DEVELOPMENT OF A NOVEL DOUBLE NEURAL NETWORK AND ITS APPLICATIONS

A Dissertation Presented

by

Hao Chen

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

December 2015

Stony Brook University

The Graduate School

Hao Chen

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Song Wu – Dissertation Advisor
Assistant Professor, Department of Applied Mathematics and Statistics

Wei Zhu – Chairperson of Defense
Professor, Deputy Chair, Department of Applied Mathematics and Statistics

Pei Fen Kuan
Assistant Professor, Department of Applied Mathematics and Statistics

Keli Xiao – Outside Member
Assistant Professor, College of Business

The dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

ABSTRACT OF THE DISSERTATION

Development of a novel double neural network and its applications

by

Hao Chen

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2015

Artificial neural network model is a powerful method that has been widely applied in many different areas. It is essentially a nonlinear statistical model, empirically proved with good prediction accuracy, and has been applied in both regression and classification problems. One challenge in applying artificial neural network models is constructing proper structure adaptive to specific problems. This thesis work is to introduce a novel, double-layered feed-forward neural network (DNN) model with special link patterns. Its applications to genome-wide association studies and stock price prediction in high frequency time scale have been explored.

Detecting gene-gene interactions in traditional Genome-wide associate studies (GWAS) is mostly at the SNP level, called SNP-SNP interactions, which ignores the existence of large amount of correlations embedded among nearby SNPs. Popular existing methods with this mechanism, such as multifactor-dimensionality reduction (MDR) and random forests, would usually suffer from redundant interaction tests, due to the correlations between SNPs, and subsequently from less powers. With our new DNN model, we can take advantage of the correlations between SNPs and perform interaction test at the level of SNP blocks. Extensive simulation studies have been conducted to compare our new method with Random Forests. And our simulation results suggest that the DNN model can have higher power than Random Forests in detecting the existence of causal SNPs – no matter the effect is interactive or marginal.

We also have applied the DNN model to financial markets, forecasting changes of stock prices in high frequency. One advantage of our DNN model is that it utilizes correlation information between different stocks, a pattern more commonly observed in high-frequency data but ignored in most existing methods. Our method has been tested on the 100 stocks with largest capital in S&P 500 using 5-minute data, and its performance has been benchmarked with a single layer neural network model and the classical ARMA-GARCH model. The DNN model clearly outperforms to the other models in terms of prediction accuracy and Sharpe ratio. Given the parallelizable scheme of our method with DNN models, it may be capable for designing profitable trading strategies in high frequency time scale.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere thanks to my PhD advisor, Professor Song Wu. It is he who has led me to this PhD program and also introduced me to the area of machine learning, which interests me a lot and has made me believe that this is exactly the field I would like to work on. It is really amazing that, through Prof. Wu's guidance, I have grown to a PhD degree holder with knowledge of statistics and machine learning much more than I entered this program. During the years of research under Prof. Wu's guide, I also learned many other important skills such as: critical thinking, presentation skills, scientific writing skills etc. I understood that I still had a long way to improve these soft skills which might be more vital to my future career than technical skills. It's really my pleasure that at the corner of changing my major, I can have Prof. Wu as my research advisor and receive many priceless counsel from him.

I would like to thank my other committee members: Prof. Wei Zhu, Prof. Pei Fen Kuan and Prof. Keli Xiao, for reading through my dissertation and kindly providing me your comments and helps.

Also, I have special thanks to Prof. Wei Hou and Prof. Jie Yang in Medical Center of Stony Brook University for their financial support throughout my PhD study. Working as a research assistant in their projects has brought me large amount of experience for statistical consulting in aspects from modeling to writing up statistical reports.

I would like to thank all of my friends at Stony Brook University or working nearby. It's you guys bringing me a lot of fun and warmness beside of my research work. All those great time with you guys will remain in my memory and remind me the happy time at Stony Brook.

At last and to me the most important, I would like to thank my parents, Xiaoxing Chen and Aijuan Ni, for uncountable aspects and never ending love from them. It is with your firm support, I can face any difficulties and challenges not only for my previous student life but also for my future work.

CHAPTER 1 FUNDEMENTAL OF ARTIFICIAL NEURAL NETWORKS

1.1 Introduction

Artificial neural networks (abbreviated as “neural networks” in this thesis) were first proposed by McCullouch and Pitts in 1943, as a simplified mathematical model to mimic brain functions. In their original seminal papers [42], it was shown that with different threshold switches for neurons in the neural networks, even simple networks of this kind were able to model nearly any logic or arithmetic functions [32]. Later in 1947, the two authors indicated a practical field of application in the recognition of special patterns by neural networks [52] [34]. However, after that, research on neural networks has been quiet for a long time due to the difficulties in estimating the unknown parameters. It was not until 1986 when Rumelhart and others [58] proposed an efficient backpropagation algorithm for parameter estimations of the neural networks, and the development of this field has almost been explosive.

In early 1990’s, Kurt Hornik [26] [25] and Lee K. Jones [29] further laid down the theoretical foundations for neural networks, in which it proved that standard multilayer feed-forward networks are capable of approximating any measurable function to any desired degree of accuracy in a very specific and satisfying sense. Also, around the same time period, Yann LeCun’s researches on handwritten zip code recognition [38] [37] witnessed the great success of neural network models on pattern recognition. The database related to this project, MNIST (Mixed National Institute of Standards and Technology), has now become a benchmark dataset to compare different image processing systems.

Neural network models are essentially a set of nonlinear regression or classification statistical models. Around and after the year of 2000, many studies have demonstrated that simple neural network models may not achieve better performance comparing with other machine learning models like Random Forests and Support Vector Machine. This seems to put the practical usage of neural networks in question. However, more recently the development of new structures of neural networks, like deep learning [23] and convolutional neural networks [36] [8], has shown much better performance among machine learning models in pattern recognitions and thus has sparked new interests in neural networks. These new developments pointed out new directions for neural network models, that is, specific structures should be constructed according to the problems and datasets to be analyzed.

1.2 Model Structures

The model structure of neural networks can vary quite differently to deal with a broad range of nonlinear problems. In this section, we introduce some of the most widely used structures, often called as the single or two hidden layer feed-forward neural network (since we will be working with feed-forward neural networks throughout the whole thesis, sometimes we will omit the word "feed-forward" for simplicity). There have been a great amount of more advanced neural network structures, which will be introduced and investigated in later sections.

1.2.1 Single Hidden Layer Neural Networks

A single hidden layer neural network is a two-stage regression or classification model, typically represented by a network diagram as shown in Figure 1. It consists of three layers: input layer – the Xs, hidden layer – the Ws, and output layer – the Ys. Each circled node in the graph is a variable (or neuron) in the model with different colors distinguishing their relative positions; each line connects a pair of nodes from two adjacent layers and represents an unknown parameter (weight). The bottom layer consists of P nodes, which are explanatory variables serving as inputs of the

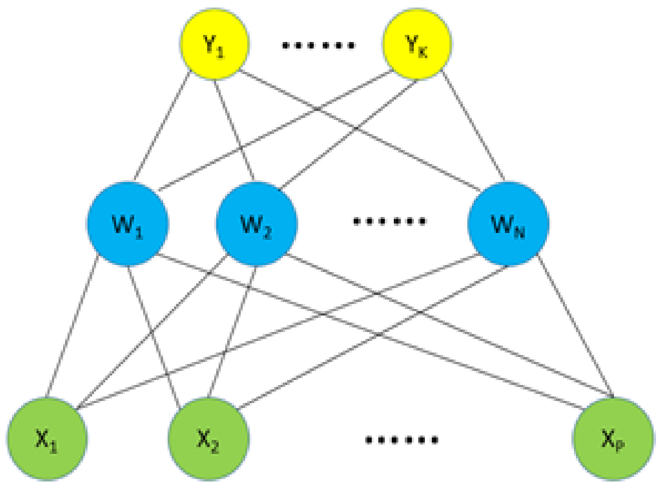


Figure 1. Single hidden layer neural network models. X neurons are the input variables, Y neurons are the output variables and W neurons are hidden neurons.

model. Similar to linear regression, categorical factors may be transformed into dummy variables, if necessary. The K nodes at the top correspond to the response variables. For a univariate regression, K simply equals to one and there is only one unit Y_1 at the top. For the purpose of generalizability, in the following we will deal with more general scenario with arbitrary dimension of K in the response. The middle layer is the most important part in the structure, consisting of N nodes, which are not directly observable. These variables are therefore “hidden” and only exist in concept. Variation in these latent variables enriches the class of neural networks.

The relationships between adjacent layers can be described by the following representations:

$$W_n = \sigma\left(\sum_{p=1}^{P} \alpha_{np} * X_p\right), \text{for } n = 1, \ldots, N \qquad (1.2.1)$$

$$Y_k = g_k\left(\sum_{n=1}^{N} \theta_{kn} * W_n\right), \text{for } k = 1, \ldots, K \qquad (1.2.2)$$

$\alpha s, \theta s$ are the model parameters, which are also called “weights”. $\sigma(\cdot)$ is the “activation function” (or “link function”) that controls the non-linearity of the model, and $g_k(\cdot)$s are the link functions in generalized linear models (identity functions for Y coming from normal distribution and softmax functions for Y coming from multinomial distribution). Basically, each W is a non-linear transformation of a linear combination of Xs, and each Y is another non-linear transformation of a linear combination of the Ws. If $\sigma(\cdot)$ is the identity function, the model degenerates to a standard linear model.

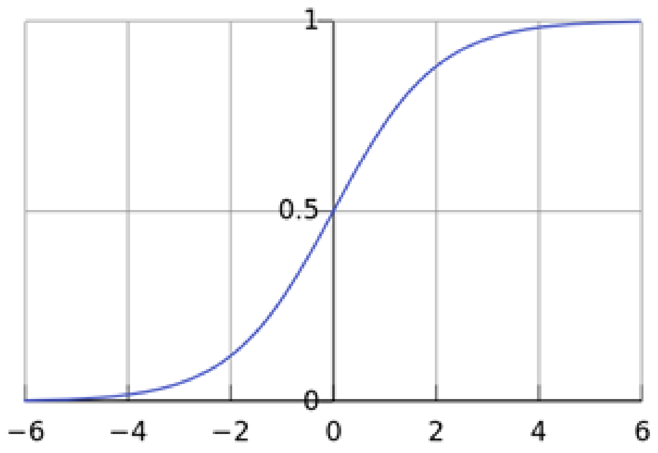


Figure 2. Sigmoid-like function $f(x) = \frac{1}{1+e^{-x}}$

The choice of activation

function σ is usually flexible, but generally it adopts the sigmoid-like shape as shown in Figure 2. The reason of restricting the choice in sigmoid-like function comes from the origins of neural networks. Simply speaking, the use of sigmoid-like function is to smoothen the step function: $s(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$, which mimic the pattern of neural activation. Since the step function is not smooth, or even not continuous, it will cause inconvenience in the optimization. Therefore the smoother, sigmoid-like function is introduced into the model and widely adopted.

