A Neural Network with Skewed Copula Method on Portfolio Optimization

Group 3 members

Li, Weijing

Liu, Sihang

Mo, Shuangning

Zhang, Meihui

Zhang, Yuyang

Zhou, Yukang

Outline

Algorithms to Predict Return

- ARMA (Benchmark)
- MLP: Multi-Layer Perceptron
- RNN: Recurrent Neural Network ✓
- PSN: Psi Sigma Network

Methods to Predict Covariance

- (Dynamic Conditional Correlation) DCC-Garch
- (Asymmetric DCC) ADCC-Garch ✓

Copulas to generate cVaR

- Gaussian
- Clayton
- Student‘s t
- Skewed Student’s t ✓

DATA

- Five stocks over the period of 01/05/2015 - 02/01/2020, the first four years' data sets as the training data, and the last year as the testing data

- The 'AAPL', 'AMZN', 'NKE', 'MSFT', 'GOOG'.

- High liquidity and high volume of assets are expected to perform better on other that are less liquid and less covered.

- Using these stocks can be considered as a tough to beat benchmark.

Benchmark: Autoregressive moving average model (ARMA)

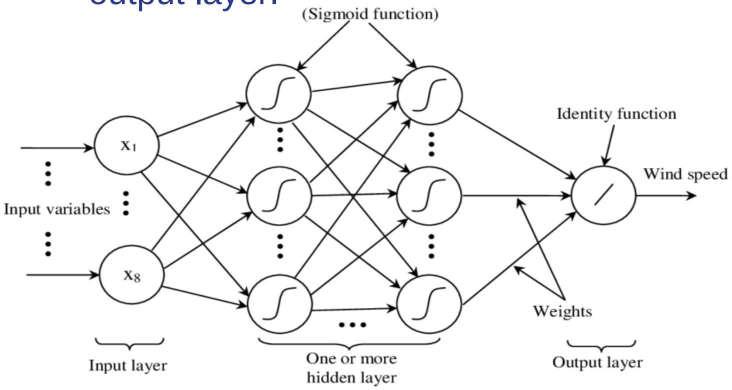
Def: In the statistical analysis of time series, **autoregressive–moving-average** (**ARMA**) **models** provide a parsimonious description of a (weakly) stationary stochastic process in terms of two polynomials, one for the autoregression (AR) and the second for the moving average (MA).

- AR: Regressing the variable on its own lagged (i.e., past) values.
- MA: Modeling the error term as a linear combination of error terms occurring contemporaneously and at various times in the past.
- ARMA(*p*,*q*) model where *p* is the order of the AR part and *q* is the order of the MA part

$$Y_t = \widehat{\varphi_0} + \widehat{\varphi_1} Y_{t-1} + \widehat{\varphi_2} Y_{t-2} + \cdots + \widehat{\varphi_p} Y_{t-\hat{p}} + \hat{\varepsilon}_t - \widehat{w_1}\widehat{\varepsilon_{t-1}} - \widehat{w_2}\widehat{\varepsilon_{t-2}} - \widehat{w_2}\widehat{\varepsilon_{t-2}} - \cdots - \widehat{w_p}\widehat{\varepsilon_{t-\hat{q}}}$$

Neural network method 1: Multi-Layer Perceptron (MLP)

- An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer.



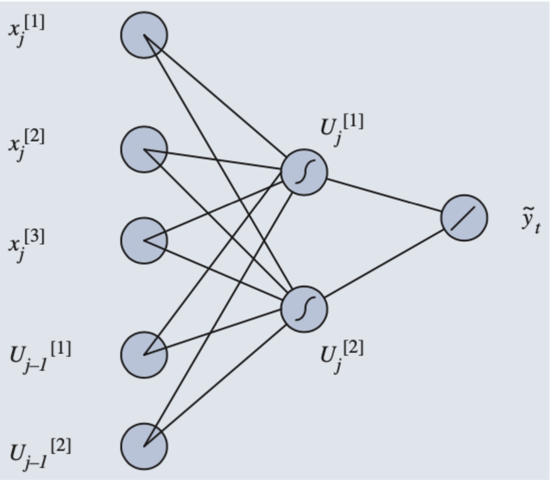
The circle with a 'squiggle' inside is the transfer sigmoid function: $s(x) = \frac{1}{1+e^{-x}}$

The circle with a slash inside is a linear function: $F(x) = \sum x_i$

The error function to be minimized: $E(u_{jk}, w_j) = \frac{1}{T}\sum_{t=1}^{T}(y_t - \tilde{y}_t(u_{jk}, w_j))^2$

- The training of the network starts with randomly chosen weights
- Fitting the training data using a learning algorithm called back-propagation of errors
- Maximizing the forecasting accuracy for the test dataset
- The predictive value of the model is evaluated applying it to the validation dataset (out-of-sample dataset).

