Research on Classification-Based Financial Market Prediction Algorithms Using Deep Learning

Wang Yimeng[1], Chen Miao[2], Yu Junqi[3]

[1]College of the Arts, Guangxi University, Nanning, Guangxi, 530004

[2]School of Economics and Management, Northeast Agricultural University, Harbin, Heilongjiang, 150038

[3]Asia Australia Business College, Liaoning University, Shenyang, Liaoning, 110136

Keywords: Financial Market, Prediction Algorithms, Deep Learning

Abstract: Deep neural networks (DNNs) are a powerful artificial neural network (ANN) using multiple hidden layers. In recent years, it has gained considerable attention in the fields of speech conversion and image recognition, because of their superior the predictive properties of the algorithm include robustness to overfitting. However, their application in algorithmic trading has not been studied before, partly because of their computational complexity. Application. Specifically, we describe the configuration and training methods, and then demonstrate their application of a reverse test of a simple trading strategy at 43-minute intervals on 43 different commodities and foreign exchange futures. All results are generated using the C++ implementation on the Intel Xeon Phi coprocessor (11.4 times faster than the serial version) and the Python policy back testing environment, both of which are open source code written by the author.

1. Introduction

Many challenges faced by financial econometrics methods include non-stationary, nonlinear or noise of time series. Although artificial neural network (ANN) has been applied in time series method, their over fitting tendency, convergence problem and implementation difficulty have aroused people's attention. Moreover, they have deviated from the foundation of financial econometrics. Alienated the financial econometric research community and financial practitioners [1].

However, electronic trading companies employ computer scientists and mathematicians, who not only regard artificial neural network as black box, but also regard it as a nonparametric modeling method based on minimization of entropy function. The progress of modern computer architecture has promoted the recent revival of this method. Deep neural network (DNN) is an artificial neural network with multiple hidden units between input layer and output layer. Because of its successful application in image classification and speech recognition, they have been popularized in the artificial intelligence community. This area is known as "deep learning" [2].

In this article, we will use DNN to partially solve some of the historical defects of ANN. In particular, we have modeled the complex nonlinear relationship between independent variables and dependent variables, reducing the trend of over fitting. In order to achieve this, we will use the advanced technology of low cost multi-core accelerator platform to train and adjust the parameters of our model [3].

For financial forecasting, especially in multivariate prediction analysis, the feed forward topology has attracted more attention and will become the method used in this paper. Compared with other training models, back propagation and gradient descent are the preferred methods for training these structures, because they are easy to implement and converge easily to better local optimum values. However, these methods may be computationally expensive. Especially when used in training DNN.

DNN has many training parameters that need to be considered, such as size (number of layers and number of cells per layer), learning rate and initial weight. Because of the cost of time and computational resources, it is not feasible to search the optimal parameters in parameter space. We

will use mini batching (one time to calculate the gradient of several training samples instead of a single sample) as a common accelerated calculation method. The back propagation algorithm is represented as a form of fast performance that can be implemented on the Intel Xeon Phi coprocessor. Shekhar and Amin describe the general hardware optimization implementation of the back propagation algorithm, but our method is tailored for the Intel Xeon Phi coprocessor.

The main contribution of this paper is to describe the application of deep neural network in financial time series data, so as to classify the motion direction of financial market. Traditionally, researchers have repeatedly tested several signals, and trained a level based method for each instrument, such as vector autoregression. Recently, some scholars have provided evidence that the classification method is superior to the level based method in predicting stock trend and maximizing transaction returns.

In the following part, we will introduce the back propagation learning algorithm, and use the minimum batch to represent the most intensive equation as matrix. Once it is expressed in matrix form, you can use the hardware optimized numerical linear algebra routines to implement the efficient mapping of the algorithm to the Intel Xeon Phi coprocessor. The third section introduces the preparation for data training for DNN. The fourth section describes the implementation of DNN. Then, the fifth section gives the results of measuring the performance of DNN. Finally, in the sixth quarter, we use the forward walking method to demonstrate the application of DNNs in reverse testing. It also provides the performance results of a simple buy hold sell strategy.

2. Deep Neural Network Classifier

Let's start with mathematics matriculation. Set D to represent M features and historical data sets of N observations. We have drawn N subsets of training $D_{Train} \subset D$ and N_{test} observed by training subsets.