Common choices of activation functions include logistic function $f(x) = \frac{1}{1+e^{-x}}$ (Figure 2), and hyperbolic tangent function $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}}$. Hyperbolic tangent function differs from logistic function in that it ranges within (-1, 1) interval. The logistic function can be extended or stretched to make it range in (-1, 1), that is $f(x) = \frac{2}{1+e^{-x}} - 1 = \frac{1-e^{-x}}{1+e^{-x}}$. The “extended” logistic function and the hyperbolic tangent function are then very similar. Since there are no restrictions on weights, i.e. they can be either positive or negative, the more symmetric hyperbolic tangent function or “extended” logistic function are more favored in real applications.

1.2.2 Two Hidden Layers Neural Networks

If one more hidden layer, the Z neurons, is added into the single layer network (Figure 3), then the model becomes a double-layer structure, which will be discussed thoroughly in this thesis and be applied into two different areas. As shown in Figure 3, it is more complicated and can better capture more complex patterns among inputs. Similarly, the relationships between variables in adjacent layers can be described by the following formulas:

$$W_n = \sigma\left(\sum_{p=1}^{P} \alpha_{np} * X_p\right), \text{for } n = 1, \dots, N \qquad (1.2.3)$$

$$Z_m = \sigma\left(\sum_{n=1}^{N} \beta_{mn} * W_n\right), \text{for } m = 1, \dots, M \qquad (1.2.4)$$

$$Y_k = g_k\left(\sum_{m=1}^{M} \theta_{km} * Z_m\right), \text{for } k = 1, \dots, K \qquad (1.2.5)$$

If more hidden layers were added to the network, more complicated multi-layer structure can be constructed. In general, activation functions used below the top layer usually take the same form (see Formula 1.2.3 and 1.2.4). In the following, we will give more discussions on how to choose between simple and complicated structures. Also, neural networks are really a large class of models, in which some parameters need to be pre-specified and some need to be tuned. In the next section, we will also discuss how to set up those important ones that are closely related to the model performance.

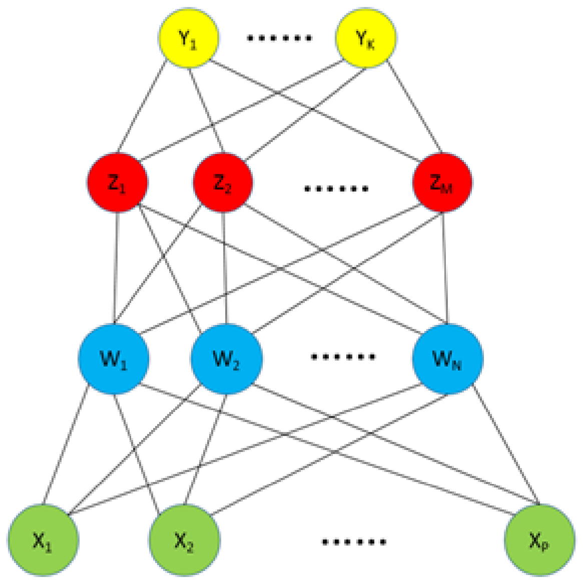


Figure 3. A two-layer neural network structure

The Y neurons at the top layer are output variables, the X neurons at the bottom layer are the input variables. The W and Z neurons in between are the first and second hidden layers respectively.

1.2.3 Single-layer vs. Multi-layer Structures

Kurt Hornik and his colleagues proved that if the number of nodes (neurons) is taken arbitrarily large, both single-layer and multi-layer structures can be universal approximations for continuous functions in $\mathbb{R}^P$ [25] [26], that is, they can approximate the target functions arbitrarily well.

In practice, single-layer networks are indeed widely adopted, due to their simple structures and maybe faster training processes. However, as shown in Figure 1, the single-layer network structure may fail to take into account of more subtle and local patterns among input variables, leading to the inefficiency in transmitting information from the bottom to the top layers. For cases that there might be local structures embedded among input features, the more flexible hierarchical multi-layer structure is more advantageous and allows different levels of resolution.

One successful application of multi-layer models is on image processing. LeCun's work on zip code data demonstrates how multi-layer network models can be applied in recognizing handwritten digits from images [38] [37]. The input factors are usually hundreds or even thousands numbers ranging from -1 to 1, describing pixel information. In this case, single hidden layer structure is inadequate since most vital features to differentiate the digits are local signatures and need more than one resolution. Multi-layer models are also very suitable for financial data, which are well known to be dependent. More examples will be given in later sections.

In general, for situations with few input factors (<30) and without much background knowledge, single-layer structures is a powerful tool. Otherwise, flexible multi-layer structures should be considered.

1.2.4 Number of Neurons in Hidden Layers

The number of neurons, e.g. N and M in the double hidden layer model in Figure 3, is another set of important parameters that can be tuned to improve the model performance. There is no systematic rule about how to specify these numbers. However, several aspects need to be considered while determining their values:

(1) Number of input factors. The first hidden layer right above the bottom resembles a feature selection process on the input factors, which suggests that $N > p$ may not fulfill the target. The same rational can be applied to the relationship between M and N.

(2) Number of observations in the data. If the number of observations in the data is small, too many parameters in the model would lead to a severe over-fitting issue (e.g. the total number of parameters in the double hidden layer model is $N \times P + M \times N + K \times M$). So, N and M cannot be too large.

(3) Datasets can suggest how to connect the input layer to the first hidden layer, so that N is controlled by such connections. We will cover this topic in more details in the next section.

Cross-validation may serve as a way to choose these tuning parameters, by performing grid-search among many, if not all, possible combinations of N and M and choosing the best pair. However, as will be discussed later, cross-validation is usually used to estimate regularization parameters, so using it to optimize N and M again seems redundant. The common practice is to specify them to be some numbers that are reasonable, in a range of 5 to 100 [20].

1.2.5 Connections between Layers

There are mainly three types of patterns that connect between layers: full connection, local connection, and direct connection. Full connection means that all nodes in the upper layer connect with all nodes in the lower layer. The structure shown in Figure 1 and Figure 3 belongs to this type, and is the most common connection. When all information from lower layer is needed to get the upper layer or any set of neurons in lower layer may interact with each other, this pattern would be a proper choice, while the main drawback is that too many parameters are needed. Several researchers [54] [16] [12] have used single-layer model with this pattern to analyze the stock market data, with input signals describing specific aspects of the whole stock market and certain stock.

If nodes in the upper layer only gather information from partial nodes in the lower

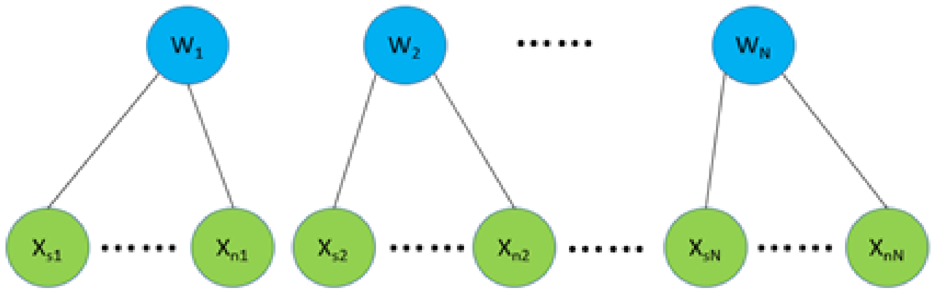


Figure 4. Local connection between input layer and 1st hidden layer.
X neurons from s_i to n_i only contribute to hidden neuron W_i.

layer, this suggests the structure of local connections. Figure 4 shows an example of how the input layer is locally connected with the hidden layer right above it. Each neuron in the hidden layer "retrieves information" only from a subset of all input factors: W_i only connects to Xs whose indices ranging from s_i to n_i, constrained that $s_1 = 1$ and $n_N = P$. Here, how to group the inputs is a vital step, which is usually determined by the background knowledge or experimentation. Additionally, overlapping between nearby groups is acceptable, which is to say s_2 may be smaller than n_1. The local connectivity has been successfully used in LeCun's work on handwritten zip code recognition in image [38] [37]. If justified, local connections can eliminate irrelevant connections and reduce the total number of parameters, yielding much more efficient neural networks.

Direct connection is a pattern which reflects the phenomenon that some input factors may affect the output more directly than others. Therefore, these factors should be "linked" to higher hidden layers or even to the output layer directly. Figure 5 shows an example for two input factors directly connected to nodes in the second hidden layer. This pattern is very useful in a variety of data sets. In genetics, some demographic variables, such as gender, age or life style, obviously do not function at the same level as more detailed genetic information, such as single nucleotide polymorphism, so it is recommended to connect those input factors directly to higher hidden layer. For stock market data, some market indicators, such

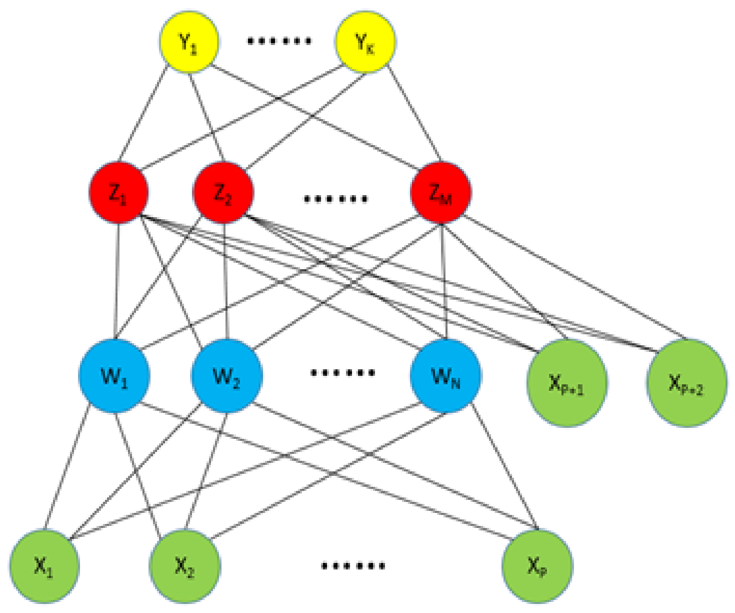


Figure 5. Direct connection.