Neural network method 2: Recurrent Neural Network (RNN)



Elman(1990) recurrent neural network architecture with two nodes in the hidden layer.

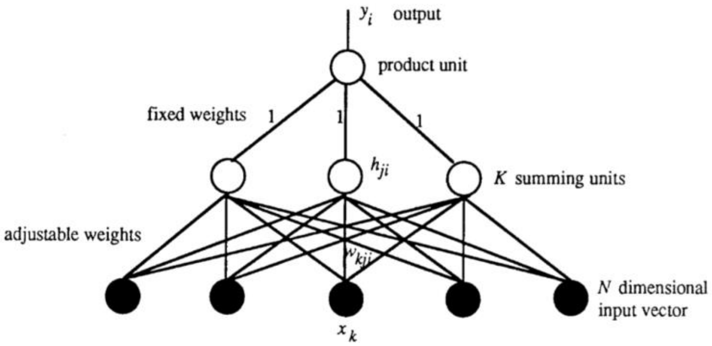
1. Model input $x_t^{[n]}(n = 1, 2, \ldots, k+1)$ $u_t^{[1]}$ and $u_t^{[2]}$
2. Model output
   3. $\tilde{y}_t$: recurrent model output
   4. $d_t^{[f]}$ $(f = 1, 2)$ $w_t^{[n]}$ $(n = 1, 2, \ldots, k+1)$: network weights
   5. $U_t^{[f]}$ $(f = 1, 2)$:outputs of hidden nodes at time t
   6. : Sigmoid function $S(x) = \frac{1}{1 + e^{-x}}$
   7. : Linear output function: $F(x) = \sum_i x_i.$

Minimize error function: $E(d_t, w_t) = \frac{1}{T}\sum_{t=1}^{T}(y_t - \tilde{y}_t(d_t, w_t))^2$

- A simple RNN has an activation feedback which embodies short-term memory.
- RNN: inputs are taken from all previous values -->provide more accurate outputs
- RNN require more computational time and yield better results than MLPs.

Neural network method 3: Psi Sigma Network (PSN)

- The training speeds for MLP are slower than PSNs which considered as <u>a class of feedforward fully connected Higher Order Neural Network.</u>



The hidden layer: $h_j = w_j^T x = \sum_{k=1}^{N} w_{kj} x_k + w_{oj}, \quad j = 1, 2, \ldots, K,$

The output unit adaptive sigmoid activation function with c the adjustable term: $\sigma(x) = \frac{1}{1 + e^{-xc}}$ $\quad \tilde{y} = \sigma(\Pi_{j=1}^{k} h_j)$

The error function to be minimized is: $E(c, w_j) = \frac{1}{T} \sum_{t=1}^{T} (y_t - \tilde{y}_t(w_k, c))^2$

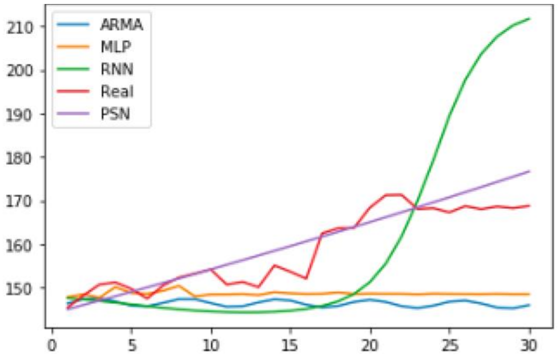
- A network combines the fast learning property of single layer networks with the powerful mapping capability of HONNs

- The weights from the hidden to the output layer are fixed to 1

- Only the weights from the input to the hidden layer are adjusted, something that <u>greatly reduces the training time</u>.