The first n(eigenvector) is represented as $x_n \in D_{Train}$. In ANN, every element of the vector becomes a node in the input layer. As shown in the figure below, when each observation has 7 input variables (characteristics), every node in the fully connected feed forward network is connected to each node in the next layer. Although it is not shown in the graph. However, the weight W^{Lij} is associated with each edge between the nodes in the previous layer and the j node in the current layer l.

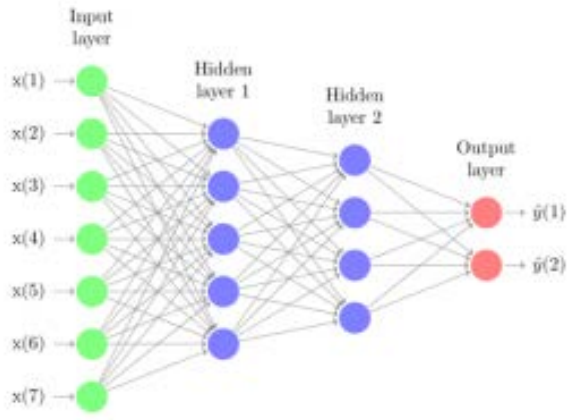

Figure1 Neural Network

Figure 1 an illustrative example of a feed forward neural network with two hidden layers, seven features and two output states. Deep learning network classifier usually has more layers, uses a large number of features and several output states or classes. The goal of learning is to find the weight of each edge that minimizes the out of sample error.

In order to find the best weight w= {w $^{(L)}$} l=1 to L between nodes in the fully connected feed-forward network with l layer, we seek the minimized form of cross entropy function:

$$E(\mathbf{w}) = -\sum_{n=1}^{N_{\text{tem}}} e_n(\mathbf{w}), \quad e_n(\mathbf{w}) := \sum_{k=1}^{K} y_{kn} \ln(\hat{y}_{kn}) \tag{1}$$

For clarity, we removed the subscript n. binary target y and output variables. $\hat{y}$ For each symbol, there are 1 to K codes, where $y_k\{0,1\}$, so that each output state associated with symbols can be interpreted as a probability weighting. In order to ensure the analytic gradient function under cross entropy error measurement, and to ensure the probability of each state is 1, the output layer is activated by the following form of softmax function.

$$\hat{y}_k := \phi_{\text{coft max}}\left(\mathbf{s}^{(L)}\right) = \frac{\exp\left(s_k^{(L)}\right)}{\sum_{j=1}^{k_s}\exp\left(s_j^{(L)}\right)} \quad (2)$$

For the fully connected feed forward network, s $^{(L)j}$ is the weighted sum of the output of the previous layer L-1, which is derived from the node j connected to the layer L.

$$s_j^{(l)} = \sum_{i=1}^{n_{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} + \text{bias}_j^{(l)} \quad (3)$$

For the fully connected feed forward network, s $^{(L)j}$ is the weighted sum of the output of the previous layer L-1, which is derived from the node j connected to the layer L.

$$\frac{\partial e(\mathbf{w})}{\partial s_k^{(L)}} = \hat{y}_k - y_k$$

The recursive relation of back propagation using conjugate gradient is:

$$\delta_i^{(l-1)} = \sum_{j=1}^{n_{i(l-1)}} \delta_j^{(l)} w_{ij}^{(l)} \sigma\left(s_i^{(l-1)}\right)\left(1-\sigma\left(s_i^{(l-1)}\right)\right)$$

So, in conclusion, given any observation as input, the recurrent feed forward network can be used to predict the probability of each symbol's output state (or category) recursively by formula 3. Now describe how the network is trained.

Stochastic gradient descent

We now review the back propagation learning algorithm based on the stochastic gradient descent (SGD) algorithm. Although SGD is only first-order, SGD can be used as an optimization method for DNNs because of the highly non convex form of utility functions.

After random sampling of the observed I, the SGD algorithm updates the first level parameter vector w $^{(l)}$ with the following parameters

To illustrate the results of level L:

$$\mathbf{w}^{(l)} = \mathbf{w}^{(l)} - \gamma\nabla E_i\left(\mathbf{w}^{(l)}\right)$$

Gamma is the learning rate. Algorithm 1 gives a high-level description of the sequential version of the SGD algorithm. Note that for the sake of simple description, we avoid some nuances of implementation.