Input neurons X_{p+1} and X_{p+2} are directly connected to the second hidden layer (Z neurons).

as risk-free interest rate and unemployment rate, which reflect the overall economic environment, may affect the prediction on the stock performance more than historical data from other stocks, so they may be directly connected to the output layer.

The combinations of the basic connections can generate a rich class of network structures.

1.3 Training Neural Networks

1.3.1 Input Processing

In fitting neural networks, the scaling of the input variables plays an important role in the parameter/weight estimation. For example, a variable that ranges between -10 and 10 will outweigh a variable that ranges between -1 and 1, that is, variables measured at different scales do not contribute equally to the model fitting. Typically, the inputs of a neural network model need to be prepossessed and scaled to the same range like (-1, 1). The rescaling can also benefit the regularization and choosing the initial values. For financial data or time series data, relative increase rate or logarithm of the rate are often used instead of raw stock prices or indexes [43], which serves to have rescaled the outcome variables.

Feature construction (like PCA) is usually an important preprocessing step for other machine learning methods, however, it is relatively less important in neural networks. The reason is that when the “information” flows from the lower layer to higher layer in the network, it already resembles a process of feature construction. The only issue is that the process is so complicated that the constructed features become almost uninterpretable. Nevertheless, if the prediction accuracy is our final goal, sacrificing the interpretability may be worthwhile.

1.3.2 Error Functions

To estimate the unknown parameters or the weights in the neural network models, we need to minimize the error functions. Depending on the nature of the response variable, we may have either regression or classification problems. Also, in order to avoid the over-fitting issue, regularization terms may be added to the error functions. Below we will discuss how to construct a proper error function.

Let's consider the double layer neural network in Figure 3, and denote its complete set of parameters (weights) as Ω, which consists of $\{\alpha_{np}, \beta_{mn}$ and $\theta_{km}: n = 1, \dots, N; p = 1, \dots, P; k = 1, \dots, K; m = 1, \dots, M\}$.

For regression models, the error function can be simply set to be the sum-of-squared errors:

$$R(\Omega) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - \hat{y}_{ik})^2 \tag{1.3.1}$$

For classification models (categorical responses), the error function can be the cross-entropy (deviance):

$$R(\Omega) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log(\hat{y}_{ik}) \tag{1.3.2}$$

And the corresponding classifier is $G(x) = argmax_k\{\hat{y}_k(x)\}$ [20].

Since the number of parameters in Ω is typically large, it is very easy to over-fit the network model, so a global minimizer of $R(\Omega)$ is usually not desired. Instead, early stopping rules may be placed to obtain local minimizer, or regularization may be added through some penalty terms to avoid overfitting. In this study, we will focus on the penalty methods, which will be discussed below.

A L_2 −penalty can be used in the network models. Then the final form of error functions becomes:

$$R_p(\Omega) = R(\Omega) + \lambda J(\Omega) = R(\Omega) + \lambda ||\Omega||^2 \tag{1.3.3}$$

In the above formula, $R(\Omega)$ is the original error function in (1.3.1) or (1.3.2), $J(\Omega)$ is the penalty term, with $\lambda \geq 0$ being a tuning parameter. Larger values of λ will tend to shrink the weights toward zero. To choose an appropriate value for the hyper-parameter λ, cross-validation can be applied.

1.3.3 Learning Algorithms

Next, we discuss how to minimize the error function defined in (1.3.3). Since this is a non-linear problem, numerical algorithms will be applied for the optimization. In the following, we will discuss three popular methods:

1. Back-propagation: This is essentially a search method that is based on gradient descent. The first-order derivatives of $R_p(\Omega)$ with respect to elements of the parameter vector Ω are computed first, and then Ω are updated with some part of the derivatives until certain stopping criteria are reached.
2. Simulated annealing: This is a stochastic search method, which does not rely on derivatives. Briefly, we start with an initial guess Ω_0, and proceed with random updating of the initial guess until a "cooling temperature" or stopping criterion is reached.
3. Genetic algorithm: This is an evolutionary stochastic search, which starts with a population of s initial guesses Ω_0, and updates the population of guesses by genetic selection, breeding, and mutation for many generations, until the best coefficient vector is found among the last generation population.

1.3.3.1 Back-propagation

This algorithm was firstly introduced by Rumelhart *et al* in 1986 [58] [59], for optimizing nonlinear functions in (1.3.3). Again, the two hidden layer structure (Figure 3) is used to demonstrate the process.

Firstly we make a random guess Ω_0 for Ω. Then suppose we are at the r^{th} iteration and the estimations for Ω are denoted as Ω_r. Then the derivatives of $R_p(\Omega)$ with respect to Ω at Ω_r are expressed as:

$$\nabla_r = \left.\frac{\partial R_p(\Omega)}{\partial \Omega}\right|_{\Omega=\Omega_r} \tag{1.3.4}$$

∇_r consists of three sets of elements:

$$\left\{\frac{\partial R_p(\Omega)}{\partial \alpha_{np}}\bigg|_{\Omega=\Omega_r} : n = 1, \dots, N; p = 1, \dots, P\right\} \tag{1.3.5}$$

$$\left\{\frac{\partial R_p(\Omega)}{\partial \beta_{mn}}\bigg|_{\Omega=\Omega_r} : m = 1, \dots, M; n = 1, \dots, N\right\} \tag{1.3.6}$$

$$\left\{\frac{\partial R_p(\Omega)}{\partial \theta_{km}}\bigg|_{\Omega=\Omega_r} : k = 1, \dots, K; m = 1, \dots, M\right\} \tag{1.3.7}$$

We use the following formula to update the estimations of Ω:

$$\Omega_{r+1} = \Omega_r - \gamma_r \cdot \nabla_r \tag{1.3.8}$$

γ_r is called the learning rate, which is a positive series decreasing to zero. It is common to set: $\gamma_r = 1/r$.

To illustrate how the formula works, let's take a look at the second-order Taylor expansion of $R_p(\Omega)$:

$$R_p(\Omega) = R_p(\Omega_r) + (\Omega - \Omega_r)' \cdot \nabla_r + \frac{1}{2}(\Omega - \Omega_r)' H_r (\Omega - \Omega_r)$$

Suppose Ω_r is close to the minimum point. Then, taking the derivative of the above equation with respect to Ω yields:

$$0 = 0 + \nabla_r + H_r(\Omega - \Omega_r)'$$

So, Ω at the step $r + 1$ is: $\Omega_{r+1} = \Omega_r - {H_r}^{-1} \cdot \nabla_r$.

H_r is the Hessian matrix of $R_p(\Omega_r)$. It is usually very time-consuming to calculate the Hessian matrix, and the inverse of H_r may bring some other problems like the singularity issue. The learning rate γ_r in (1.3.8) can be viewed as a simplification of the inverse of Hessian matrix. There are some other ways to handle this problem as well, such as the BFGS (Boyden-Fletcher-Goldfarb-Shanno) algorithm, please refer to other reference for details [48].

The explicit form of ∇_r could be derived by the chain rule for differentiation. To derive the explicit form of ∇_r, we first need to rewrite Formula (1.2.3) – (1.2.5) with an additional subscript i representing different observations:

$$W_{in} = \sigma\left(\sum_{p=1}^{P} \alpha_{np} * X_{ip}\right), \text{for } n = 1, \dots, N \qquad (1.2.3)^*$$

$$Z_{im} = \sigma\left(\sum_{n=1}^{N} \beta_{mn} * W_{in}\right), \text{for } m = 1, \dots, M \qquad (1.2.4)^*$$

$$\hat{y}_{ik} = g_k\left(\sum_{m=1}^{M} \theta_{km} * Z_{im}\right), \text{for } k = 1, \dots, K \qquad (1.2.5)^*$$

Then by the chain rule of differentiation, we have:

$$\begin{aligned}\frac{\partial R_p(\Omega)}{\partial \theta_{km}} &= \sum_i \frac{\partial R_p(\Omega)}{\partial \hat{y}_{ik}} \frac{\partial \hat{y}_{ik}}{\partial \theta_{km}} = \sum_i \frac{\partial R_p(\Omega)}{\partial \hat{y}_{ik}} g_k'\left(\sum_{m=1}^{M} \theta_{km} * Z_{im}\right) Z_{im} \\ &= \sum_i \epsilon_{ik} Z_{im}\end{aligned} \qquad (1.3.9)$$

In the last equation of (1.3.9), we assign:

$$\epsilon_{ik} = \frac{\partial R_p(\Omega)}{\partial \hat{y}_{ik}} g_k'\left(\sum_{m=1}^{M} \theta_{km} * Z_{im}\right) \qquad (1.3.10)$$

Using the defined ϵ_{ik}, we can have:

$$\begin{aligned}\frac{\partial R_p(\Omega)}{\partial \beta_{mn}} &= \sum_i \frac{\partial R_p(\Omega)}{\partial Z_{im}} \frac{\partial Z_{im}}{\partial \beta_{mn}} = \sum_i \sum_{k'} \epsilon_{ik'} \theta_{k'm} \sigma'\left(\sum_{n=1}^{N} \beta_{mn} * W_{in}\right) W_{in} \\ &= \sum_i \eta_{im} W_{in}\end{aligned} \qquad (1.3.11)$$

In the last equation of (1.3.11), we assign:

$$\eta_{im} = \sum_{k'} \epsilon_{ik'} \theta_{k'm} \sigma'\left(\sum_{n=1}^{N} \beta_{mn} * W_{in}\right) \qquad (1.3.12)$$

Using the defined η_{im}, we can have:

$$\frac{\partial R_p(\Omega)}{\partial \alpha_{np}} = \sum_i \frac{\partial R_p(\Omega)}{\partial W_{in}} \frac{\partial W_{in}}{\partial \alpha_{np}} = \sum_i \sum_{m'} \eta_{im'} \beta_{m'n} \sigma'\left(\sum_{p=1}^{P} \alpha_{np} * X_{ip}\right) X_{ip} \qquad (1.3.13)$$

We firstly calculate the derivatives for θ_{km}, then through ϵ_{ik}, the derivatives of θ_{km} are "propagated" to the next level, β_{mn}. After calculating the derivatives for β_{mn}, the derivatives are again "propagated" to α_{np}, through η_{im}. At last, we come up with the derivatives for α_{np}. Since the derivatives are calculated from the output layer parameters θ_{km} to the input layer parameters α_{np}, this algorithm is named as "back-propagation".

