Stevens Institute of Technology

Financial Engineering

An Extension of Black-Litterman Model with AI-derived Views for ETFs Portfolio

Author:
Jian Shun Tan
Disha An

Supervisor:
Dr. Peter Lin

December 15, 2017

Abstract

Black-Litterman model is widely used by institutional investors since it was introduced by Black and Litterman of Goldman Sachs in 1992. Investor's view is the main feature that distinguish the Black-Litterman model with modern portfolio theory. Black-Litterman model uses a Bayesian approach to combine the subjective views of an investor with the market equilibrium vector of expected returns to form a new estimate of expected returns. This paper provides an application of the Black-Litterman model for portfolio management. The novel feature of this paper related to the extent literature on Black-Litterman methodology is that we use artificial intelligence method to obtain views as the proxies of investor's view in Black-Litterman model.

Keywords: Black-Litterman Model, Artificial Intelligence, Views, Portfolio

1 Introduction

Portfolio management is one of the most important parts in financial planning industry with practitioners who always try to improve their investment decisions. The divergence between traditional mean-variance portfolio and real world investment practice is extensive. This is primarily due to investment practitioners prefer to generate intuitive portfolio rather than to rely on traditional mean-variance portfolio method. One of the reasons might be that when allocating assets, mean-variance portfolio produces extreme weights, which is impractical in practice. Black and Litterman (1992) made a long-awaited step in narrowing the gap between investment theory and the practice. Black-Litterman model adding investor views into the traditional portfolio optimization theory. The results are portfolio weights allocation that depend on the investor confidence in his or her views and the willingness to assume risk in the portfolio.

Since Black-Litterman model was introduced in 1992, the determination of views in the model become one of the important question that investment practitioners have to answer. Zhou (2008) extends the Black-Litterman model by adding a data-generating process as an additional information for views. Beach and Orlov (2007) propose the use of GARCH-derived views as proxies for investor views in the Black-Litterman model. These research provide some pragmatic approaches for practical purpose, since there is no model can effectively describes an investor's views. This paper extends the Black-Litterman model by using the AI-derived views as proxies for investor views in the model.

2 Artificial Intelligence

Artificial intelligence (AI) has become a very important method for stock market prediction because of its ability to deal with uncertain, insufficient data which fluctuate rapidly in short period of time. Numerous research of AI in solving business problems have proven their advantage in relation to classical methods that do not include artificial intelligence.

AI in finance has its most frequent applications in stock performances and stock selection predictions. Quah and Srinivasan (1999) and Trippi and Turban (1992) apply artificial intelligence in stock selection, analysis of financial condition, business failure prediction, debt risk assessment, security market applications, and neural network approaches to financial forecasting. In this paper we applied two artificial intelligence methods, which are Artificial Neural Networks (ANN) and Recurrent Neural Network (RNN) to predict the views, monthly return of ETFs, and monthly return the Black-Litterman model.

2.1 Artificial Neural Network

Artificial Neural Networks (ANN) are computing systems inspired by the biological neural networks. This kind of system can learn by experiencing examples and use the learned experiences to do tasks, which is similar as a human brain. A simple illustration of the ANN can be seen in Figure 1.

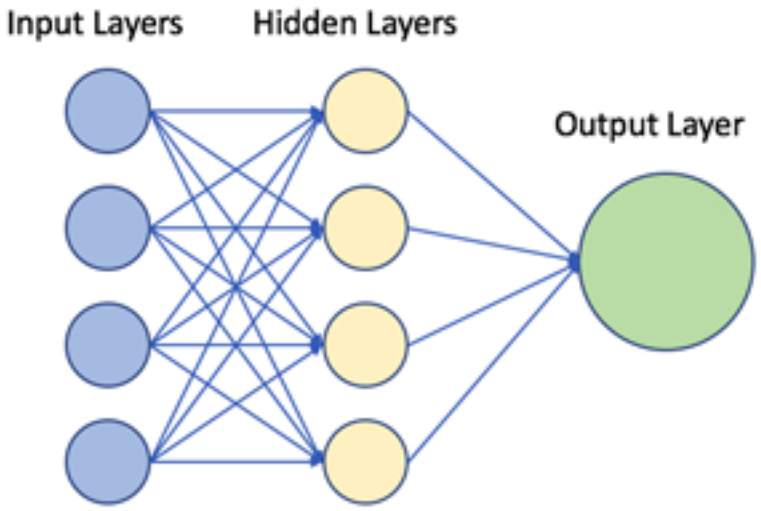

Figure 1: Architecture of Artificial Neural Network.

An ANN model consists of an input layer, at least one hidden layer and an output layer, each of which is connected to the other. At least one neuron should be employed in each layer of the ANN model. Input layers on the left part are the parameters considered to be put into the ANN algorithm. Hidden layers are the layers for the algorithm to assign different weights to each different neuron. The number of neurons in the hidden layer is determined empirically. The neurons of a layer are linked to the neurons of the neighboring layers with connectivity coefficients (weights). Using a learning procedure, these weights are adjusted to classify the given input patterns correctly for a given set of input-output pairs. The initial values of these weights are randomly assigned. The back-propagation learning algorithm is used to train the multi-layered feed forward ANN structure in the model. Back-propagation is the process that

minimizes the forecast error between predicted output and actual output by adjusting the weights in the hidden layers.

2.2 Recurrent Neural Network

Recurrent neural network (RNN) is an extension of ANN wchih can make use of sequential information. In traditional neural network it is assumed that all inputs and outputs are independent of each other, but it is not a good assumption for stock market prediction. RNN perform the same task for every element of a sequence, with the output being depended on the previous computations. RNN have a 'memory' which captures information about what has been calculated and fed back into next calculation process.