Algorithm 1 STOCHASTIC GRADIENT DESCENT

```
w ← r, r_i ∈ N(μ, σ), ∀i
E ← 0
for i = 0 to n − 1 do
    E ← E + E_i(w)
end for
while E ≥ τ do
    for t = 0 to n − 1 do
        i ← sample with replacement in [0, n − 1]
        w ← w − γ∇E_i(w)
    end for
    E ← 0
    for i = 0 to n − 1 do
        E ← E + E_i(w)
    end for
end while
```

Algorithm1 Stochastic gradient descent

3. Data

Our historical data set contains the 5 minute median price for the 43 commodity exchanges and the foreign exchange futures between March 31, 1991 and September 30, 2014. We used the data of the last 15 years, because the poor liquidity of some symbols in the previous period resulted in no price fluctuation in the long period of 5 minutes. By normalizing each characteristic by subtracting the mean and dividing the standard deviation, the training set was set at 25.000 continuous observations are made, and the test set consists of the next 12500 observations. As mentioned in section sixth, these collections start from the liquid observation period and roll forward 10 times with 1000 observations, until the last 37500 observations from March 31, 2005 to the end of the data set.

The whole training dataset is composed of a set of characteristic training sets for each symbol. The training set of each symbol is composed of the price difference and engineering characteristics. It includes the lag price difference from 1 to 100, the average price of the window size from 5 to 100, and the pairwise correlation between the returns and returns of all other symbols. The whole training set contains 9895 characteristics. The motivation of containing these functions in the model is to capture the cooperative movement between memory and symbols in historical data.

4. Experimental Scheme

Our network architecture contains five learning fully connected layers. The first layer of the four hidden layers contains 1000 neurons, and each layer is reduced by 100. The last layer contains 129 output neurons. Each symbol of the -43 futures contract has three values. The result containing a large number of features and multiple hidden layers is a total of 12174500 weights.

The weights are initialized by using the Intel MKL VSL random number generator using the routines of Mersenne Twistor (MT19937). The inverse Gauss cumulative distribution function with zero mean and standard deviation 0.01 is used to transform the uniform random number to generate Gauss random numbers. We use constant 1 to initialize the neuron bias in the hidden layer.

We use the same learning rate for all layers. The learning rate is adjusted according to heuristics. The heuristic is described in the following algorithm 2, and is similar to the method adopted by Krizhevsky et al. We use cross entropy instead of validation error. We scanned the parameter space of learning rate from [0.1, 1] to 0.1 increments. If the cross entropy between epoch generations did not decrease, we further divide the learning rate gamma by 2. In the algorithm 2, and the subset of training sets for each epoch is defined.

Algorithm 2 DEEP LEARNING METHODOLOGY

for $\gamma := 0.1, 0.2, \ldots, 1$ **do**
  $w_{i,j}^{(l)} \leftarrow r,\ r \in \mathcal{N}(\mu, \sigma),\ \forall i, j, l$ ▷ Initialize all weights
  for $e = 1, \ldots, N_e$ **do** ▷ Iterate over epochs
    Generate $\mathcal{D}_e$
    for $m = 1, \ldots, M$ **do** ▷ Iterate over mini-batches
      Generate $\mathcal{D}_m$
      for $l = 2, \ldots, L$ **do**
        Compute all $x_j^{(l)}$ ▷ Feed-Forward network construction
      end for
      for $l = L, \ldots, 2$ **do**
        Compute all $\delta_j^{(l)} := \nabla_{s_j^{(l)}} E$ ▷ Backpropagation
        Update the weights: $\mathbf{w}^{(l)} \leftarrow \mathbf{w}^{(l)} - \gamma X^{(l-1)} \left(\delta^{(l)}\right)^T$
      end for
    end for
  end for
  If cross_entropy(e) $\leq$ cross_entropy(e-1) then $\gamma \leftarrow \gamma/2$
end for
Return final weights $w_{i,j}^{(l)}$

Algorithm 2 Deep learning method

As mentioned earlier, the small batch formula of the algorithm contributes to efficient parallel implementation. Dixon et al. Described its details and timing. Taking into account the time of error

measurement on the test set, the total time to train DNN on Intel Xeon Phi using the above data is about 8 hours, so retraining is needed every day. Training can be run as an overnight batch operation. This is 11.4 times faster than the serial version of the running algorithm.