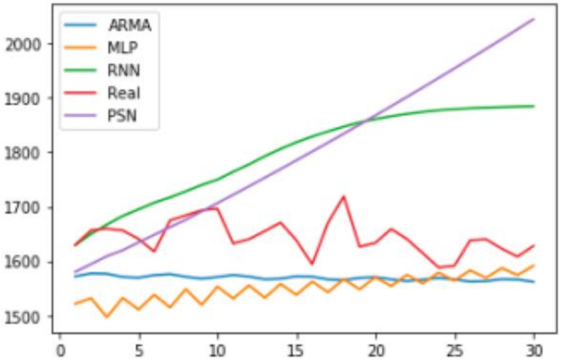
Forecasting Models' Performance

AAPL



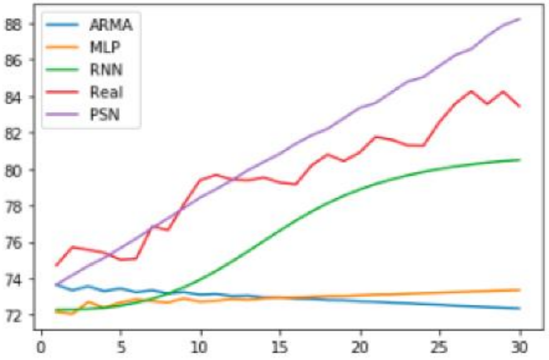
MLP	RNN	PSN
0.263	0.018	0.72

AMZN



MLP	RNN	PSN
0.35	0.022	0.80

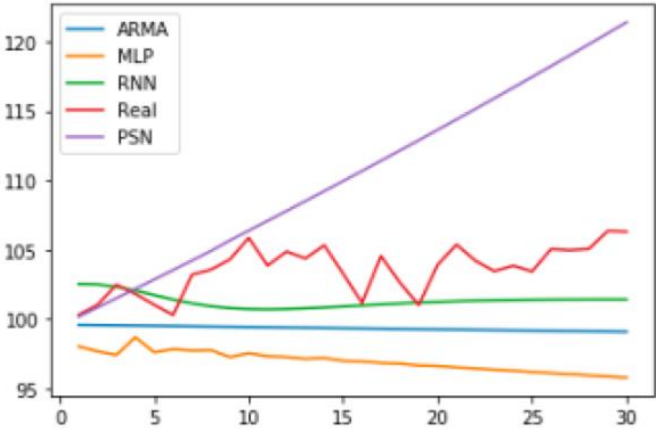
NKE



MLP	RNN	PSN
0.235	0.0166	0.90

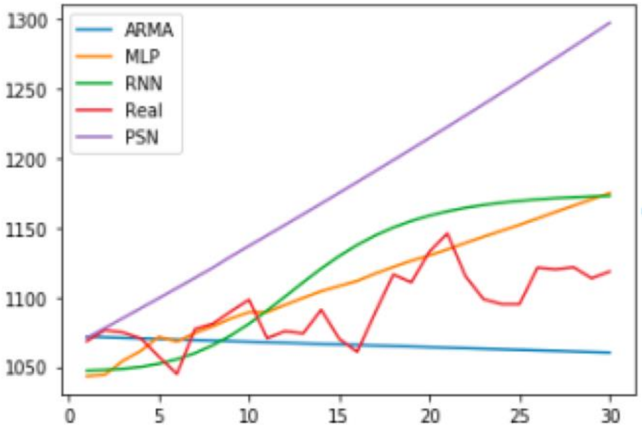
Forecasting Models' Performance

MSFT



MLP	RNN	PSN
0.225	0.0379	0.78

GOOG



MLP	RNN	PSN
0.218	0.0141	0.95

Conditional variance forecasted by (Dynamic Conditional Correlation) DCC-Garch

- Assume we have k assets
- $r_t \mid \Omega_{t-1} \sim N(0, H_t) \quad H_t = D_t R_t D_t$
  - r_t : Return on asset
  - Ω_{t-1} : information upon time t-1
  - $D_t = diag\{h_{i,t}\}$
  - R_t :Correlation matrix
- Log-likelihood function

$$L = -\frac{1}{2}\sum_{t=1}^{T}\left[k\log(2\pi)+\log(|H_t|)+r_t' H_t^{-1} r_t\right]$$

$$= -\frac{1}{2}\sum_{t=1}^{T}\left[k\log(2\pi)+\log(|D_t R_t D_t|)+r_t' D_t^{-1} R_t^{-1} D_t^{-1} r_t\right]$$

$$= -\frac{1}{2}\sum_{t=1}^{T}\left[k\log(2\pi)+\log(|D_t|)+\log(|R_t|)+\varepsilon_t' R_t^{-1} \varepsilon_t\right]$$

