedures: Principal Components Analysis, Principal Least Squares
- Examples of models: Ordinary Least Squares, LASSO Regression, Generalized Linear Models, Decisions Trees, Neural Networks

Why apply Machine Learning in Finance?

- Well suited for prediction
- Better equipped to deal with large dimensionality
- Capable of capturing non-linear transformations humans cannot realistically find

Why apply Machine Learning in Finance?

- Machine Learning Algorithms have been applied successfully to solve some of the problems present
- Kozak et al. (2017), Rapach and Zhou (2013), Freyberger et al. (2017), among others have applied shrinkage and selection methods to identify important factors
- Gu et al. (2018), Hsu and Kalesnik (2014), Feng et al. (2018), among others have constructed machine learning portfolios that historically outperform traditional portfolios
- Capable of capturing non-linear transformations humans cannot realistically find

Model Overview

- Returns are modelled as an additive error model



$$r_{i,t+1} = E(r_{i,t+1}|\mathcal{F}_t) + \epsilon_{i,t+1} \quad (1)$$

where

$$E(r_{i,t+1}|\mathcal{F}_t) = g^*(z_{i,t}) \quad (2)$$

with $g^*(z_{i,t})$ representing the model approximation using the predictor set $z_{i,t}$

Sample Splitting

- Two main approaches to dealing with temporal data
 - ▶ Rolling window - training, validation, and test set lengths are fixed and move forwards in time
 - ▶ Growing window - training set grows in size, but validation and test set lengths are fixed and move forwards in time
- Hybrid approach was chosen for feasibility
- Define a training set, validate on the next year, forecast for the next year
- Increase training set by one more year and move the validation and test sets forward one year

Sample Splitting

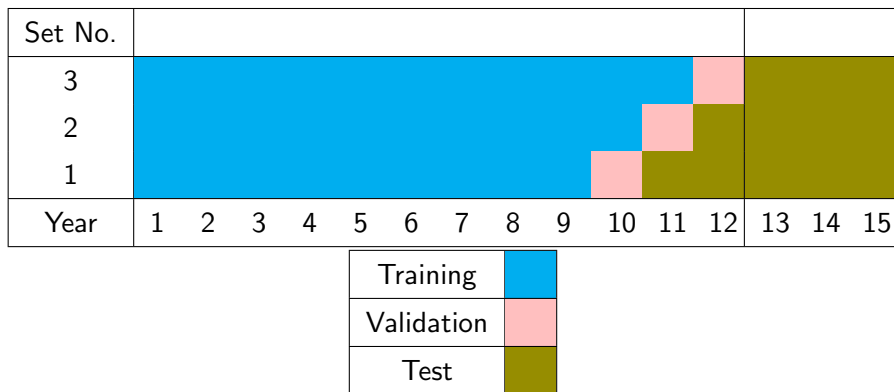


Figure 1: Sample Splitting Procedure

Loss Functions

- Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n} \sum_{j=i}^n |y_j - \hat{y}_j| \quad (3)$$

- Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n} \sum_{j=i}^n (y_j - \hat{y}_j)^2 \quad (4)$$

- Huber Loss

$$H(\epsilon_j = y_j - \hat{y}_j; \xi) = \begin{cases} (y_j - \hat{y}_j)^2, & \text{if } |y_j - \hat{y}_j| \leq \xi; \\ 2\xi|y_j - \hat{y}_j| - \xi^2, & \text{if } |y_j - \hat{y}_j| > \xi \end{cases} \quad (5)$$

Loss Functions

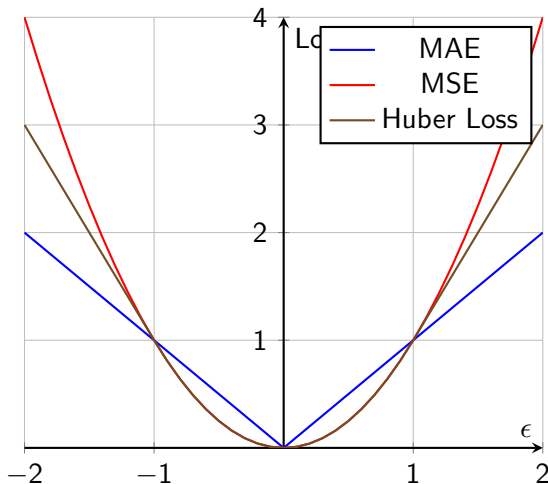


Figure 2: Illustration of MAE, MSE and Huber Loss when $\xi = 1$

Models Considered

- Linear Models
- Penalized Linear Models (Elastic Net)
- Random Forests
- Neural Networks