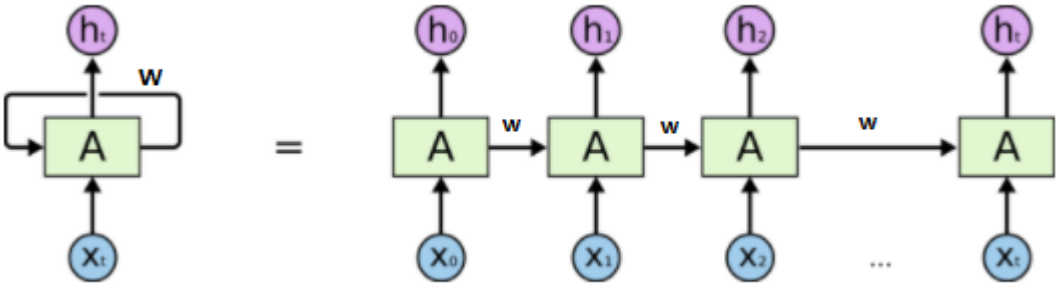

Figure 2: Recurrent Neural Network

Figure 2 shows a RNN unrolled into a full network. The formulas that govern the computation happening in a RNN are as follows:

- x_t is the input at time step t. For example, x_1 could be a time-series vector of stock prices.
- A is the hidden state at time step t. It's the memory of the network. A is calculated based on the previous hidden state and the input at the current step: $A_t = f(U_{x_t} + W_{A_{t-1}})$. The function f usually is an activation function such as tanh or ReLU. A_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeros.
- h_t is the output at step t. For example, if we wanted to predict the daily prices of next month, it would be a vector of daily prices.

Unlike a traditional neural network, which uses different parameters at each layer, a RNN shares the same parameters across all steps. This reflects the fact that they are performing the same task at each step, with different inputs. This process greatly reduces the total number of parameters we need to learn (Mikolov et al. (2010)).

2.2.1 Long Short Term Memory Networks

RNN have feedback loops in the recurrent layer which can maintain information in 'memory' over time. It can be difficult to train standard RNN to solve problems that

require learning long-term temporal dependencies. This is because the gradient of the loss function decays exponentially with time (called the vanishing gradient problem) (Bengio et al. (1994)).

Long Short Term Memory networks (LSTM) are a special kind of RNN, that uses special units in addition to standard units. They were introduced by Hochreiter and Schmidhuber (1997). LSTM units include a 'memory cell' that can maintain information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it's output, and when it's forgotten (Chung et al. (2014)). This architecture helps avoid longer-term dependencies problem.

The LSTM contains memory cells in the hidden layer. The memory cells store the information in temporal state of the network and use some special multiplicative units called gates to control the flow of information. The flow of information are control by a architecture contains an forget gate, an input gate and an output gate. The forget gate controls the flow of information from previous step and the inputs gate controls the flow of information from current step. A sigmoid layer is apply within both gates and decide how much of information will go through the gates. Sigmoid layer outputs a number between 0 and 1, 1 represent all information will go through the gate and 0 represent nothing will go through the gate. By controlling the flow of information from previous and current step, we will get the new adjusted information for the neural network learning process. Finally the output gate will apply an activation function to generate the output results.

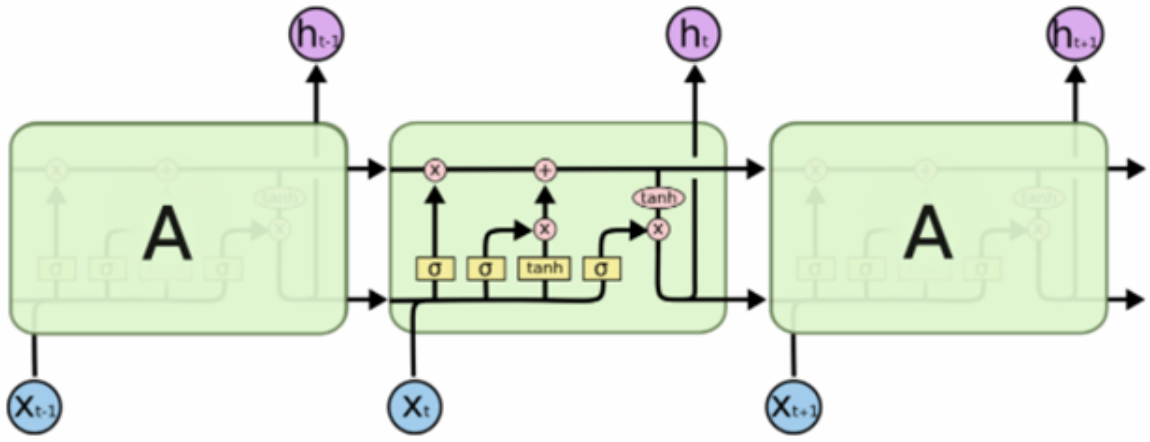

Figure 3: Long Short Term Memory Networks

An LSTM network computes a mapping from an input sequence $x = (x_1, ..., x_T)$ to an output sequence $h = (h_1, ..., h_T)$ by calculating the network unit activations using the following equations iterative from t = 1 to T:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{1}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$
$$C_t = f_t * C_{t-1} + i_t * g(W_C[h_{t-1}, x_t] + b_C) \tag{3}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{4}$$
$$h_t = o_t * h(C_t) \tag{5}$$

where the W terms denote weight matrices, the b terms denote bias vectors, sigma is the logistic sigmoid function, and i, f, o and C are the input gate, forget gate, output gate and cell activation vectors, g and h are the cell input and cell output activation functions respectively.