The inconvenience is that every time ∇_r needs to be re-derived for different network structures. Back-propagation is probably the most widely used algorithm in dealing with neural networks, due to its fast implementation. However, since its estimation heavily depends on initial values that are chosen randomly, there is no guarantee that any single estimation is optimal. To solve this problem, we might use several different initial values to start the back-propagation processes and choose the best one or average on top ones to alleviate the issue of initial values.

1.3.3.2 Simulated Annealing

Simulated annealing originated from stochastic methods, consisting of a mutation process and an accept/reject algorithm. The algorithm works as follows [20]:

1. At the step $r = 0$, generate a randomly vector Ω_0 for the unknown parameters, based on which the error function is calculated: $\Psi_0 = R_p(\Omega_0)$
2. At the step $r + 1$, randomly perturbate previous vector Ω_r to be $\widehat{\Omega}$. For example, let $\widehat{\Omega} = \Omega_\mathrm{r}(1 + \epsilon)$ with $\epsilon \sim N(0, 0.03^2)$. And then calculate the corresponding error function: $\widehat{\Psi} = R_p\big(\widehat{\Omega}\big)$
3. The cooling temperature at step $r + 1$ is then defined as:
$$T(r+1) = \frac{\bar{T}}{1 + \ln(r+1)}$$
where $\bar{T}$ is a pre-specified scalar, i.e. the "temperature", which will be discussed in more details after introducing the whole process.
4. Define metropolis ratio:
$$M(r+1) = \exp\left(\frac{-(\widehat{\Psi} - \Psi_r)}{T(r+1)}\right)$$

5. Let $C(r+1) = \min(1, M(r+1))$, and generate a random number U uniformly from the interval $(0, 1)$. If $U < C(r+1)$, $\Omega_{r+1} = \widehat{\Omega}$; otherwise $\Omega_{r+1} = \Omega_r$.
6. Continue until $r = \bar{T}$

Parameter $\bar{T}$ is a key parameter in this algorithm, since it controls the total number of iterations. So, on one hand, $\bar{T}$ should be large enough to ensure many perturbations have been tried before reaching the computational minima; on the other hand, $\bar{T}$ cannot be too large as it may lead to longer estimation, but with only marginal gain on the parameter estimation. It is suggested that some trials should be performed to find a balanced value for $\bar{T}$ in this trade-off.

The difference of new error function to the error function in the previous step: $\widehat{\Psi} - \Psi_r$ determines how unlikely the process will accept the new "guess". Note that since the scaling factor of the difference $T(r+1)$ is a deceasing function of step r, this suggests that for similar values of $\widehat{\Psi} - \Psi_r$, Ω in later steps (larger r) is more likely to stay unchanged.

1.3.3.3 Genetic Algorithm

Genetic algorithm (GA) is another popular stochastic search method in nonlinear optimization, which originated from the evolutionary theory. Same as simulated annealing, no derivatives need to be calculated throughout the algorithm, and it also starts with one random initialization vector Ω_0. The GA algorithm works as follows:

1. Set up a set of random guesses for Ω, which is called a population (e.g. N^* random vectors): $\boldsymbol{\Omega_0} = \{\Omega_{0i}: i = 1,2, \dots, N^*\}$.
2. In the Gth generation, N^* pairs of "parents" were chosen from the population $\boldsymbol{\Omega_G}$ to "generate" new children –members in $\boldsymbol{\Omega_G}$ can be parents of more than one child.
   - 2.1 Suppose Ω_{G1} and Ω_{G2} are chosen as one pair. Crossover and mutation are then performed on these two members to yield a new vector.
   - 2.2 Crossover is to swap part of parents' "DNA" – the parameter vectors in the algorithm. There are several kinds of crossovers: shuffle crossover, arithmetic crossover, single-point crossover [43]. Arithmetic crossover can only be used when the parameters are in real-valued encoding.

2.3 Mutation is to make changes directly on a member. For example, define mutation probability $\widetilde{pr}$ as: $\widetilde{pr} = 0.15 + \frac{0.33}{G}$, which determines whether mutation happens on certain child. If mutation exists for example on the jth parameter of child $C1$, then change $C1_j$ to $\widetilde{C1_j}$ as following:

$$\widetilde{C1_j} = \begin{cases} C1_j + s\left[1 - r_2^{\left(1-\frac{G}{G^*}\right)^b}\right] & if\ r_1 > 0.5 \\ C1_j - s\left[1 - r_2^{\left(1-\frac{G}{G^*}\right)^b}\right] & if\ r_1 \leq 0.5 \end{cases}$$

r_1 and r_2 are two real numbers randomly chosen from [0,1]. s and b are two parameters which should be pre-determined. More detailed discussions on the crossover and mutation can be found in [43].

3. Evaluate the error functions R_p in (1.3.3) for $\{\Omega_{G1}, \Omega_{G2}, C1, C2\}$, and chooses the two with smallest R_p to be the members in generation $(G + 1)$.
4. R_p is calculated on both the Gth and $(G + 1)$th generations. If the smallest R_p among Gth generation is also the smallest among $(G + 1)$th generation, that particular member will replace the one with highest R_p in $(G + 1)$th generation.
5. Repeat the above steps to G^* times, with the best member in the last generation being the desired parameter estimation. G^* is the parameter determining when to stop, just like $\bar{T}$ in simulated annealing algorithm.

Both genetic algorithm and simulated annealing are stochastic methods for optimization, and share the common advantage of trying new solutions randomly in the neighborhood of existing solutions. If the new solutions are better or not too worse than existing ones, the new solutions are used for searches in the next step. However, one more advantage for genetic algorithm is that it keeps several possible solutions (N^* members in current population) and generate more ways to move from the existing solution to new better solutions through the crossover (mutation is like the "updating" method in simulated annealing).

Back-propagation is a greedy algorithm that can perform exhaustive search in a particular local area. However, since it can only move the target function in one direction, the optimization can be easily trapped in a local minimum. In contrast, stochastic methods may move the target function both up and down so it can escape from local minima; however, the optimization then has the risk of leaving an area too soon. Based on these characteristics, hybrid methods that combine the strengths of both back-propagation and stochastic methods have also been proposed [43].

1.3.4 Parameter Selection

After several trials of model fitting, researchers may notice that some input factors may be more important than others, and some may be more dispensable and should be excluded. So how to retain those important factors and filter out unimportant ones is an interesting problem. Here we will describe a method called parameter decreasing method (PDM) for parameter selection [65]. Although for the prediction purpose, it may not be necessary or productive to conduct parameter selection in neural networks due to its unique structure, for some cases where the model interpretation is important, selecting proper factors can be very meaningful.

Let's use the local connections in Figure 4 to illustrate how the PDM works. Each group of input factors $\{X_{si}, \ldots, X_{ni}\}$ together with their corresponding higher neuron W_i is viewed as one unit. PDM conducts selection at this "unit" level, that is, the process eliminates one unit each time and evaluates how model performance may change, such as the accuracy rate based on a validation data set. After going through all units, the one having least impact on performance would be discarded. This process will iterate until some stopping criteria meets – the maximum number of remaining units or the maximum decrease of performance when eliminating one unit. This resembles a backward selection procedure.

1.4 Neural Networks in Research and Applications

After backpropagation algorithm was proposed, the research field of neural networks had undergone dramatic changes, and gradually evolved to cover a wide range of models and learning methods in many fields, from explosives detection system in airport [62] to handwriting recognition in artificial intelligence [37], and to photographing and cancer cell detection [74].

As a powerful prediction tool, neural network can be a great option for certain applications. For example, it can be applied in marketing analyses to help identify potential customers and their preference for specific needs, based on their browsing history on internet, goods bought in the past and other demographic information. With these information, merchants would know who may be their potential customers and what they need, and then could design targeted advertisements. Not only could this type of analysis help merchants reduce their advertisement costs significantly [69], it can also help customers receive relevant information more efficiently. Classification, predictive modeling, pattern recognition and novelty detection (like E-mail spam filtering [9]) might be the most wide application field for neural networks.

Neural networks can also contribute in unsupervised learning like clustering and nonlinear principal component analysis [21] [33]. The idea of performing nonlinear principal component analysis utilizing neural networks is trying to replicate input neurons in output neurons, so that the hidden neurons would serve as a good set of representatives for input neurons. Neural networks have had a promising application in artificial intelligence, since the origin of neural network models is to mimic functionalities of human brains. Many aspects of artificial intelligence really need powerful machine learning models like: learning, natural language processing and manipulation.

More recently, applications of neural networks in financial industry have been proposed. Many published papers have proved its success in financial market. Some of the automated trading systems are also built upon neural network models. In Chapter 3 of this thesis, we will give a more detailed summary on how the neural networks have been applied in finance, especially on stock price predictions.

1.5 Motivations and Contributions

Although neural network models have been widely applied in different areas, most of the applications still utilize the single layer neural networks, which is the simplest model structure in the family of neural networks. Single layer neural networks might be enough for analyzing some simple data sets, when the number of covariates is small compared with the sample size and covariates are almost independent with each other. However, for more complicated datasets, single layer neural networks may not perform well. Many researchers comparing neural networks with other machine learning methods like SVM or Random Forests often concluded that neural networks cannot outperform to the other competitors in some specific applications. As we mentioned in Section 1.1, the reason may lie on the fact that most of them actually only tried the single layer neural networks.

After reviewing many papers on neural network applications, we believe that to achieve the best performance of neural networks, special structures should be designed for different datasets, tailored to their own data characteristics. In this thesis work, we will design a novel double layer neural network model with local connections from input layer to the first hidden layer. Our model with this unique design is expected to utilize the correlation information and help improve the model performance. We demonstrated that neural network models with this structure can better deal with covariates which high correlations. Our model have been applied and tested in two different areas:

- Genome-wide Association Studies (GWAS). In GWAS, hundreds and thousands of SNPs are genotyped, and SNPs that are physically close are usually highly correlated with each other. Our model tries to utilize these correlation information to help detect causal SNPs (and their interactions). Simulation studies have been conducted to test our model and compare with Random Forests. The model has also been applied to a real COGEND data. (Chapter 2)
- Prediction of stock price movement at 5-minute level. Stock prices show greater correlations in high frequency scale than in daily or monthly level. Our model will incorporate the price correlation information and other technical indicators to help in price prediction in high-frequency data. We also design a framework for our unique model to help make decisions on transactions. The model was tested on S&P 500 stock prices from 1/1/2013 to 5/31/2013. (Chapter 3)

In addition, we have developed in-house software for the model implementation, which are mainly written in C language to achieve better computational speed. Different from existing packages for neural networks, we calculate the first derivatives of error functions in their explicit form instead of using numerical methods. So our in-house software can train our neural network models much faster than those existing ones, e.g. the “nnet” packages in R.