- Univariate GARCH model:

$$h_{it} = \omega_i + \sum_{p=1}^{P_i} \alpha_{ip} \varepsilon_{it-p}^2 + \sum_{q=1}^{q_i} \beta_{iq} h_{it-q}$$

  - α_{ip} : coefficient of the square term of the previous residual
  - β_{iq} : coefficient of the previous conditional variance

Need to meet the non-negative and stable conditions: $\alpha_m \geq 0, \ \beta_n \geq 0, \ \sum_{m=1}^{M} \alpha_m + \sum_{n=1}^{N} \beta_n < 1$

$$R_t = Q_t^{*-1} Q_t Q_t^{*-1}$$

$$Q_t = \left(1 - \sum_{m=1}^{M} \alpha_m - \sum_{n=1}^{N} \beta_n\right)\bar{Q} + \sum_{m=1}^{M} \alpha_m (\varepsilon_{t-m} \varepsilon_{t-m}') + \sum_{n=1}^{N} \beta_n Q_{t-n}$$

Conditional variance forecasted by (Asymmetric DCC) ADCC-Garch

- The ADCC model is <u>a generalized version of the DCC model, which permits conditional asymmetries in correlations.</u>

- In the scalar A-DCC model,

$$Q_t = (\bar{P} - a^2\bar{P} - b^2\bar{P} - g^2\bar{N}) + a^2\varepsilon_{t-1}\varepsilon'_{t-1} + g^2 n_{t-1}n'_{t-1} + b^2 Q_{t-1},$$

$$\bar{P} = E[\varepsilon_t\varepsilon'_t] \quad \bar{N} = E[n_t n'_t]$$

- A necessary and sufficient condition is $\alpha^2+\beta^2+\delta g^2<1$ (where δ = maximum eigenvalue [P^-.5 N P^-.5]).

- This constraint can be implemented during estimation of the conditional correlation.
  The estimation of the scalar ADCC is no more difficult than the scalar DCC.

Outcome

- **ADCC: optimal set 0.05，0.65，0.1**
- **DCC: optimal set 0.05，0.25**

Modelling Marginal Density

- Since autocorrelation and heteroscedasticity, we use the conditional mean is modelled with a simple ARMA model to address problem in autocorrelation.

$$r_{i,t} = c + \sum_{j=1}^{p} \varphi_j r_{i,t-j} + \sum_{k=1}^{q} \theta_k \varepsilon_{i,t-k} + \varepsilon_{i,t} \qquad \varepsilon_{i,t} = \sigma_{i,t} Z_{i,t}$$

- To capture the heteroscedasticity and asymmetric volatility clustering of stock returns. We model the conditional variance using the GJR-GARCH dynamics:

$$\sigma_{i,t}^2 = \omega + \sum_{j=1}^{p} \alpha_j \varepsilon_{i,t-j}^2 + \sum_{k=1}^{q} \beta_k \sigma_{i,t-k}^2 + \sum_{k=1}^{q} \gamma_k \varepsilon_{i,t-k}^2 I\left[\varepsilon_{i,t-k} < 0\right]$$