3 Black-Litterman Model

3.1 Discussion

In the investment industry, the Black-Litterman model is a mathematical model for portfolio allocation developed by Goldman Sachs in 1990 and published in 1992. This model has been available to institutional investors and financial advisor and is used by numerous investment banks. The Black-Litterman asset allocation model utilizes CAPM theory (to establish equilibrium returns and risk), Sharpes reverse optimization, Bayesian mixed estimation, and mean variance optimization based on Markowitz (1952). Some technical works that introduce the Black-Litterman model and attempt to make it more accessible to practitioners are Bevan and Winkelmann (1998), He and Litterman (1999), Satchell and Scowcroft (2000) and Drobetz (2001).

The Black-Litterman model seeks to overcome the universal problem that institutional investors have encountered in applying modern portfolio theory: although the covariance of a few assets can be adequately estimated, it is difficult to come up with reasonable estimates of expected returns. This model overcome this problem by assumes that the initial expected returns are whatever is required so that the equilibrium asset allocation is equal to what we observe in the markets. The views held by the investor regarding expected returns are an additional input to the asset allocation decision. In the research presented in this paper, AI models are used to provide the investor views.

3.2 Black-Litterman Equation

The formulas given below are known as the Black-Litterman equation and represent the expected return vector and covariance matrix that are established from a Bayesian mixing of the implied equilibrium excess return vector (Π) and the vector of investors views (Q). The new return vector and covariance matrix calculated from Black-Litterman model will apply into traditional mean-variance portfolio method to generated the Black-Litterman portfolio. The descriptions of Black-Litterman parameters given in Table 1.

$$E(R) = [(\tau\Sigma)^{-1} + P'\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\Pi + P'\Omega^{-1}Q] \quad (6)$$

$$Var(R) = [(\tau\Sigma)^{-1} + P'Q^{-1}P]^{-1} \quad (7)$$

Table 1: Variables of Black-Litterman Model

Black-Litterman Variables		
Parameters	Description	Value/Data
τ	Value between 0 and 1, reflects confidence correlation in views	0.01
Σ	N x N implies covariance matrix of excess return	Derived from historical data
w_{eq}	Portfolio initial weights	Derived from historical data by using Markowitz optimization method
Π	N x 1 implies equilibrium return vector	Derived from historical data
P	Kx N matrix identifies the assets with views	Diagonal identity matrix; the views imposed for all assets
Q	K x 1 matrix contain views	Monthly returns predicted from AI-models
Ω	K x K matrix expresses the confidence in views	$\Omega = P\Sigma P'$

4 Data

We have applied the same dataset period for both ANN and RNN view prediction AI-models. The technical indicators in ANN model were downloaded from Python package called stockstats. The ETFs included in our algorithms are shown in Table 2:

Table 2: All ETFs in Dataset

ETFs	Description
IJK	iShares S&P MidCap 400 Growth
IVV	iShares S&P MidCap 400 Growth
IWP	iShares Russell Midcap Growth Index
IWV	iShares Russell 3000 Index
IYY	iShares Dow Jones U.S. Index Fund
LQD	iShares iBoxx $ Invmt Grade Corp Bond Fund
VOT	Vanguard Mid-Cap Growth

For both AI models we discussed in this paper, the training dataset period is set as January 2010 to December 2016 and the testing dataset period is set as January 2017 to October 2017.

5 Black-Litterman Model Views Prediction

For the Black-Litterman model, we have designed the portfolios which investment period is from January 2017 to October 2017 and the portfolios were rebalanced every month. On the day right before each month, we applied ANN and RNN view prediction models to predict the monthly returns for the next following month of every ETFs and implemented the predictions as the investor's views in Black-Litterman model.

5.1 ANN-Derived Views

The first AI model we have applied to predict the monthly returns of ETFs is Artificial Neural Networks. This model consists of an input layer, four hidden layers and an output layer, each of which is connected to the other. Inputs for the network were daily log returns of previous 20 days and 24 technical indicators which were represented by 44 neurons in the input layer. Output of the network was monthly log returns of ETFs. The output layer of the network consisted of only one neuron that represents the monthly log returns. The mean squared error (MSE) was used to evaluate the performance of the ANN model. The gradient-descent method was used as the weight update algorithm to minimize MSE. The ANN parameters and their levels are summarized in Table 3.

Table 3: Parameters of the Artificial Neural Network model

Artificial Neural network Parameters	
Parameters	Values/Data
AI-Model	Artificial Neural Network
Training Set	January 2010 - December 2016
Testing Set	January 2017- October 2017
Inputs	Daily returns of previous 20 days and 24 technical indicators
Outputs	Monthly returns of next 20 trading days
Hidden Layer	4
Hidden Neurons	35
Dropout	0.2
Optimizer Function	adam

The input layers in this ANN model consist of two parts: historical log returns and technical indicators. Daily log returns is calculated for each ETFs, for each time step daily returns of previous 20 days will be applied as a part of inputs in the model. 24 technical indicators are chosen as another inputs in the ANN model. Selected 24 technical indicators are shown in Table 4:

Table 4: Technical indicators for ANN model

Indicators	Description	Indicators	Description
MACD	Moving Average Convergence Divergence	wr10	10 days Williams %R
cr	CR indicator	wr6	6 days Williams %R
cr ma1	CR indicator, including 5 days moving average	cci14	14 days Commodity Channel Index
cr ma2	CR indicator, including 10 days moving average	cci20	20 days Commodity Channel Index
cr ma3	CR indicator, including 20 days moving average	tr	True Range
kdjd	Line D in KDJ indicator. Default to 9 days	atr	Average True Range
kdjj	Line J in KDJ indicator. Default to 9 days	dma	DMA difference of 10 and 50 moving average
macds	MACD signal line	pdi	Positive Directional Indicator
macdh	MACD histogram	mdi	Negative Directional Indicator
bolling	Middle Bollinger band. 20-day Simple Moving Average	dx	Directional Movement Index
rsi6	6 days Relative Strength Index	trix	TRIX
rsi12	12 days Relative Strength Index	vr	Volume Reversal

We assume that there are 20 trading days for each month. In this ANN model, we use the daily return of previous 20 trading days and 24 technical indicators to predict the monthly returns of next 20 trading days. Other parameters, such as the number of hidden neurons, dropout rate and optimizer function are chosen by using GridSearchCV function in Python. This function creates loops and tries all possible combinations of these parameters and will result in a combination of parameters which have lowest prediction error. The result is in a form of the comparison between the real log return and the predicted log return based on different months as well as different ETFs. Figure 4 is the prediction results shown in a format of heatmap. The accuracy of prediction moving directions and forecast error are shown in Table 5 and Table 6.