Linear Models

- Linear Models assume that the underlying conditional expectation $g^*(z_{i,t})$ can be modelled as a linear function of the predictors and the parameter vector θ :

$$g(z_{i,t}; \theta) = z'_{i,t} \theta \quad (6)$$

- Optimizing θ with respect to minimizing MSE yields the Pooled OLS estimator
- Limitations:
 - ▶ Need to manually consider and specify non-linear interactions
 - ▶ Struggles with high dimensionality

Penalized Linear Models

- Penalized linear models have the same underlying statistical model as simple linear models, add a new penalty term in the loss function:

$$\mathcal{L}(\theta; \cdot) = \underbrace{\mathcal{L}(\theta)}_{\text{Loss Function}} + \underbrace{\phi(\theta; \cdot)}_{\text{Penalty Term}} \quad (7)$$

- Focus on the popular "elastic net" penalty (Zou and Hastie, 2005), which takes the form for the penalty function $\phi(\theta; \cdot)$:

$$\phi(\theta; \lambda, \rho) = \lambda(1 - \rho) \sum_{j=1}^P |\theta_j| + \frac{1}{2} \lambda \rho \sum_{j=1}^P \theta_j^2 \quad (8)$$

Regression Trees and Random Forests

- Fully non-parametric models that can capture complex multi-way interactions.
- A tree "grows" in a series of iterations:
 - 1 Make a split ("branch") along one predictor, such that it is the best split available at that stage with respect to minimizing the loss function
 - 2 Repeat until each observation is its own node, or until the stopping criterion is met
- The eventual model slices the predictor space into rectangular partitions, and predicts the unknown function $g^*(z_{i,t})$ with the "average" value of the outcome variable in each partition, with respect to minimizing the loss function

Regression Trees and Random Forests

The prediction of a tree, \mathcal{T} , with K "leaves" (terminal nodes), and depth L is

$$g(z_{i,t}; \theta, K, L) = \sum_{k=1}^K \theta_k \mathbf{1}_{z_{i,t} \in C_k(L)} \quad (9)$$

where $C_k(L)$ is one of the K partitions in the model.

Only recursive binary trees are considered.

Regression Trees and Random Forests

Trees can be grown with respect to a variety of loss functions, including mean absolute error, mean squared error and Huber Loss:

$$H(\theta, C) = \frac{1}{|C|} \sum_{z_{i,t} \in C} L(r_{i,t+1} - \theta) \quad (10)$$

where $|C|$ denotes the number of observations in set C (partition).
Optimal prediction for each partition is the mean of the partition to minimize MSE, and the median of the partition to minimize MAE.

Random Forests

Trees have very low bias and high variance

They are very prone to overfitting and non-robust

Random Forests were proposed by Breiman (2001) to address this

- Create B bootstrap samples
- Grow a highly overfit tree to each, but only using m random subset of all predictors for each
- Average the output from all trees as an ensemble model

Neural Networks

Most complex type of model available

Able to capture several non-linear interactions through their many layers,
hence its other name “deep learning”

Highly flexible and therefore often the most parameterized and least
interpretable models

The scope of this paper is limited to traditional “feed-forward” networks.

The feed forward network consists of an “input layer” of scaled data inputs, one or more “hidden layers” which interact and non-linearly transform the inputs, and finally an output layer that aggregates the hidden layers and transform them a final time for the final output. A neural network with no hidden layers reduces to already familiar regression models, such as OLS and Probit (depending on activation function chosen).