5. Experimental Results

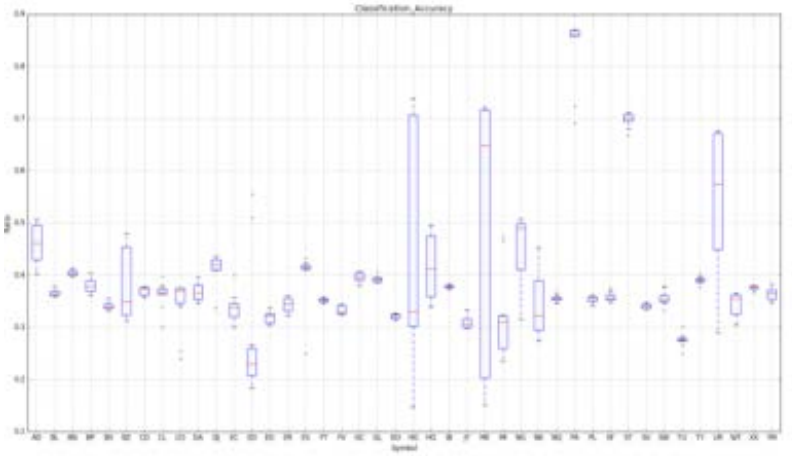

Figure2 classification accuracy of DNN applied to 43 CME commodities and forex futures.

Figure 2 this graph shows the classification accuracy of DNN applied to 43 kinds of CME commodities and foreign currency futures. Each symbol is represented by the vertical bar of the box and the beard. The box represents the area between the upper and lower four quantiles of the sample distribution of the classification accuracy. The median of the sample distribution of the classification accuracy is expressed by a red horizontal line.

This section introduces the DNN reverse testing of simple algorithm trading strategies. The purpose is to link the accuracy of classification with the measurement of strategic performance rather than providing detailed exploration of trading strategies or their performance. We calculate the classification accuracy of the mobile test window for each 130 days. Repeat this process to give a set of ten classification errors. Figure2 shows the block diagram of the DNN classification accuracy of all 43 CME commodities and foreign exchange futures. Each symbol is represented by a box and a vertical bar. The box represents the area between the upper and lower four quantiles of the sample distribution of 8 classification accuracies.The median is represented by a red horizontal line. The following Figure3 shows the average classification accuracy distribution of 10 DNN samples of 43 CME commodities and forex futures. The accuracy is about 0.35 greater than that of random selection.

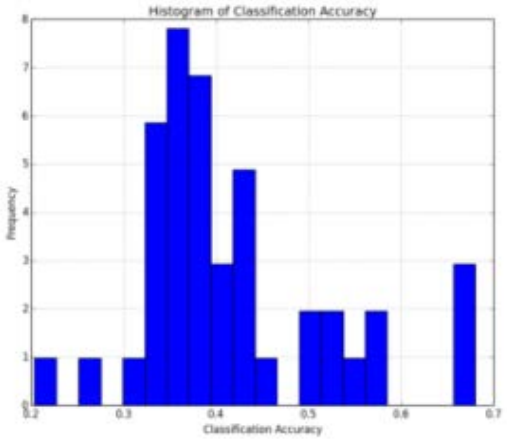

Figure 3 Average classification accuracy distribution of DNN applied to 43 CME commodities and foreign exchange

Table1 shows the top five mean samples with the highest average sample size in ten forward walking experiments. It also shows the F1 score ("harmonic average"). It is considered a more robust performance measure, because its sensitivity to class imbalance is lower than that of classification accuracy. It also provides the average sample classification accuracy and the average and standard deviation of F1 scores for 43 futures.

Table 1 In the ten forward walking experiments, the first five instruments with the highest classification rate were the highest.

Symbol	Futures	Classification Accuracy	F1-score
HG	Copper	0.68	0.59
ST	Transco Zone 6 Natural Gas (Platts Gas Daily) Swing	0.67	0.54
ME	Gulf Coast Jet (Platts) Up-Down	0.67	0.54
TU	Gasoil 0.1 Cargoes CIF NWE (Platts) vs. Low Sulphur Gasoil	0.56	0.52
MI	Michigan Hub 5 MW Off-Peak Calendar-Month Day-Ahead Swap	0.55	0.5
mean	-	0.42	0.37
std	-	0.11	0.1

Note that in the ten experiments, the average performance of the five tools with the worst performance is no better than that of white noise, or even 10 times that of white noise.