		ETF						
Month	Category	IJK	IVV	IWP	IWV	IYY	LQD	VOT
Jan	Actual	1.5521%	1.0318%	2.5879%	1.0768%	1.3288%	0.0597%	2.9704%
	Predicted	-0.0428%	-1.0010%	-0.1100%	-1.1326%	-0.3676%	0.5128%	0.9745%
Feb	Actual	3.3860%	3.8456%	2.9525%	3.6043%	3.6914%	0.0991%	2.9434%
	Predicted	-2.5175%	-2.6125%	-0.7175%	-2.2159%	-2.1029%	0.4002%	-0.6269%
Mar	Actual	-1.7571%	-1.2138%	-0.5601%	-1.3171%	-1.2409%	0.2973%	-0.5772%
	Predicted	-1.6128%	0.9444%	0.2751%	0.4515%	0.2595%	0.5405%	0.6308%
Apr	Actual	1.9430%	1.1292%	2.0431%	1.2530%	1.2465%	0.3130%	2.5204%
	Predicted	1.7526%	0.6947%	-0.0168%	0.6413%	0.5917%	0.9149%	-0.1876%
May	Actual	-0.0565%	1.1095%	2.1657%	0.7383%	0.7975%	0.3471%	1.4757%
	Predicted	0.3863%	2.7027%	2.5284%	3.2796%	3.0191%	0.8037%	2.2817%
Jun	Actual	-0.2352%	-0.1511%	-1.0677%	-0.0278%	-0.0187%	0.4574%	-0.8687%
	Predicted	0.1700%	0.8805%	1.6575%	1.1075%	1.4817%	0.8201%	1.7611%
Jul	Actual	0.2577%	1.8647%	1.6495%	1.6007%	1.7207%	0.7338%	1.7261%
	Predicted	-0.3899%	0.7152%	1.0472%	1.6081%	1.5422%	0.4056%	1.1518%
Aug	Actual	-1.3672%	0.0602%	0.5544%	-0.0683%	-0.0161%	0.9791%	0.1334%
	Predicted	-1.0459%	0.0041%	-2.1385%	-0.7048%	-1.0867%	0.5959%	-1.0385%
Sep	Actual	3.0692%	1.8484%	2.4287%	2.1725%	1.9372%	1.4355%	1.5460%
	Predicted	5.3146%	1.9356%	2.7452%	2.8093%	2.5418%	0.3902%	2.6715%
Oct	Actual	2.5915%	1.8878%	2.2240%	1.6262%	1.7860%	1.6349%	1.8489%
	Predicted	1.9954%	1.0482%	0.4235%	0.1878%	0.0149%	0.3508%	0.7559%

Figure 4: Artificial Neural Network predictions

Table 5: Summary of accuracies and errors of the predicted monthly returns based of months

Month	JAN	FEB	MAR	APR	MAY
Accuracy	28.57%	14.29%	28.57%	71.43%	85.71%
MSE	0.0369%	0.0243%	0.0174%	0.0246%	0.0542%
Month	JUN	JUL	AUG	SEP	OCT
Accuracy	14.29%	85.71%	71.43%	100%	100%
MSE	0.009%	0.0021%	0.004%	0.0151%	0.0454%

Table 6: Summary of accuracies and errors of the predicted monthly returns based on ETFs

ETFs	LQD	IYY	VOT	IWV	IWP	IJK	IVV
Accuracy	77.09&	74.89%	74.45%	74.01%	73.57%	70.04%	70.04%
MSE	0.0059%	0.0304%	0.0381%	0.0293%	0.0412%	0.0384%	0.0314%

5.2 RNN-Derived Views

RNN Long Short Term Memory Networks model is created to predict the views for Black- Litterman model. This model consists of an input layer, four hidden layers and an output layer, each of which is connected to the other. Inputs for the network are daily log returns of previous 120 days. Output of the network is monthly log returns of ETFs. The output layer of the network consists of only one neuron that

represents the monthly log returns. The mean squared error (MSE) is used to evaluate the performance of the ANN model. The gradient-descent method is used as the weight update algorithm to minimize MSE. The RNN parameters and their levels are summarized in Table 7.

Table 7: Parameters of the Recurrent Neural Network model

Recurrent Neural network Parameters	
Parameters	Values/Data
AI-Model	Recurrent Neural Network-LSTM
Training Set	January 2010 - December 2016
Testing Set	January 2017- October 2017
Inputs	Daily returns of previous 120 days
Outputs	Monthly returns of next 20 trading days
Hidden Layer	4
Hidden Neurons	50
Dropout	0.2
Optimizer Function	adam

In this model, we assumes that there are 20 trading days for each month. In this RNN model, we use the daily return of previous 6 months (120 trading days) to predict the monthly returns of next 20 trading days. The number of hidden layers and hidden neurons, dropout rate and optimizer function are determined empirically.

As the market might differ in terms of currency and index representation, all data is normalized for each ETFs. The data was normalized using the following equations, where P is the complete time series.

$$p_{norm} = \frac{p - min(P)}{max(P) - min(P)} \tag{8}$$

The result of monthly return predictions is shown in Figure 5, The predicted monthly return will be considered as the proxies of views in Black-Litterman model. The accuracy of prediction moving directions and forecast error are shown in Table 8 and Table 9. On average, the accuracy of the predictions of moving direction is around 77%, which means in this model we are able to predict whether the ETFs price is moving up or going down with an accuracy of 77%.