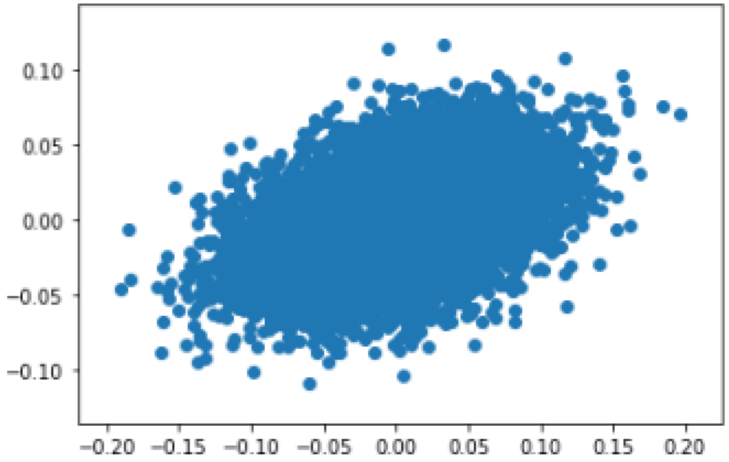
- Estimated standardized residual:

$$z_{i,t} = \frac{r_{i,t} - c - \sum_{j=1}^{p} \varphi_j r_{i,t-j} - \sum_{k=1}^{q} \theta_k \varepsilon_{i,t-k}}{\sigma_{i,t}}$$

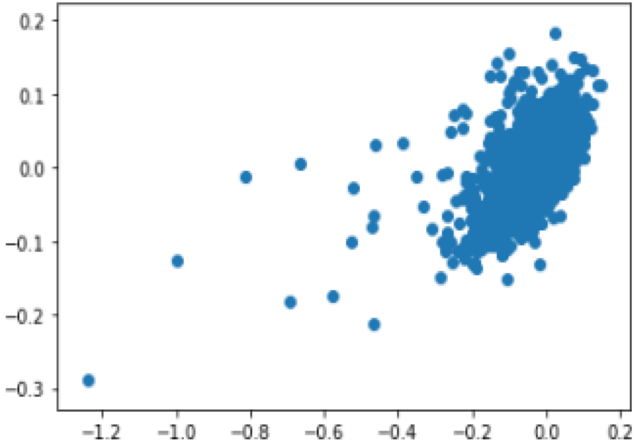
- Because of significant skewness and the hypothesis of normality is rejected by the Jarque-Bera test
  - Use the $z_{i,t} \sim \mathbf{F}_{skt}(\eta_i, \lambda_i)$ wed t distribution of Hansen (1994) to model the standardized residuals of each stock
  - $u_{i,t} = \mathbf{F}_{skt}\left(z_{i,t}; \eta_i, \lambda_i\right), \ \eta_i \in (2, \infty), \ \lambda_i \in [-1, 1]$

Copula MC Simulation:

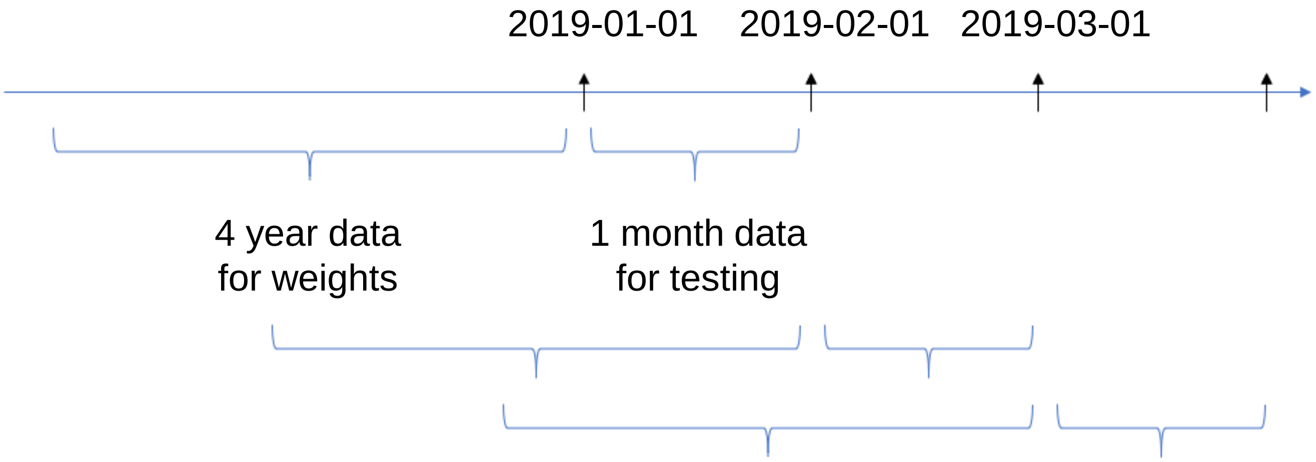
Gaussian Copula



Skewed t Copula

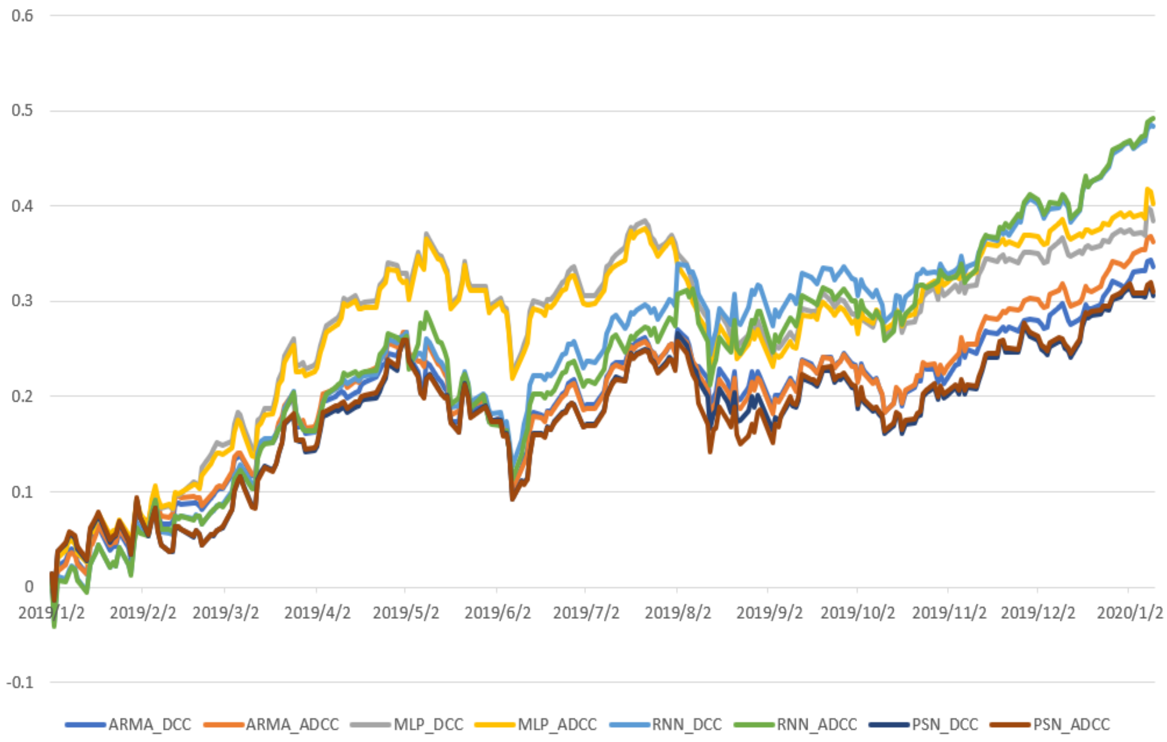


Out of sample test:



Out of sample test:

Out of sample test:



	Return	Std	Sharpe
ARMA_DCC	0.34	0.19	1.73
ARMA_ADCC	0.36	0.19	1.90
MLP_DCC	0.38	0.20	1.88
MLP_ADCC	0.40	0.20	1.99
RNN_DCC	0.48	0.21	2.33
RNN_ADCC	0.49	0.22	2.29
PSN_DCC	0.31	0.21	1.47
PSN_ADCC	0.31	0.21	1.47

Reference:

- Zhao, Yang, et al. "Neural network copula portfolio optimization for exchange traded funds." *Quantitative Finance* 18.5 (2018): 761-775.
- Mba, Jules Clement, Edson Pindza, and Ur Koumba. "A differential evolution copula-based approach for a multi-period cryptocurrency portfolio optimization." *Financial Markets and Portfolio Management* 32.4 (2018): 399-418.
- Victor, H., Rustam Ibragimov, and Shaturgun Sharakhmetov. "Characterizations of joint distributions, copulas, information, dependence and decoupling, with applications to time series." *Optimality*. Institute of Mathematical Statistics, 2006. 183-209.