CHAPTER 2 GENOME-WIDE ASSOCIATION STUDY USING NEURAL NETWORK MODELS

2.1 Introduction

2.1.1 Some Notations in Genetics

In modern genetics, genome refers to the entirety of an organism's hereditary information, which is usually mediated through a combination of four possible DNA nucleobases, namely A, T, C and G. Particularly, a human genome consists of approximately 3 billion of nucleobases, packed into 23 pairs of chromosomes. For a diploid species like human, one of the paired chromosomes originates from the paternal and another one from the maternal. Both the paternal and maternal chromosomes contain almost identical nucleobases sequences with over 99.9% similarity, except at certain

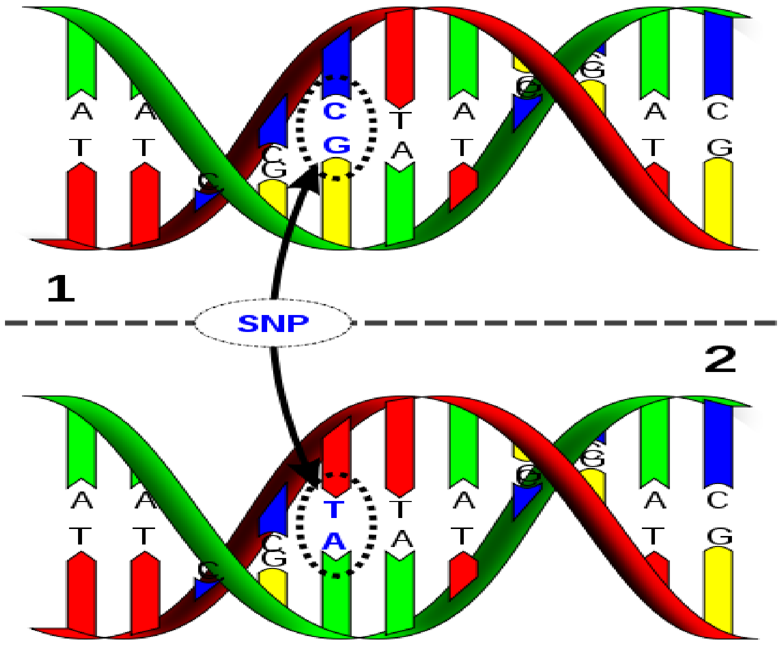


Figure 6. An illustration of Single Nucleotide Polymorphism (SNP)
Two double strands are selected here as an explanation of SNPs. For the first double-strand, there is a loci with genotype C/G. At the same loci of the second double-strand, the genotype is T/A. Then we know this loci is a C/T polymorphism if we choose the red strands as the representative (http://en.wikipedia.org/wiki/Single-nucleotide polymorphism).

nucleobases, which are called as single-nucleotide polymorphism, or SNPs. Figure 6 shows an example of SNP with C/T polymorphism. Base on the newest summary data of dbSNP database (http://www.ncbi.nlm.nih.gov/projects/SNP), there are approximately 38 million (about 1 percent of the total genome) of validated SNPs in human genome.

Although in theory, any of the four nucleobases may appear at any genomic position, in practice, it has been observed that most SNPs are actually bi-allelic, which means they only contain two possible nucleobases, generating three genotypes. For example, if a SNP has two alleles, A and C, the possible outcomes (genotypes) of this SNP could only be AA, AC and CC. AA and CC are called homozygotes, and AC is called heterozygote. In the rest of this paper, we assume SNPs are bi-allelic.

In a nature population, if subjects mate at random, a phenomenon called Hardy-Weinberg Equilibrium (HWE) may occur, which states that the allele frequencies of a SNP in a population remain constant from generation to generation unless specific disturbing influences are introduced. To illustrate HWE, let's take a single SNP locus with A/C alleles as an example: Suppose the frequency of allele A to be $P(A) = p$, then allele frequency of allele C is$P(C) = q = 1 - p$; from the HWE, we have $P(AA) = p^2; P(AC) = 2pq; P(CC) = q^2$. Generally speaking, the allele frequency of each SNP is different.

Minor allele frequency (MAF) is an important characteristic of SNP, defined as: $MAF = \min\{p, 1 - p\}$ for each SNP. Usually, SNPs with very small MAF (for example less than 0.01) is difficult to be studied thoroughly due to their low frequencies, which also cause some difficulties for the genome- wide association studies. This topic will be discussed in the third section of this chapter. With the development of new genotyping technology, more SNPs with small MAF that could not be discovered before are reported more and more recently.

Another important characteristic of SNP is the Linkage Disequilibrium (LD) among SNP loci. The LD refers to non-random associations between the genotypes of different loci. If a group of SNP loci follow an LD pattern, then some combinations of their genotypes would appear more often than random formations. It is known that the LD strength between SNPs is closely related to their physical distance. Because of the LD, we can infer unknown SNPs based on known SNPs and it serves as the major principal for the artificial neural network methods we plan to study in this thesis.

Table 1 shows the joint probabilities of genotypes of two SNP loci, expressed as their haplotype frequencies. If the two loci are independent, then:

$$\begin{aligned} p_{AB} &= p_A p_B; \quad p_{Ab} = p_A p_b \\ p_{aB} &= p_a p_B; \quad p_{ab} = p_a p_b \end{aligned} \tag{2.1.1}$$

Otherwise, an extra parameter D needs to be introduced and the haplotype frequencies are redefined as:

$$\begin{aligned} p_{AB} &= p_A p_B + D; \quad p_{Ab} = p_A p_b - D \\ p_{aB} &= p_a p_B - D; \quad p_{ab} = p_a p_b + D \end{aligned} \tag{2.1.2}$$

The quantity D is the so-called LD measure. It reflects the difference compared with the expected genotype combination frequencies under the assumption of independence. $D \neq 0$ means the two loci are in linkage disequilibrium.

Table 1. Joint probabilities of genotypes of two SNP loci.
The probabilities of each genotype combination are shown in Formulas (2.1.1) and (2.1.2) for the two SNP loci being independent or not.

Probabilities	AA	Aa	aa
BB	p_{AB}^2	$2p_{AB}p_{aB}$	p_{aB}^2
Bb	$2p_{AB}p_{Ab}$	$2(p_{AB}p_{ab} + p_{Ab}p_{aB})$	$2p_{aB}p_{ab}$
bb	p_{Ab}^2	$2p_{Ab}p_{ab}$	p_{ab}^2

From the definition of D, we know that D's for different loci pairs may have different ranges. Then the direct comparison between D's is not sensible. We need to do some normalization on the parameter. One type of normalization is $D' = \frac{D}{D_{max}}$, with:

$$D_{max} = \begin{cases} \min(p_A p_B, p_a p_b), when\ D < 0 \\ \min(p_a p_B, p_A p_b), when\ D > 0 \end{cases}$$

Another kind of normalization is to use the correlation coefficient, which is expressed as: $r = \frac{D}{\sqrt{p_A p_a p_B p_b}}$.

2.1.2 Genome-Wide Association Study

It is well known that most common human traits, such as diseases, are likely to be complex, that is, many genetic factors may contribute to development of these complex traits. Typically, it is very hard to identify these genetic factors, as each causal gene may only make a small contribution to the disease phenotype. Many approaches have been developed to uncover causal genes (or SNPs) for complex traits/diseases. Roughly speaking, the existing methods can be categorized into three groups: candidate-gene studies, linkage mapping and genome-wide association studies [24].

Here we give a brief introduction to the candidate-gene association studies, which can be easily extended to the genome-wide association studies. In a simple case-control study, candidate-gene studies compare the frequencies of alleles or genotypes of a particular variant between disease (case) and control groups. The genes are usually preselected, based on certain biological hypotheses or the location of the candidate within a previously determined region of linkage. One limitation of this approach is that they can identify only a fraction of genetic risk factors, even for diseases with well-understood pathophysiology and if the test hypotheses are broad (for example, involving the testing of all genes in the insulin-signaling pathway). When the fundamental physiological defects of a disease are unknown, the candidate-gene approach is clearly inadequate.

As an extension of candidate-gene studies, genome-wide association approaches surveys most of the genome for causal genetic variants. Here the genetic variants could be genes or base loci - SNPs. This is a data-driven discovery approach with no assumption on causal variants, and it could exploit the strengths of association studies without having to guess the identity of the causal genes. The genome-wide association approach therefore represents an unbiased yet fairly comprehensive option that can be attempted even in the absence of convincing evidence regarding the function or location of the causal genes or SNPs.

2.1.3 Difficulties in GWAS

From genetics, we know that a major part of the genetic variations are due to SNPs. This means if we had known all SNPs in human genome, we could in theory understand how most of the genetic factors may affect different phenotypes. However, even the most recent SNP array platform can only survey 1 million SNPs, which is still a small fraction of all the SNP loci. So a GWAS study is typically performed with incomplete SNP list. In this case, the LD pattern we introduced previously may help identify the missing causal SNPs, based on the hope that sampled SNPs correlated with the missing causal SNPs provide information for that disease. Figure 7 shows how LD pattern is used in genome-wide association studies. Figure 7a shows the desired situation: the causal SNP is genotyped, so we can directly analyze its association with some phenotype (for example diseased or not). In Figure 7b, the casual SNP (blue one) is not genotyped; however, SNPs are in an LD pattern with it are genotyped (red ones). In either case, we can still perform association studies based on the genotyped SNPs, because they provide some genetic information of casual SNPs, however, the power may be less when causal SNPs is missing from the genotyped set.