		ETF						
Month	Category	IJK	IVV	IWP	IWV	IYY	LQD	VOT
Jan	Actual	1.5521%	1.0318%	2.5879%	1.0768%	1.3288%	0.0597%	2.9704%
	Predicted	-0.0428%	-1.0010%	-0.1100%	-1.1326%	-0.3676%	0.5128%	0.9745%
Feb	Actual	3.3860%	3.8456%	2.9525%	3.6043%	3.6914%	0.0991%	2.9434%
	Predicted	-2.5175%	-2.6125%	-0.7175%	-2.2159%	-2.1029%	0.4002%	-0.6269%
Mar	Actual	-1.7571%	-1.2138%	-0.5601%	-1.3171%	-1.2409%	0.2973%	-0.5772%
	Predicted	-1.6128%	0.9444%	0.2751%	0.4515%	0.2595%	0.5405%	0.6308%
Apr	Actual	1.9430%	1.1292%	2.0431%	1.2530%	1.2465%	0.3130%	2.5204%
	Predicted	1.7526%	0.6947%	-0.0168%	0.6413%	0.5917%	0.9149%	-0.1876%
May	Actual	-0.0565%	1.1095%	2.1657%	0.7383%	0.7975%	0.3471%	1.4757%
	Predicted	0.3863%	2.7027%	2.5284%	3.2796%	3.0191%	0.8037%	2.2817%
Jun	Actual	-0.2352%	-0.1511%	-1.0677%	-0.0278%	-0.0187%	0.4574%	-0.8687%
	Predicted	0.1700%	0.8805%	1.6575%	1.1075%	1.4817%	0.8201%	1.7611%
Jul	Actual	0.2577%	1.8647%	1.6495%	1.6007%	1.7207%	0.7338%	1.7261%
	Predicted	-0.3899%	0.7152%	1.0472%	1.6081%	1.5422%	0.4056%	1.1518%
Aug	Actual	-1.3672%	0.0602%	0.5544%	-0.0683%	-0.0161%	0.9791%	0.1334%
	Predicted	-1.0459%	0.0041%	-2.1385%	-0.7048%	-1.0867%	0.5959%	-1.0385%
Sep	Actual	3.0692%	1.8484%	2.4287%	2.1725%	1.9372%	1.4355%	1.5460%
	Predicted	5.3146%	1.9356%	2.7452%	2.8093%	2.5418%	0.3902%	2.6715%
Oct	Actual	2.5915%	1.8878%	2.2240%	1.6262%	1.7860%	1.6349%	1.8489%
	Predicted	1.9954%	1.0482%	0.4235%	0.1878%	0.0149%	0.3508%	0.7559%

Figure 5: Recurrent Neural Network predictions

Table 8: Summary of accuracies and errors of the predicted monthly returns based of months

Month	JAN	FEB	MAR	APR	MAY
Accuracy	85.71%	85.71%	14.19%	100%	85.71%
MSE	0.0041%	0.0502%	0.0942%	0.0049%	0.0243%
Month	JUN	JUL	AUG	SEP	OCT
Accuracy	14.19%	85.71%	57.14%	85.71%	85.71%
MSE	0.0784%	0.0139%	0.0562%	0.0085%	0.0204%

Table 9: Summary of accuracies and errors of the predicted monthly returns based on ETFs

ETFs	LQD	IYY	VOT	IWV	IWP	IJK	IVV
Accuracy	76.21%	76.21%	74.45%	80.18%	82.82%	73.13%	75.33%
MSE	0.0039%	0.0170%	0.0335%	0.0154%	0.0351%	0.0774%	0.0230%

6 Black-Litterman Portfolio

In previous sections, we have made the predictions of monthly return of each month for every ETFs by using ANN and RNN models. We use these predictions as the proxies of views in Black-Litterman model and generate the portfolio. In contrast to most discussions of Black-Litterman model, which emphasize the use of relative views, our

application uses absolute views on monthly returns.

The portfolio investment period is set from January 2017 to October 2017, the portfolios with be rebalanced every month on the day right before the first day of each month. For every month we have 7 predicted monthly returns for 7 ETFs which are applied to Black-Litterman model. By using the Black-Litterman equation, we can obtain the views-adjusted return vector and covariance matrix. By using the views-adjusted return vector and covariance matrix, we generate the tangency portfolio, which the portfolio have highest expected sharpe ratio, based on ANN-derived views and RNN-derived views. We have set the portfolio constraints as maximum 100% for long purchase and maximum 100% for short selling. These constraints are set to avoid the extreme weights asset allocation. We have generated three tangency portfolios, which are based on Markowitz Mean-Variance, Black-Litterman model with ANN-views and Black-Litterman model with RNN-views. We implement the back test on these portfolios and study how the view predictions models affect the performance of the portfolios.

7 Black-Litterman Result

Table 10 summarizes the allocations and results of tangency portfolios based on Markowitz mean-variance, Black-Litterman with ANN-views and Black-Litterman with RNN-views. Figure 6 shows the performance of the portfolios through the investment period. Markowitz allocations to the 7 ETFs have resulted in a return of 6.4603% throughout the investment period. The returns of portfolio based on BLM with ANN views and BLM with RNN views are respectively 5.3316% and 9.3379%. The standard deviations of three portfolios are similar to each other and the level of risk is assumed acceptable. We can see that the portfolio based on BLM with RNN view is performed better than the portfolio based on BLM with ANN views. We hypothesize that the quality of Black-Litterman view prediction is a main factor that can have an influence on the portfolio, we have summarized the monthly returns predicted from AI model and the result is shown in Table 11.