Neural Network Specifications

Neural networks with up to 5 hidden layers were considered. The number of neurons in each layer was chosen according to the geometric pyramid rule (Masters, 1993)

All units are fully connected: each neuron receives input from all neurons in the layer before it

ReLU activation function was chosen for all hidden layers for computational speed, and hence popularity in literature:

$$\text{ReLU}(x) = \max(0, x) \quad (11)$$

Computation

- Stochastic Gradient Descent using ADAM
- Batch Normalization
- Randomize initial starting weights and biases, then average these into an ensemble model
- See references for specific details

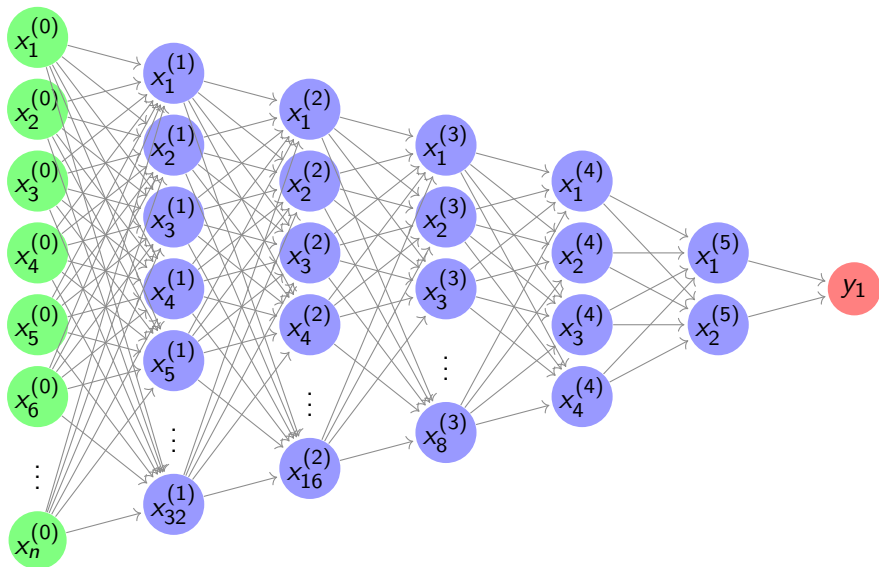


Figure 3: Neural Network 5 (most complex considered)

Real World Observations

- Though Gu et al. (2018) explore the performance of machine learning on simulated returns series, their design used factors which did not adequately capture the behaviour of stock characteristics observed in real life
- In particular, their specification lacks:
 - ▶ Cross Sectional correlation among factors
 - ▶ Stochastic Volatility in errors
 - ▶ More complex interactions
 - ▶ Multivariate time series

Overall Simulation Design

Therefore, we simulate an extension: a latent factor model with stochastic volatility for excess return, r_{t+1} , for $t = 1, \dots, T$:

$$r_{i,t+1} = g(z_{i,t}) + \beta_{i,t+1} v_{t+1} + e_{i,t+1}; \quad (12)$$

$$z_{i,t} = (1, x_t)' \otimes c_{i,t}; \quad \beta_{i,t} = (c_{i1,t}, c_{i2,t}, c_{i3,t}); \quad (13)$$

$$e_{i,t+1} = \exp\left(\frac{\sigma_{i,t+1}^2}{2}\right) \varepsilon_{i,t+1}; \quad (14)$$

$$\sigma_{i,t+1}^2 = \omega + \gamma_i \sigma_{t,i}^2 + w_{i,t+1} \quad (15)$$

v_{t+1} is a 3×1 vector of errors, $w_{i,t+1}, \varepsilon_{i,t+1}$ are scalar error terms. The parameters of these were tuned such that the R squared for each individual return series was 50% and annualized volatility 30%.