6. Strategy Backdating Test

So far, the prediction characteristics of deep neural networks have been considered. Using the historical data of commodity futures every 5 minutes from March 31, 1991 to September 30, 2014, this section describes the application of the forward walking optimization method in the simple return strategy.

Following the forward walking optimization method described in the previous literature, we choose 25000 initial windows of 5 minute observation period or about 260 days (slightly more than a year) to train all models using all symbol data and their designed time series. The range of scanning learning rate is to find out the highest classification rate of the model out of sample (persisting) set of the best sample out prediction rate.The collection consists of 12500 continuous and recent observations.

Using the optimized model, the expected profit and loss of the trading strategy is evaluated in the sample period consisting of 12500 continuous 5 minute observation periods or about 130 days. Even if all symbols are trained together with a DNN model, cumulative p&l is calculated independently for each symbol. As shown in Figure4, the training window is moved forward by 1000 observation periods. We repeat the sample error analysis and policy performance measurement in 10 windows.

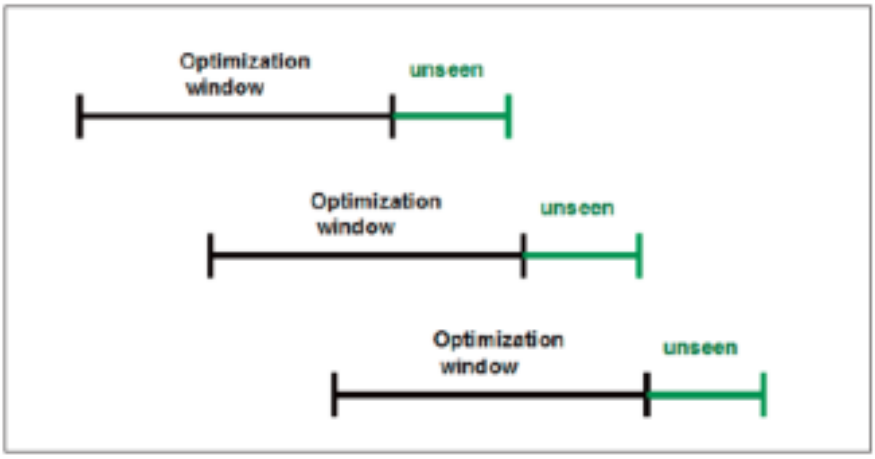

Figure 4 An example of the forward stepping optimization method for policy re examination is given.

6.1 Example of Trading Strategy

In order to demonstrate the application of DNNs in algorithmic trading, a simple buy hold sell intraday trading strategy is selected according to the possible increase, neutral or fall of the price of the tool in the next time interval. For simplicity, the strategy only orders a number of market orders. If the label is 1, the strategy closes the short position and takes the multi position; if the label is zero, then the position is maintained. If the label is -1, the multi head position is closed and the short position is taken. The following simplified assumptions are made when calculating the accumulated unrealized gains and losses.

(1) The account was opened for $100 thousand.

(2) Sufficient cash is available to maintain brokerage account margin through profit or other means.

(3) There is no limit to the shortest or longest holding period, and the position can be spent overnight;

(4) Transaction costs are ignored.

(5) No operational risk measures have been deployed, such as stop loss orders.

(6) The market always has enough liquidity, and the market orders will be filled immediately at the intermediate price of 5 minutes, so the slip effect is ignored.

(7) The 1 batch of market orders every 5 minutes has no significant impact on the market, so the forecast does not take account of the constraints on the execution of orders.

These assumptions, especially those about transaction execution and no real time simulation in the environment of back test, are of course not enough to prove the Alfa generating ability of DNN based strategies, but they can serve as the starting point for the commercial application of this research. Figure 5 shows that the strategy is applied to the sample distribution of the time average daily returns of commodity futures and foreign exchange futures in the first 43 months of the 43 CME, respectively.