Table 10: Markowitz and Black-Litterman Portfolio Allocations and Summary of Resulting Generated returns

	Markowitz Mean Var			BLM-ANN Views			BLM-RNN Views		
	Average	Max	Min	Average	Max	Min	Average	Max	Min
LQD	95.75%	100%	88.17%	92.82%	100%	84.28%	83%	100%	37.89%
IYY	-42.25%	-32.45%	-56.53%	-44.27%	-21.43%	-76.86%	-41.82%	-7.85%	-55.70%
VOT	25.12%	35.72%	13.20%	21.80%	44.81%	-10.38%	40.60%	100%	-1.80%
IWV	66.18%	67.23%	65.05%	77.59%	100%	56.14%	53.03%	77.51%	4.94%
IWP	6.25%	29.67%	-7.70%	3.45%	13.46%	-21.98%	14.66%	31.21%	-9.50%
IJK	48.95%	55.74%	43.36%	48.61%	79.04%	-4.23%	50.53%	79.68%	6.42%
IVV	-66.18%	-30.15%	-80.68%	-60.18%	-10.15%	-100%	-63.24%	-20.49%	-100%

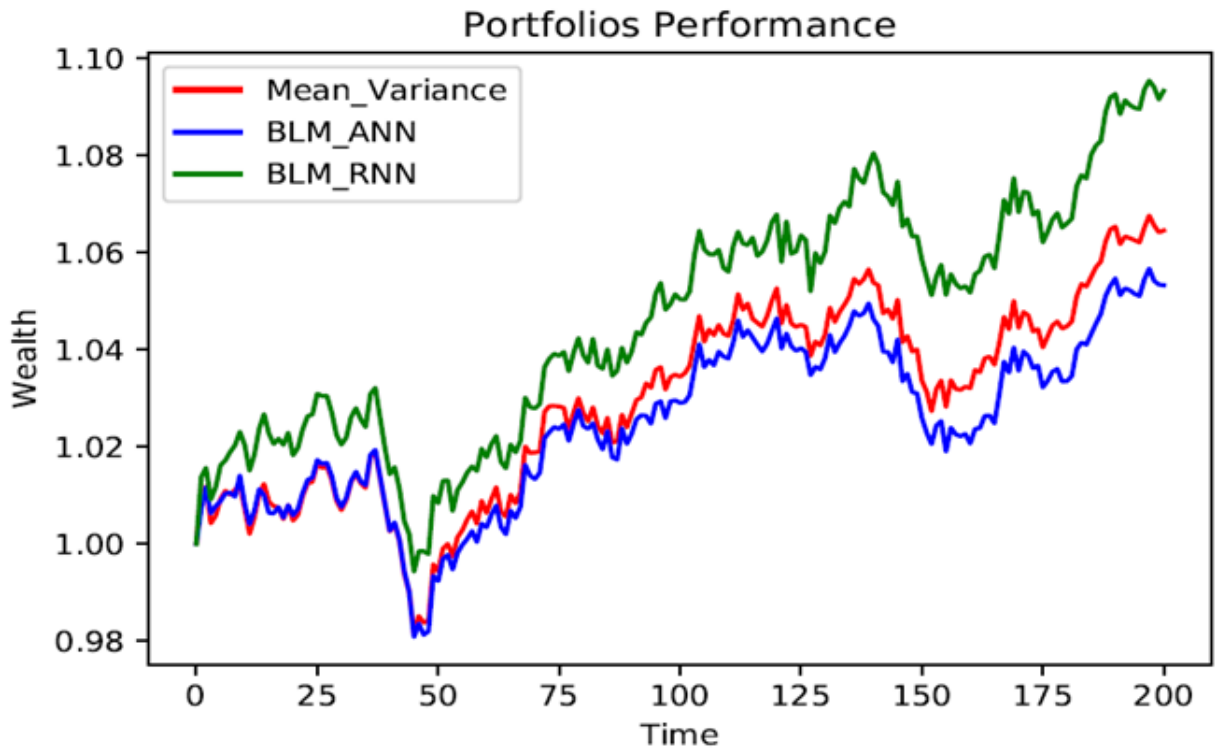

Figure 6: Performances of Markowitz and Black-Litterman Portfolios (JAN 2017 - OCT 2017)

Table 11: Summary of Portfolio Losses caused by wrong moving direction predictions

	ANN		RNN	
	Losses	Accuracy	Losses	Accuracy
JAN	-1.6892%	28.57%	-0.0635%	85.71%
FEB	-8.0541%	14.29%	-0.0042%	85.71%
MAR	-1.2478%	28.57%	-2.1124%	14.29%
APR	-0.4158%	71.43%	-	100%
MAY	-0.0243%	85.71%	-0.0053%	85.71%
JUN	-0.4925%	14.29%	-0.9338%	14.29%
JUL	-0.0002%	85.71%	-0.0003%	85.71%
AUG	-0.2061%	71.43%	-3.0229%	57.14%
SEP	-	100%	-0.0021%	85.71%
OCT	-	100%	-0.0024%	85.71%
Wrong Moving Direction Rate	40%		30%	
Average Loss	-1.21298%		-0.6138%	
MSE (Returns)	0.0487%		0.0147%	

From Table 11, we can see that the accuracies of predicted moving directions of ANN and RNN models are 40% and 30% respectively, and the mean-squared error of

the predicted monthly return with actual monthly returns of ANN and RNN model are 0.0487 and 0.0147 respectively. Here we can see that RNN model make a better prediction than ANN model, since RNN has higher accuracy and smaller mean-squared error of the view predictions. We have summarized the portfolio's losses caused by the ETFs which have contrary predicted moving direction with actual moving direction for every month and we find that the losses caused by contrary predictions is much higher in portfolio based on BLM with ANN views.