The matrix C_t is an $N \times P_c$ vector of latent factors, where the first three

Simulating Characteristics

A simulation mechanism for C_t that gives some correlation across the factors time was used. First consider drawing normal random numbers for each $1 \leq i \leq N$ and $1 \leq j \leq P_c$, according to

$$\bar{c}_{ij,t} = \rho_j \bar{c}_{ij,t-1} + \epsilon_{ij,t}; \quad \rho_j \sim \mathcal{U}\left(\frac{1}{2}, 1\right) \quad (16)$$

Then, define the matrix

$$B := \Lambda \Lambda' + \frac{1}{10} \mathbb{I}_n, \quad \Lambda_i = (\lambda_{i1}, \dots, \lambda_{i4}), \quad \lambda_{ik} \sim N(0, 1), \quad k = 1, \dots, 4 \quad (17)$$

Simulating Characteristics

Transform this into a correlation matrix W via

$$W = (\text{diag}(B))^{-\frac{1}{2}} (B) (\text{diag}(B))^{-\frac{1}{2}} \quad (18)$$

To build in cross-sectional correlation, from the $N \times P_c$ matrix \bar{C}_t , we simulate characteristics according to

$$\hat{C}_t = W \bar{C}_t \quad (19)$$

Simulating Characteristics

Finally, the "observed" characteristics for each $1 \leq i \leq N$ and for $j = 1, \dots, P_c$ are constructed according to:

$$c_{ij,t} = \frac{2}{n+1} \text{rank}(\hat{c}_{ij,t}) - 1. \quad (20)$$

with the rank transformation normalizing all predictors to be within $[-1, 1]$

Simulating Macroeconomic Time Series

For simulation of x_t , a 3×1 multivariate time series, we consider a VAR model:

$$x_t = Ax_{t-1} + u_t, \quad u_t \sim N(\mu = (0, 0, 0)', \Sigma = \mathbb{I}_3) \quad (21)$$

Simulating Macroeconomic Time Series

Consider 3 different specifications for matrix A :

$$A_1 = \begin{pmatrix} .95 & 0 & 0 \\ 0 & .95 & 0 \\ 0 & 0 & .95 \end{pmatrix} \quad (22)$$

$$A_2 = \begin{pmatrix} 1 & 0 & .25 \\ 0 & .95 & 0 \\ .25 & 0 & .95 \end{pmatrix} \quad (23)$$

$$A_3 = \begin{pmatrix} .99 & .20 & .10 \\ .20 & .90 & -.30 \\ .10 & -.30 & -.99 \end{pmatrix} \quad (24)$$

Simulating Return Series

We will consider four different functions $g(\cdot)$:

$$(1) g_1(z_{i,t}) = (c_{i1,t}, c_{i2,t}, c_{i3,t} \times x'_t) \theta_0$$

$$(2) g_2(z_{i,t}) = (c_{i1,t}^2, c_{i1,t} \times c_{i2,t}, \text{sgn}(c_{i3,t} \times x'_t)) \theta_0$$

$$(3) g_3(z_{i,t}) = (1[c_{i3,t} > 0], c_{i2,t}^3, c_{i1,t} \times c_{i2,t} \times 1[c_{i3,t} > 0], \text{logit}(c_{i3,t})) \theta_0$$


$$(4) g_4(z_{i,t}) = (\hat{c}_{i1,t}, \hat{c}_{i2,t}, \hat{c}_{i3,t} \times x'_t) \theta_0$$

$g_1(z_{i,t})$ is a linear specification

$g_2(z_{i,t})$ and $g_3(z_{i,t})$ is a non-linear specification with interactions

$g_4(z_{i,t})$ builds returns using \hat{c} , which are the unobserved characteristics without cross sectional correlation built in

θ^0 was tuned so that the cross sectional R^2 was around 25%, and the predictive R^2 5%.

The simulation design results in $3 \times 4 = 12$ different simulated datasets, 

Sample Splitting

$T = 180$ monthly periods corresponds to 15 years. The training sample was set to start from $T = 108$ or 9 years, a validation set 1 year in length. The last 3 years were reserved as a test set never to be used for validation or training.