Epistasis (SNP-SNP interaction) is another layer of difficulties in GWAS [13]. For a set of 1 million SNPs, an exhausted search for all possible interactions is almost

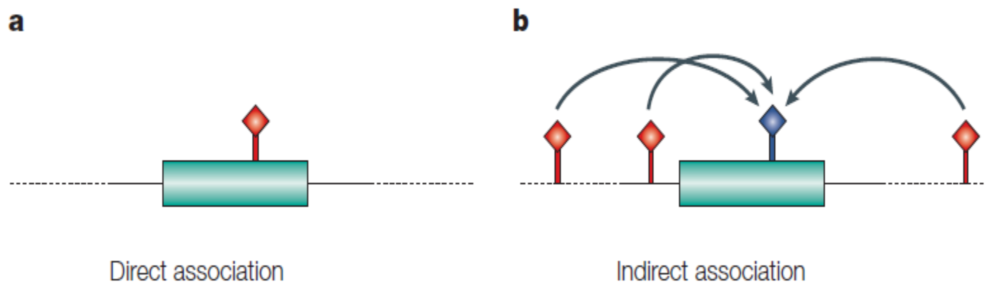


Figure 7. Two association patterns – causal SNPs are genotyped or not

Two kinds of associations are shown here. Picture a shows the direst association which means the SNP marked red is the causal SNP and genotyped in the study. Picture b shows the indirect association - the blue SNP is the causal one but not genotyped, those red ones are genotyped and associated with the causal blue one [24].

computationally infeasible. However, some pseudo-exhaustive searching has been developed to specifically deal with epistasis, such as Multifactor-Dimensionality Reduction method (MDR, introduced in Section 2.2.1). They can usually handle about ten SNP loci at one time. Another group of methods try to discover epistasis effects based on SNPs with significant marginal effects. However, some constructed examples [70] show that SNPs having high interaction effects may not have high marginal effect at all. How to find out the SNP interactions has become one of the hardest problems in GWAS, which will be a focus of this thesis.

In addition to SNP-SNP interactions, SNP-environment interaction also plays an important role in GWAS. For most biological traits, genetic factors alone cannot fully predict the phenotypes. Environmental factors like ethnicity and lifestyle may affect the phenotypes as well. Determining which environmental factors should be included in the study is a hard but important problem

2.2 Existing Popular Epistasis Methods in GWAS

In a simplified form, a GWAS can be viewed as a statistic model: the SNP data are the input variables, and the phenotype of interest, e.g. disease status, is the response variable. Then in the case of binary phenotype, the most straightforward idea is to fit a logistic regression model to the data. However, typically the number of SNPs are much larger than the number of observations. This yields a "small p, large n" problem and the common regression methods cannot be directly applied. In this chapter, we will review three popular epistasis methods: multifactor-dimensionality reduction, random forests and Bayesian epistasis association mapping. These three methods are suggested to be the most widely used ones among tons of existing methods [3] [10].

2.2.1 Multifactor-Dimensionality Reduction

MDR method is a well-known method in GWAS, not only because it is straightforward and its computation process is very easy to implement, but also because it performs exhaustive search: searching for all possible interaction combinations among the potential causal factors (genes or SNPs).

MDR was first introduced by Ritchie and others in 2001 [57]. In the original paper, the authors used this method to investigate sporadic breast cancer and discovered a four-locus interaction term having high association with the disease. It was arguably the first time to identify such a high-order interaction for a complex multifactorial disease. In a following paper [56], they discussed the power of MDR with more details.

Figure 8 demonstrates the four general steps to implement the MDR method in a case-control study:

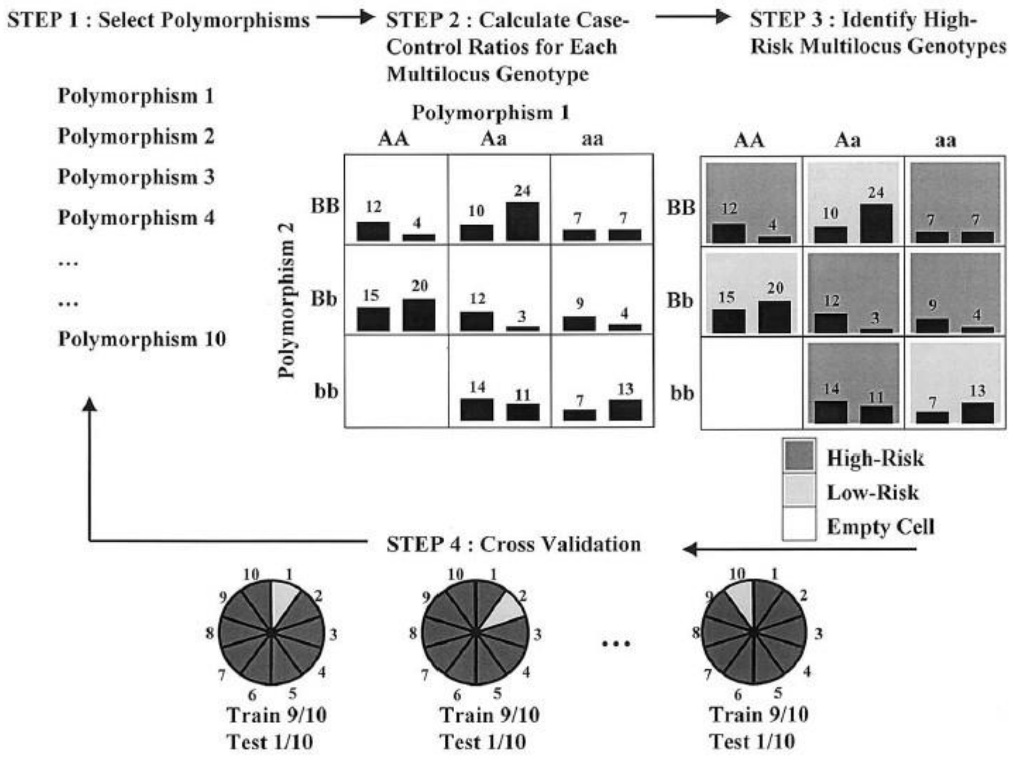


Figure 8. Steps for Multifactor-Dimensionality Reduction method

1. Determine the potential causal SNP loci and their genotypes; 2. For each possible interaction term, generate case-control ratios for each genotype; 3. Identify high risk multilocus genotypes; 4. Cross-validate step 2 and 3. [57]

(1) A set of n genetic and/or discrete environmental factors is selected from the pool of all factors.

(2) The n factors and their possible multifactor classes or cells are represented in a n-dimensional space. As an example in this figure, a two-locus case is considered, which means a two-dimensional space. Totally there are nine two-locus genotype combinations. Then, the ratio of the number of cases to the number of controls is estimated within each multifactor class.

(3) Each multifactor cell in n-dimensional space is labeled either as high-risk if the cases/controls ratio meets or exceeds some threshold or as low-risk otherwise. In this way, a model is formed by pooling high-risk cells into one group and low risk cells into another group. Usually we choose 1 as the threshold in case-control studies because it is the totally randomized ratio for each cell.

(4) A 10-fold cross-validation is done for the above three steps. That is, the MDR model is developed for each possible 9/10 of the subjects and then used to make predictions about the disease status for each subject in the rest 1/10 of the subjects. The proportion of subjects on which incorrect predictions are made is an estimation of the prediction error of that model.

Suppose the total number of factors is m, and is bigger than two. We should try each $n = 2, 3, \dots, m$ and each possible combination of n factors in the above four steps. Among all the n-factor models, the one maximizing the cases/controls ratio of the high-risk group is selected. [57] computes the training error for the 9/10 data and chooses the one with the lowest training error. For the selected n-factor model, there are two quantities which can represent its goodness of fit. One is the prediction error, and the other is the times the model is chosen among the 10-fold cross-validation – the cross-validation consistency of size of n. Then the optimal n among $2, 3, \dots, m$ is selected according to the two quantities (usually the two quantities give the same conclusion [57]. And the prediction error is the more important quantity, if the two quantities does not imply the same conclusion, we need to either use prediction error as the criteria or redesign the cross-validation process). This n is the final size of the model, and the optimal set for the chosen order forms the best multifactor model.

A strong selling point of MDR is that it can simultaneously detect and characterize multiple genetic loci associated with diseases. It searches through any levels of interaction regardless of the significance of the main effects. It is therefore able to detect high-order interactions even when the underlying main effects are statistically insignificant. However, as mentioned in [50], this "strength" is also a weakness: MDR can "only" identify interactions. If the real model has three loci and the effect is additive, MDR can only consider them all as a 3-factor interaction effect, although one can post-process the final model to further examine the 3-factor interaction in fact is additive.

In the situation with small number of factors, MDR can usually perform very well. However, when the number of factors become larger, MDR will quickly become computational infeasible. Another difficulty is empty cells in high-dimensional situation. For example, for a 5-dimension table, there are 243 genotypes cells, so it is very likely that many genotype cells may have zero or few observations. This will make the MDR model very unstable.

2.2.2 Classification Tree and Random Forests

Classification and regression tree (CART) is a popular method in statistical learning and data mining, and has been well adopted in GWAS. Since Random Forests with CART is already well-known in other fields, we will not explain this model in details but focus on vital aspects when the Random Forest model is applied in GWAS. Detailed introductions could be found in [20], and the application of Random Forest in GWAS could be found in [71].

Two central steps are involved in single tree (CART model) construction: partitioning and pruning. Training population in parent node is divided into two offspring nodes in order to achieve the best partition “score”. The term “score” could have different choices which serve as a measurement of how well certain covariate partitions the population in parent node. Popular choices include Gini index and cross-entropy. After recursively going through the partition steps and being stopped by certain criteria, a large tree is generated, with each node representing a partition rule. Usually this tree would over-fit the training data and a second step called pruning is needed. A quantity called maximum chi-square statistics can be applied in the pruning process, which shows how well the partition is not only for this single node but also for the whole sub-tree rooted at this node. So all nodes with maximum chi-square statistics below some threshold will be pruned because the whole sub-tree would not contribute much to distinguish population. The variables in the remaining nodes have the most impacts on distinguishing the training population. The tree method is very intuitive and has easy interpretation, however it has some instability issues, that is, a small change on an important variable could lead to a very different tree.

A much more popular tree-based method in GWAS is Random Forests (abbreviated as RF in this section), which is an ensemble of many CARTs. All SNPs – our potential biomarkers for certain disease – consists of the pool of variables from which subsets of variables are chosen to build up single CARTs. And also the training population for each CART is a random sample from the original population. Taking into account of the above two randomness, every CART in RF will probably choose out different set of variables, which can lead to the definition of important index for every variable, showing the impact of each variable to distinguish population. With the final predictive model, the important indexes also provide the significance order for all potential causal SNPs, which would be of great convenience in GWAS.