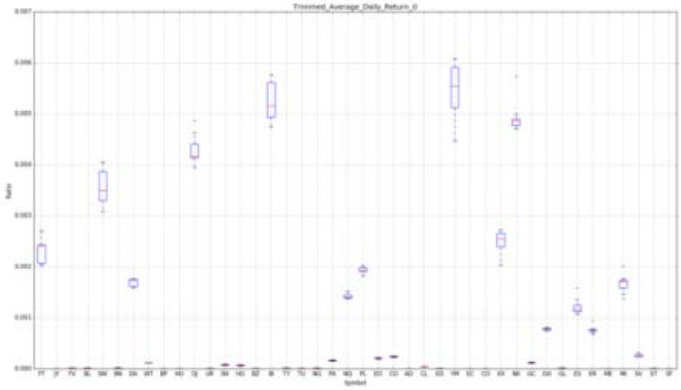

Figure 5 Box diagram of sample distribution for average daily income within 130 days

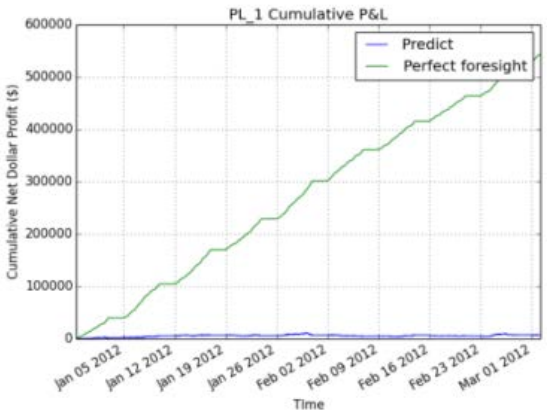

Figure 6 Accumulative unrealized net dollar profit of a simple strategy

In order to quantify the impact of information loss, the profit under the full prediction information is expressed as "perfect foresight" (green line), and the profit predicted by DNN is expressed as "forecast" (blue line). This chart shows the 130 day trading period of PL futures in recent months.

Figure 6 compares the cumulative unrealized net profit of the strategy in the case of perfect prediction information ("perfect foresight") and the use of DNN prediction ("prediction"). This chart shows the 130 day trading period of PL futures in recent months.

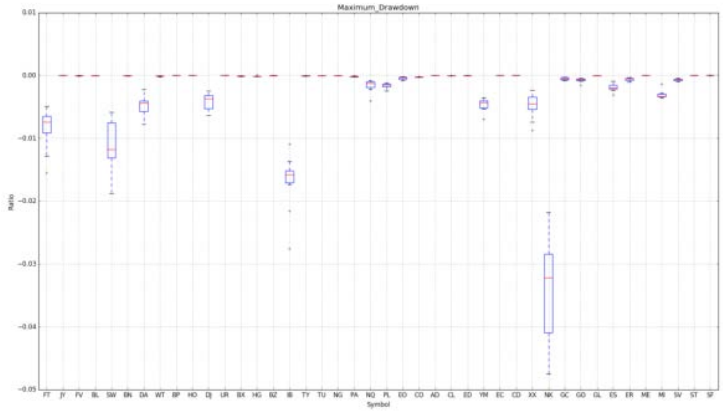

Figure7 A block diagram of the maximum descending value of a simple strategy.

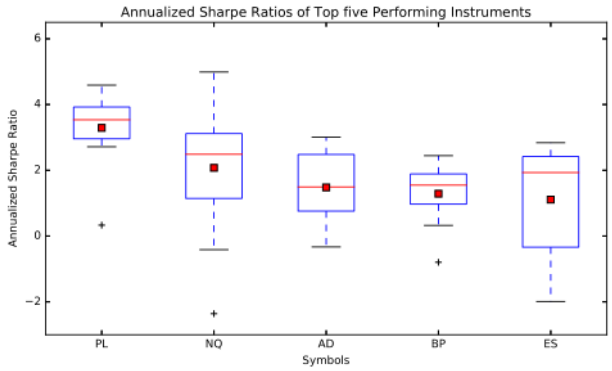

Figure 8 SHARP ratio distribution box diagram

Only the five best futures contracts are considered. Simple trading strategies mentioned above. Keywords: pl: platinum, nq: e- Mini Nasdaq 100 futures, ad: Australian dollar, bp: sterling, es: e-Mini S & P 500 futures.