Furthermore, we find that the accuracy of predictions does have relationship with the portfolio performance. From Table 11, in ANN view prediction model, the predictions accuracies, which are lower than 30% are observed in January, February, March and June. We also can see that the losses of these months are the highest throughout the investment period. Similar phenomenon is observed in RNN model, March, June and August have the lowest accuracies of predictions throughout the investment period, and the losses of portfolio are the lowest throughout the period. Hence, we can conclude that the accuracies and forecast errors of views prediction are the major factors which will affect the performance of portfolio generated by Black-Litterman model. The view prediction which has higher accuracy and lower forecast error will help to generate a Black-Litterman portfolio with better performance.

8 Conclusion

We have three main conclusions as follows:

1. According to the portfolio performance, the RNN derived views BLM model works better than Markowitz Mean-Variance. Both RNN derived views BLM model and Markowitz Mean-Variance work better than ANN derived views BLM model.
2. For the views prediction, in our project, which is the log return prediction, RNN algorithm works much better than ANN algorithm. This is probably because RNN algorithm works better in time series prediction and ANN algorithm works better in independent parameters prediction.
3. For the portfolio optimization, there are some important factors that have effect on the performance of BLM portfolio optimization. The accuracy of predictions of moving directions plays a great role in predicting optimized portfolio. Another important factor is the forecasting error between actual log returns and predicted log returns.

RNN algorithm in our project only takes the historical log returns into consideration, there may be potential better or multiple features, which can be included in our prediction. The views we have predicted so far are absolute views. But for the views in BLM, some paper indicate the relative views may perform better than absolute views. Therefore, in the future research, we may think of implementing relative views to see the performance of predictions given by AI models.

We attach the our code in Appendix A, B and C.

Group Contribution Form

Table 12: Group Contribution Form

Member	Contributions
Disha An	Literature review, search and study research papers Artificial neural network view prediction model construction Study and improving the ANN, RNN view prediction model ANN-derived view summarization Visualize and analyze the view prediction result Portfolio optimization model construction PPT preparation Write the final report
Jian Shun Tan	Literature review, search and study research papers Recurrent neural network view prediction model construction RNN-derived view summarization Black-Litterman model construction Portfolio back-test model construction Portfolio back-test and visualization PPT preparation Write the final report

A ANN View Prediction Model

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense

def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(units = 35, kernel_initializer = "uniform", input_dim = 48))
    classifier.add(Dense(units = 35, kernel_initializer = "uniform"))
    classifier.add(Dense(units = 35, kernel_initializer = "uniform"))
    classifier.add(Dense(units = 35, kernel_initializer = "uniform"))
    classifier.add(Dense(units = 35, kernel_initializer = "uniform"))
    classifier.add(Dense(units = 1, kernel_initializer = "uniform"))
    classifier.compile(optimizer = 'adam', loss = 'mean_squared_error')
    return classifier

train_start_date = '2010-01-01'
train_end_date = '2016-12-31'
test_start_date = '2017-01-01'
test_end_date = '2017-10-31'

# ETFs list : IJK, IVV, IWP, IWV, IYY, LQD, VOT
## LQD
all_data = pd.read_csv('LQD.csv') # This is the file generated from ANN's inputs data preprocessing sode
mask = (all_data['Date'] > train_start_date) & (all_data['Date'] <= train_end_date)
train_dataset = all_data.loc[mask]
X_train = train_dataset.iloc[:, 1:49].values
y_train = []
l = len(train_dataset)
for i in range(l-19):
    y_train.append(sum(train_dataset.iloc[i:20+i, 20:21].values))
y_train = np.array(y_train)
X_train = X_train[:l-19,:]

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)

# Training the data
classifier = build_classifier()
classifier.fit(X_train, y_train, batch_size = 49, epochs = 500)

```

```
# Predicting the result
mask = (all_data['Date'] > test_start_date) & (all_data['Date'] <=
    test_end_date)
test_dataset = all_data.loc[mask]
X_test = test_dataset.iloc[:, 1:49].values
y_test = []
l = len(test_dataset)
for i in range(l-19):
    y_test.append(sum(test_dataset.iloc[i:20+i, 20:21].values))
y_test = np.array(y_test)
X_test = X_test[:l-19,:]
sc = StandardScaler()
X_test = sc.fit_transform(X_test)
y_pred = classifier.predict(X_test)
```

B RNN View Prediction Model

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from sklearn.preprocessing import StandardScaler

train_start = '2010-01-01'
train_end = '2016-12-31'
test_start = '2016-07-13'
test_end = '2017-10-31'

def build_regressor():
    regressor = Sequential()
    regressor.add(LSTM(units = 50, return_sequences = True,
    input_shape = (X_train.shape[1], 1)))
    regressor.add(Dropout(0.2))
    regressor.add(LSTM(units = 50, return_sequences = True))
    regressor.add(Dropout(0.2))
    regressor.add(LSTM(units = 50, return_sequences = True))
    regressor.add(Dropout(0.2))
    regressor.add(LSTM(units = 50))
    regressor.add(Dropout(0.2))
    regressor.add(Dense(units = 1))
    regressor.compile(optimizer = 'adam', loss = 'mean_squared_error
    ')
    return regressor

# ETFs list : IJK, IVV, IWP, IWV, IYY, LQD, VOT
### LQD ###
data = pd.read_csv('LQD.csv') # Preprocessed daily log-returns data
    file
mask = (data['Date'] >= train_start) & (data['Date'] <= train_end)
```

```
train_dataset = data.loc[mask]
train_dataset = train_dataset.iloc[:,1:2].values
sc = StandardScaler()
scaled_train = sc.fit_transform(train_dataset)
X_train = []
y_train = []
for i in range(len(scaled_train)-120):
    X_train.append(scaled_train[i:i+120,0])
    y_train.append(sum(train_dataset[i+120:i+140]))
y_train = sc.fit_transform(y_train)
X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1],
    1))

regressor = build_regressor()
regressor.fit(X_train, y_train, batch_size = 32, epochs = 100)

mask = (data['Date'] >= test_start) & (data['Date'] <= test_end)
test_dataset = data.loc[mask]
test_dataset = test_dataset.iloc[:,1:2].values
scaled_test = sc.fit_transform(test_dataset)
X_test = []
y_test = []
for i in range(len(scaled_test)-120):
    X_test.append(scaled_test[i:i+120,0])
    y_test.append(sum(test_dataset[i+120:i+140]))
X_test, y_test = np.array(X_test), np.array(y_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

y_pred = regressor.predict(X_test)
y_pred = sc.inverse_transform(y_pred)
```