2.2.3 Bayesian Epistasis Association Mapping (BEAM)

BEAM method was first proposed in 2007 [72]. In the original paper, the authors reported that this algorithm could handle about 10^6 SNPs at the same time, which is quite impressive and very different from previous methods in GWAS. In a real data application with age-related macular degeneration (AMD), it was shown that BEAM method dealt with 96,932 SNPs from 146 individuals and ran for about 5 hours [72]. Below is a brief introduction of BEAM [72].

Suppose there are N_d cases and N_u controls genotyped at L SNP loci. Let case genotypes be $D = (d_1, d_2, \dots, d_{N_d})$ with $d_i = (d_{i1}, d_{i2}, \dots, d_{iL})$ representing genotypes of patient i at L SNP loci, and let control genotypes be $U = (u_1, u_2, \dots, u_{Nu})$ with $u_i = (u_{i1}, \dots, u_{iL})$ The L SNPs are partitioned into three groups: group 0 contains SNPs unlinked to the disease, group 1 contains SNPs contributing independently to the disease risk and group 2 contains SNPs that jointly influence the disease risk (interactions). Let $I = (I_1, \dots, I_L)$ indicate the membership of the SNPs with $I_j = 0, 1, 2$, respectively. Then all we have to do is to infer the set $j: I_j > 0$. Let l_0, l_1, l_2 denote the number of SNPs in each group ($l_0 + l_1 + l_2 = L$), and let D_0, D_1 and D_2 denote case genotypes of SNPs in group 0, 1 and 2, respectively.

Let $\Theta_1 = \{(\theta_{j1}, \theta_{j2}, \theta_{j3}): I_j = 1\}$ be the genotype frequencies of each SNP in group 1 in the disease population, (n_{j1}, n_{j2}, n_{j3}) be the genotype counts for cases of SNP j. Then the likelihood of D_1 is

$$P(D_1|\Theta_1) = \prod_{j:I_j=1} \prod_{k=1}^{3} \theta_{jk}^{n_{jk}}$$

Assuming a Dirichlet(α) prior for $(\theta_{j1}, \theta_{j2}, \theta_{j3})$, where $\alpha = (\alpha_1, \alpha_2, \alpha_3)$. Integrate out Θ_1 and obtain the marginal probability:

$$P(D_1|I) = \prod_{j:I_j=1} \left(\frac{\Gamma(|\alpha|)}{\Gamma(N_d + |\alpha|)} \prod_{k=1}^{3} \frac{\Gamma(n_{jk} + \alpha_k)}{\Gamma(\alpha_k)} \right)$$

Here $|\alpha| = \alpha_1 + \alpha_2 + \alpha_3$.

For the SNPs in group 2, they contribute to the disease through interactions. Thus we should take every combination of genotypes into consideration - there are totally 3^{l_2} possible genotype combinations with frequency $\Theta_2 = (\rho_1, \dots, \rho_{3^{l_2}})$ in the disease population. Let n_k be the counts for cases of genotype combination k. The likelihood of D_2 is

$$P(D_2|\Theta_2) = \prod_{k=1}^{3^{l_2}} \rho_k^{n_k}$$

Again, a Dirichlet(β) distribution is assumed for prior of $\Theta_2 = (\rho_1, \dots, \rho_{3^{l_2}})$. After integration, we get:

$$P(D_2|I) = \frac{\Gamma(|\beta|)}{\Gamma(N_d + |\beta|)} \prod_{k=1}^{3^{l_2}} \frac{\Gamma(n_k + \beta_k)}{\Gamma(\beta_k)}$$

With $|\beta| = \sum_{k=1}^{3^{l_2}} \beta_k$.

The remaining data D_0 consists of SNPs with the same distribution as in the control population. Let n_{jk} and m_{jk}, $k = 1, 2, 3$ denote the counts of genotype k at SNP j in D_0 and U. Also assuming Dirichlet priors with parameters $\gamma = (\gamma_1, \gamma_2, \gamma_3)$ for $\theta_j, j = 1, \dots, L$. Here $\theta_j, j = 1, \dots, L$ are the genotype frequencies in the control population. We integrate out Θ and obtain:

$$P(D_0, U|I) = \prod_{j=1}^{L} \left(\frac{\Gamma(|\gamma|)}{\Gamma(|\gamma| + \sum_{k=1}^{3} (n_{jk} + m_{jk}))} \prod_{k=1}^{3} \frac{\Gamma(n_{jk} + m_{jk} + \gamma_k)}{\Gamma(\gamma_k)} \right)$$

With $|\gamma| = \gamma_1 + \gamma_2 + \gamma_3$.

The posterior distribution of I is

$$P(I|D, U) \propto P(D_1|I) P(D_2|I) P(D_0, U|I) P(I)$$

With the prior of I being $P(I) = p_1^{l_1} p_2^{l_2} (1 - p_1 - p_2)^{L - l_1 - l_2}$. It is mentioned that this prior could be modified based on some knowledge about the disease.

The algorithm will use the last formula to give an estimation to the indicator I. In such a situation, the posterior distribution is known except for a normalizing constant, MCMC sampling method (more specifically the Metropolis-Hastings algorithm) is used. Each proposed move is accepted according to the MH ratio, which is just a ratio of gamma functions. What we get is the penetrance of each marginal effect and interaction effect (the distribution of I). There are some strategies making the whole algorithm more efficient mentioned in [72].

To test whether a set of SNPs is associated with the disease, the authors introduced a new statistics called B statistics. For a set M of k SNPs to be tested, the null hypothesis is that SNPs in M are not associated with the disease. The effect could be marginal and every interaction effects. Then:

$$B_M = ln\frac{P_A(D_M, U_M)}{P_0(D_M, U_M)}$$

Here, D_M and U_M denote the genotype data for the SNPs in M. P_0 and P_A are the likelihood function under null hypothesis and alternative hypothesis.

For the alternative model, $P_A(D_M, U_M)$ should be factored into $P_A(D_M)$ and $P_A(U_M)$. The authors also assumed that $P_A(U_M)$ and $P_0(D_M, U_M)$ follow a common distribution – an equal mixture of two distributions: one assumes independence among SNPs in M, $P_{ind}(X)$ which is given in Formula *; the other assumes saturated interaction among SNPs in M, $P_{join}(X)$ which is given in Formula *. X here could be U_M or (D_M, U_M). $P_A(D_M) = P_{join}(D_M)$ because we want to test a saturated interaction effect of the SNPs in M. So above all, the B_M could be written as:

$$B_M = ln\frac{P_{join}(D_M)(P_{ind}(U_M) + P_{join}(U_M))}{P_{ind}(D_M, U_M) + P_{join}(D_M, U_M)}$$

In [72], the authors also introduced a conditional B statistics which is used to test additional association effects conditioned on some known causal SNPs in M.

2.3 Neural Network Models Applied in GWAS

2.3.1 Existing Applications of Neural Networks in GWAS

Neural networks have been applied to genome-wide association studies in some papers. Tomita and others [65] have used neural networks to analyze the SNP data in a Childhood Allergic Asthma study. For each SNP, its three genotypes are coded as (0.1, 0.1), (0.1, 0.9) and (0.9, 0.9), respectively. In this form, the number of inputs in the ANN is twice as many as the number of SNPs. The SNP can also be coded as 0, 1 and 2. However, based on our simulation studies, the two different SNP coding show very little difference on the prediction errors. So in this thesis, we just use the original form of SNP data. Another feature of this paper is the usage of parameter decreasing method (PDM) for feather selection, which has been introduced in section 1.3.4. As more input variables (SNPs) being removed from the model, the remaining SNPs would be the "significant" factors for the outcome.

Later, Mutoh and others [46] also used the similar SNP coding techniques as Tomita in a study of Helicobacter pylori infection, but they employed a different method to select significant SNPs. Interestingly, they used a parameter increasing method to add SNPs into the neural network models, which can be viewed as an inverse process of PDM. Also, they have tried an exhausted search among all the combinations of three SNPs.

Both the above papers use the single layer neural network models and backpropagation method for the parameter estimation. Both give prediction accuracy rates around 70% to 80%, which seems to be very good compared with others. Some other researches like Okut and others' [48] apply neural network models in Bayesian frame in order to achieve better computational properties, but again based on the single layer neural network.

2.3.2 Our Novel Structure of Neural Network Models

One limitation for the single layer neural networks is that they did not take into consideration the fact that SNPs with a LD block are highly correlated with each other, but are rarely correlated with SNPs outside of the block. Ignoring this information may lead to less efficient detection of causal SNPs. This motivates us to construct a double-layer neural network, referred as DNN hereafter, to analyze SNP epistasis effects, which utilizes the information within LD blocks.

One big advantage of DNN is to avoid redundant SNP-SNP interactions within the LD blocks while still retaining informative SNPs in the model. For example, suppose we have three SNPs A, B and C; SNPs A and B belong to one gene and SNP C belongs to another gene that interacts with the first gene; then, due the LD between A and B, we can predict that the A-C interaction would be very similar to the B-C interaction, and only one interaction may be significant because of the masking effect between them; however, ignoring either interaction may introduce bias to the model and subsequently be less powerful. To circumvent this, we proposed the DNN model, in which the first layer is used to "summarize" the SNP information in one gene or in one LD block, and the second layer is used to analyze interactions at gene level.

Additionally, in genome-wide association studies, some environmental variables may also be considered into the models, such as age, ethnicity and lifestyle. These factors may be added as "neurons" in the second layer.