Sample Splitting

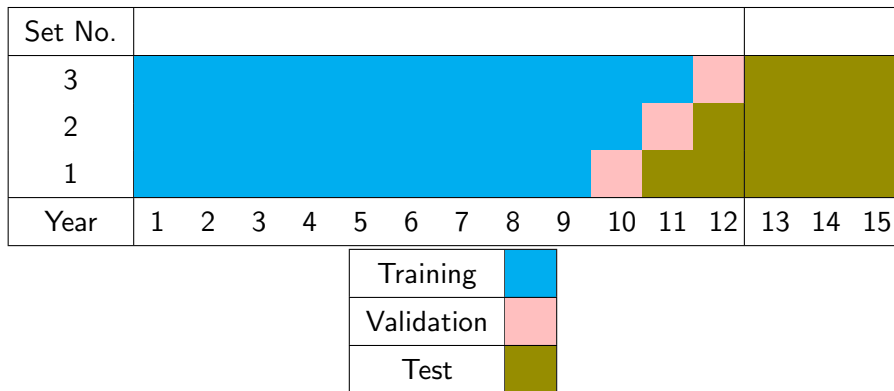


Figure 4: Sample Splitting Procedure

Data Source

CRSP/Compustat database for stock returns with stock level characteristics such as accounting ratios and macroeconomic factors will be queried.

Most previous studies have included decades of data in their training sample - this does not make much sense as several factors are different now
Only more recent data will be used, such as period before and after 2008
GFC

Out of Sample R Squared

Overall predictive performance for individual excess stock returns were assessed using the out of sample R^2 :

$$R_{OOS}^2 = 1 - \frac{\sum_{(i,t) \in \mathcal{T}_3} (r_{i,t+1} - \hat{r}_{i,t+1})^2}{\sum_{(i,t) \in \mathcal{T}_3} (r_{i,t+1} - \bar{r}_{i,t+1})^2} \quad (25)$$

where \mathcal{T}_3 indicates that the fits are only assessed on the test subsample, which is never used for training or tuning.

Diebold Mariano Tests for Predictive Accuracy

- The Diebold-Mariano test (Diebold and Mariano (2002) and Harvey et al. (1997)) compares the forecast accuracy of two forecast methods
- Different to the overall R squared metric because it tests whether or not the models' forecast accuracy is significantly different
- Tests whether or not the difference series ($d_t = e_{1t} - e_{2t}$) between two forecast methods' errors is different from zero
- As all models in this paper will be producing forecasts for an entire cross section of stocks, e_{1t} and e_{2t} will instead represent the average forecast errors for each model

Diebold Mariano Tests for Predictive Accuracy

Under the null hypothesis (forecast errors from compared models are the same):

$$S_1^* = \left[\frac{n+1-2h+n^{-1}h(h-1)}{n} \right]^{1/2} S_1; \quad S_1^* \sim N(0,1) \quad (26)$$

$$S_1 = \left[\hat{V}(\bar{d}) \right]^{-1/2} \bar{d} \quad (27)$$

$$\hat{\gamma}_k = n^{-1} \sum_{t=k+1}^n (d_t - \bar{d})(d_{t-k} - \bar{d}) \quad (28)$$

$$V(\bar{d}) \approx n^{-1} \left[\gamma_0 + 2 \sum_{k=1}^{h-1} \gamma_k \right] \quad (29)$$

where d_t represents the difference series between the forecast errors of the two models $e_{1t} - e_{2t}$.

Variable Importance

The importance of each predictor j is denoted as VI_j , and is defined as the reduction in predictive R-Squared from setting all values of predictor j to 0, while holding the remaining model estimates fixed.

Despite obvious limitations, this allows us to visualize which factors machine learning algorithms have determined to be important.

Results

References

- Breiman, L., 2001. Random forests. Machine learning 45, 5–32.
- Diebold, F. X., Mariano, R. S., 2002. Comparing predictive accuracy. Journal of Business & economic statistics 20, 134–144.
- Feng, G., He, J., Polson, N. G., 2018. Deep Learning for Predicting Asset Returns. arXiv:1804.09314 [cs, econ, stat] ArXiv: 1804.09314.
- Freyberger, J., Neuhierl, A., Weber, M., 2017. Dissecting characteristics nonparametrically. Tech. rep., National Bureau of Economic Research.
- Gu, S., Kelly, B., Xiu, D., 2018. Empirical asset pricing via machine learning. Tech. rep., National Bureau of Economic Research.
- Harvey, C. R., Liu, Y., 2019. A Census of the Factor Zoo p. 7.
- Harvey, D., Leybourne, S., Newbold, P., 1997. Testing the equality of prediction mean squared errors. International Journal of Forecasting 13, 281–291.

Questions and Answers