C Black-Litterman Portfolio Optimization

```
import pandas as pd
from numpy import zeros
import numpy as np
import portfolioopt as pfopt

def view_link(etf_list, views):
    n = len(etf_list) # Number of ETFs
    m = len(views) # Number of Views
    # View Matrix
    view_m = [views[i][2] for i in views]
    view_m = pd.DataFrame({'Views':view_m})
    # Link Matrix
    link_m = zeros([m,n]) # Create Link Matrix Filled with 0
    i = 0 # View Record
    for each in views.values():
        num_etf = len(each[0])
        for j in range(num_etf):
            link_m[i][etf_list.index(each[0][j])] = each[1][j]
```

```
            i += 1
    link_m = pd.DataFrame(link_m, index = list(views.keys()),
    columns = etf_list)
    return(view_m, link_m)

## preprocess daily returns of previous month ##
dailyreturn = pd.read_csv('returns_17_10.csv')
data = dailyreturn
d = {}
for i in data.columns.values:
    dval = data[i].tolist()
    d[i] = dval
# d is the dictionary we can use to put into the blm model
stk_list = list(d.keys())
data = list(d.values())
n = len(data[0])
cov_m = pd.DataFrame(np.cov(data))
rtn_m = pd.Series([np.mean(each) for each in data])

#tangency_portfolio(pandas.DataFrame, pandas.Series)
w_neutral = pfopt.tangency_portfolio(cov_m, rtn_m, allow_short =
    True)
w_neutral = pd.DataFrame([round(each,4) for each in w_neutral],
                         index = stk_list, columns = ['Weights'])

### The asolute views obtained from view prediction AI-model
###############
view = {}
view['view1'] = (['LQD'],[1],0.003507782)
view['view2'] = (['IYY'],[1],0.000190048)
view['view3'] = (['VOT'],[1],0.007558567)
view['view4'] = (['IWV'],[1],0.001878394)
view['view5'] = (['IWP'],[1],0.004234864)
view['view6'] = (['IJK'],[1],0.019953683)
view['view7'] = (['IVV'],[1],0.01048204)
###############

p,q = view_link(stk_list, view) # p: View Matrix   q: Link Matrix

delta = 2.65 # Risk Aversion
tau = 0.01 # Scaling Factor
q = q.as_matrix()
p = list(p['Views'])
omega = tau * (q.dot(cov_m).dot(q.transpose()))
omega = np.diag(np.diag(omega, k = 0))

adj_rtn = rtn_m+tau*cov_m.dot(q.transpose()).dot(
    np.linalg.inv(omega+tau*(q.dot(cov_m).dot(
        q.transpose())))).dot(p-q.dot(rtn_m))

# new covariance matrix for BLM
tauV = tau*cov_m
mid_eq = np.linalg.inv(np.dot(np.dot(q,tauV),q.T) + omega)
adj_cov = cov_m + tauV - tauV.dot(q.T).dot(mid_eq).dot(q).dot(tauV)
```

```
71
72 w_expressing = pfopt.tangency_portfolio(adj_cov, adj_rtn,
       allow_short = True)
73 ##w_expressing = np.linalg.inv(delta*cov_m).dot(rtn_m)
74 w_expressing = pd.DataFrame([round(each,4) for each in w_expressing
       ],
75                             index = stk_list, columns = ['Weights'])
76 print("Weights with Expressing Views:")
77 print(w_expressing)
```

References

Beach, S. L. and A. G. Orlov (2007). An application of the black–litterman model with egarch-m-derived views for international portfolio management. *Financial Markets and Portfolio Management 21*(2), 147–166.

Bengio, Y., P. Simard, and P. Frasconi (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks 5*(2), 157–166.

Bevan, A. and K. Winkelmann (1998). Using the black-litterman global asset allocation model: three years of practical experience. *Fixed Income Research*, 1–19.

Black, F. and R. Litterman (1992). Global portfolio optimization. *Financial analysts journal 48*(5), 28–43.

Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Drobetz, W. (2001). How to avoid the pitfalls in portfolio optimization? putting the black-litterman approach at work. *Financial Markets and Portfolio Management 15*(1), 59–75.

He, G. and R. Litterman (1999). The intuition behind black-litterman model portfolios.

Hochreiter, S. and J. Schmidhuber (1997). Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pp. 473–479.

Markowitz, H. (1952). Portfolio selection. *The journal of finance 7*(1), 77–91.

Mikolov, T., M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur (2010). Recurrent neural network based language model. In *Interspeech*, Volume 2, pp. 3.

Quah, T.-S. and B. Srinivasan (1999). Improving returns on stock investment through neural network selection. *Expert Systems with Applications 17*(4), 295–301.

Satchell, S. and A. Scowcroft (2000). A demystification of the black–litterman model: Managing quantitative and traditional portfolio construction. *Journal of Asset Management 1*(2), 138–150.

Trippi, R. R. and E. Turban (1992). *Neural networks in finance and investing: Using artificial intelligence to improve real world performance*. McGraw-Hill, Inc.

Zhou, G. (2008). Beyond black–litterman: Letting the data speak. *The Journal of Portfolio Management 36*(1), 36–45.