Figure7 shows a block diagram of the maximum descending value of a simple strategy. The strategy applies ten forward walking experiments for each symbol. Figure8 shows the range of the annual five SHARP futures contracts measured in each mobile period of 12500 observations. Table2 also supplements this figure, which shows that in the ten forward walking experiments, the first five instruments with the highest average sample size of the annualized SHARP ratio indicate the standard deviation of the ten experiments. The paper also gives the sample mean and standard deviation of the capacity ratio (n=130) under the assumption of normal return.

Table 2 The ten best tools to optimize the average age and shape ratio are the top five tools.

Symbol	Futures	Annualized Sharpe Ratio	Capability Ratio
PL	Platinum	3.29 (1.12)	12.51 (4.27)
NQ	E-mini NASDAQ 100 Futures	2.07 (2.11)	7.89 (8.03)
AD	Australian Dollar	1.48 (1.09)	5.63 (4.13)
BP	British Pound	1.29 (0.90)	4.90 (3.44)
ES	E-mini S&P 500 Futures	1.11 (1.69)	4.22 (6.42)

The values in brackets represent the standard deviations of the ten experiments. Under the assumption that the yield is a normal distribution, the mean and standard deviation of the capacity ratio are also given.

Table3 Initial margin, maintenance margin and contract size specified by CME

Symbol	initial margin	maint. margin	contract size
PL	2090	1900	50
NQ	5280	4800	50
AD	1980	1800	100000
BP	2035	1850	62500
ES	5225	4750	50

Table 3 this table lists the initial margin, maintenance margin and contract size designated by the Chicago Mercantile Exchange to calculate the accumulated profit and loss and the strategic performance of the five futures positions with the best performance.

Table 4 shows the correlation between the strategic daily gains of the five most mobile tools in the 43 CME futures and their related ETF benchmarks. These values represent the aggregate statistical information of the correlation in ten experiments. When the average of ten experiments is over, the correlation between the strategic return and the benchmark return is very weak, and the absolute value of correlation is below 0.5 in any given experiment.

Table 4 The correlation between the daily strategic return and the relevant ETF benchmark of the 43 most mobile CME instruments in the five futures markets

Symbol	Benchmark ETF	Correlation			
		Mean	Std. Dev.	Max	Min
NQ	PowerShares QQQ ETF (QQQ)	0.013	0.167	0.237	-0.282
DJ	SPDR Dow Jones Industrial Average ETF (DIA)	0.008	0.194	0.444	-0.257
ES	SPDR S&P 500 ETF (SPY)	-0.111	0.110	0.057	-0.269
YM	SPDR Dow Jones Industrial Average ETF (DIA)	-0.141	0.146	0.142	-0.428
EC	CurrencyShares Euro ETF (FXE)	-0.135	0.108	0.154	-0.229

These values represent the aggregate statistics of correlation in ten experiments. The key is: NQ: E-mini Nasdaq 100 futures, DJ: DJIA (10 US dollars) futures, ES: E-mini Standard & Poor's 500 futures, YM: E-mini Dow Jones index (5 U.S. dollars) futures, EC: Euro euro futures.

7. Conclusion

DNNs's layout and training. Deep neural network (DNNs) is a powerful artificial neural network (ANN) using multiple hidden layers. In this article, we describe its implementation. We observed the historical data set of the 5 minute median price of listed futures prices and other lags and filters for several Chicago commodity exchanges (CME). If training is done across several markets on the tag data, DNN as a classifier has substantial predictive power. We further demonstrate the application of DNNs in reverse testing of a simple trading strategy. The accuracy of prediction and its relationship with strategic profitability are proved. All the results in this article are generated by C++ in the Intel Xeon Phi coprocessor (11.4 times faster than the serial version) and the Python strategy back test environment. These two environments are the source code written by the author.

References

[1] J. Chen, J. F. Diaz, and Y. F. Huang. High technology ETF forecasting: Application of Grey Relational Analysis and Artificial Neural Networks. Frontiers in Finance and Economics, 10(2): 129-155, 2013.

[2] M. Dixon, D. Klabjan, and J. H. Bang. Implementing deep neural networks for financial market prediction on the Intel Xeon Phi. In Proceedings of the 8th Workshop on High Performance Computational Finance, WHPCF'15, pages 6: 1-6: 6, New York, NY, USA, 2015.

[3] J. Faraway and C. Chatfield. Time series forecasting with neural networks: A comparative study using the airline data. Journal of the Royal Statistical Society: Series C (Applied Statistics), 47(2): 2310-250, 1998.