Figure 9 illustrate the DNN model structure we construct. Suppose there are P SNPs, which serve as inputs of DNN and are divided into N groups based on the LD information among them. Then N is the number of "normal" neurons in the first hidden layer. The first layer also contains N_1 "abnormal" neurons which represent environmental factors or other covariates. We call them "abnormal" because they are not generated from the input layer. The second hidden layer contains M neurons which are summarized from the first hidden layer information. In this application, we consider a case-control study,

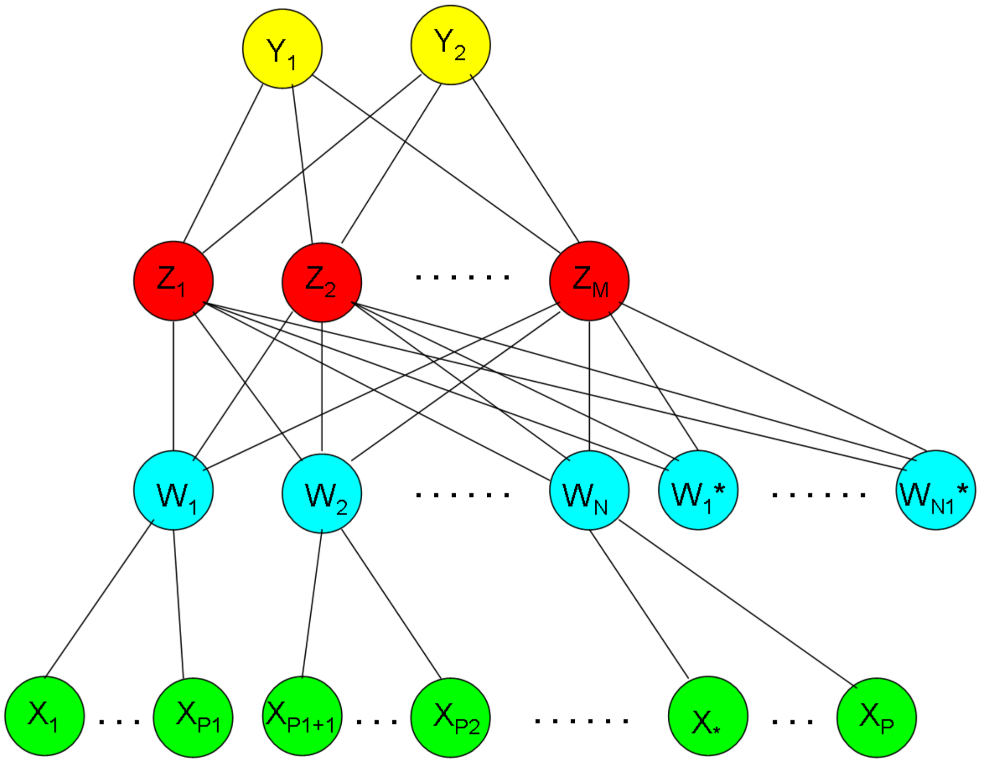


Figure 9. Double layer neural network model with local connections from input layer to the first hidden layer

The input X neurons are locally connected to W neurons (as in formula (2.3.1)). Input neurons $W*$ are directly connected to Z neurons. This structure is designed specifically for GWAS to consider interaction effects in gene level. The partition of X neurons is determined by linkage-disequilibrium.

which has a binary outcome, i.e. 0 and 1, corresponding to two outcomes, Y_1 and Y_2.

The formulas to describe relations between adjacent layers are similar to those in (1.2.3)* - (1.2.5)* in section 1.3.3.1, and they are expressed as:

$$W_{iu} = \sigma\left(\sum_{p=s_u}^{n_u} \alpha_{up} * X_{ip}\right), \text{for } u = 1, \dots, N \tag{2.3.1}$$

$$Z_{im} = \sigma\left(\sum_{n=1}^{N+N_1} \beta_{mn} * W_n\right), \text{for } m = 1, \dots, M \tag{2.3.2}$$

$$\hat{y}_{ik} = g_k\left(\sum_{n=1}^{N} \theta_{kn} * Z_{in}\right), \text{for } k = 1, \dots, K \tag{2.3.3}$$

Notice that Formula (2.3.1) is different from (1.2.3)*, because neuron W_{iu} only gather information from X_{is_u} to X_{in_u} as introduced in Section 1.2.5 of local connections. Then the first derivatives for all parameters could be calculated as:

$$\begin{aligned}\frac{\partial R_p(\Omega)}{\partial \theta_{km}} &= \sum_i \frac{\partial R_p(\Omega)}{\partial \hat{y}_{ik}} \frac{\partial \hat{y}_{ik}}{\partial \theta_{km}} = \sum_i \frac{\partial R_p(\Omega)}{\partial \hat{y}_{ik}} g_k'\left(\sum_{m=1}^{M} \theta_{km} * Z_{im}\right) Z_{im} \\ &= \sum_i \epsilon_{ik} Z_{im}\end{aligned} \tag{2.3.4}$$

With: $\epsilon_{ik} = \frac{\partial R_p(\Omega)}{\partial \hat{y}_{ik}} g_k'(\sum_{m=1}^{M} \theta_{km} * Z_{im})$ (2.3.5)

$$\begin{aligned}\frac{\partial R_p(\Omega)}{\partial \beta_{mn}} &= \sum_i \frac{\partial R_p(\Omega)}{\partial Z_{im}} \frac{\partial Z_{im}}{\partial \beta_{mn}} = \sum_i \sum_{k'} \epsilon_{ik'} \theta_{k'm} \sigma'\left(\sum_{n=1}^{N+N_1} \beta_{mn} * W_{in}\right) W_{in} \\ &= \sum_i \eta_{im} W_{in}\end{aligned} \tag{2.3.6}$$

With: $\eta_{im} = \sum_{k'} \epsilon_{ik'} \theta_{k'm} \sigma'(\sum_{n=1}^{N} \beta_{mn} * W_{in})$ (2.3.7)

$$\frac{\partial R_p(\Omega)}{\partial \alpha_{up}} = \sum_i \frac{\partial R_p(\Omega)}{\partial W_{iu}} \frac{\partial W_{iu}}{\partial \alpha_{np}} = \sum_i \sum_{m'} \eta_{im'} \beta_{m'n} \sigma'\left(\sum_{p=s_u}^{n_u} \alpha_{up} * X_{ip}\right) X_{ip} \tag{2.3.8}$$

Formula (2.3.4) – (2.3.8) are quite similar with (1.3.9) – (1.3.13), only with (2.3.8) slightly different with (1.3.13) because of local connections. Then the vector of all the parameters Ω is updated through:

$$\Omega_{r+1} = \Omega_r - \gamma_r \cdot \nabla_r \tag{2.3.9}$$

γ_r is the learning rate, and is usually set to: $\gamma_r = 1/r$. ∇_r is the derivatives of all parameters (shown in Formula (2.3.4), (2.3.6) and (2.3.8)) at rth step. More detailed discussions could be found in Section 1.3.3.1.

2.4 Simulation Studies and Real Data Application

In this section, we performed extensive simulations to examine the statistical properties of our method, and then applied it to a real dataset. To mimic the real data setting, our simulation studies are designed based on a real data, called COGEND (Collaborative Genetic Study of Nicotine Dependence). In this dataset, there are 2022 observations (908 in control group and 1114 in case group). The covariates include 216 candidate SNPs that were identified from previous linkage studies [39], and three patient-level characteristics: age, gender and color (representing ethnicity). Age is a continuous variable, with gender and color being binary variables. Each SNP loci is coded as 3 possible values: 0, 1 and 2. The phenotype of interest is binary, i.e. whether a patient is addicted to Nicotine. The use of our DNN model will focus on identifying genetic factors of Nicotine dependence based on this group of candidate SNPs, and improved understanding of these factors may suggest novel, powerful strategies to reduce or eliminate nicotine dependence.

We first divided the 216 SNPs into different linkage disequilibrium groups. The pairwise SNP LD measures – the D parameters (as introduced in section 2.1.1) – were calculated through maximizing the likelihood function of the whole population based on the probabilities in Table 1. The byproducts of this process will give p_a and p_b in Table 1, which are the MAFs for each SNP locus. The pairwise LD values are shown in Figure 10, in which there are totally 17 LD blocks based on their within-block correlations. Our DNN model will summarize block level information in the first hidden layer and then test any interactions between block level information could contribute to determine the outcome.

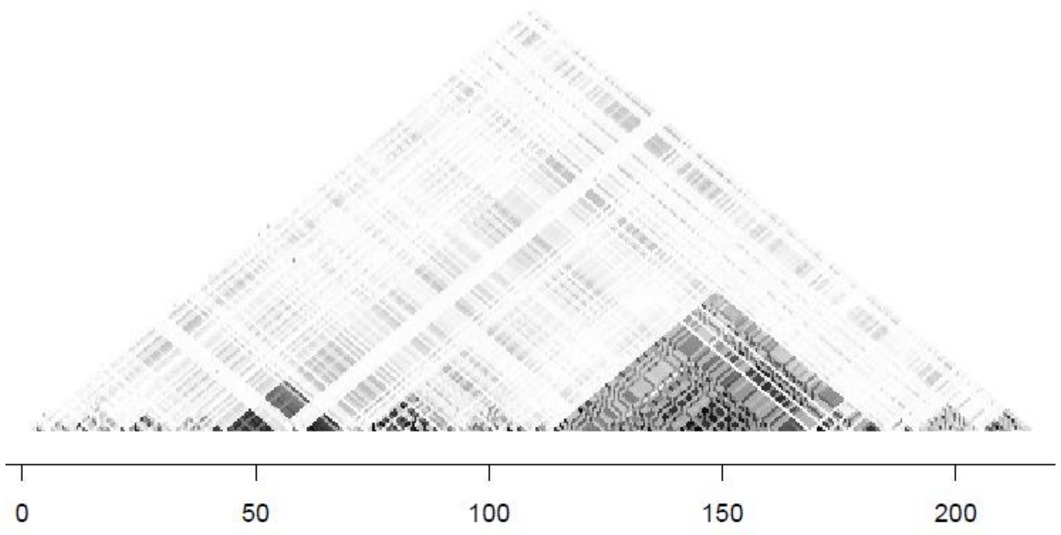


Figure 10. LD pattern among 216 SNPs in COGEND study

For each point in the figure, we draw two lines with slopes equal to 1 and -1 from that point. Suppose the lines intersect with x-axis on coordinates A and B, then the point represents the *D* parameter between SNPs A and B. Black point means a strong correlation and white point means a weak correlation or even independence. Based on this figure, there are 17 groups of SNPs.

In the following, we will design corresponding simulation studies to apply our DNN model on different settings and compare it with the random forest model to benchmark its performance. After the simulation study, our DNN model with PDM (Section 1.3.4) will be applied on the whole COGEND data to identify the most possible LD blocks that contain causal SNPs.

2.4.1 Overall Simulation Procedures

(1) We adopted two LD blocks in Figure 10 in our simulations: block 1 contains SNP 44 to SNP 54 and SNP 63 to SNP 69; block 2 contains SNP 74 to SNP 113. Here we combine SNP 44~54 and 63~69 into one block because the two sets of SNPs have strong LD. In this simulation study, no patient characteristics are included. So there would be totally 58 SNPs in this data.

(2) In each block, a few SNPs are set to be the causal SNPs. Causal SNPs from the same block interact with each other to determine a block level information and the two block level information will again interact to determine the outcome – Nicotine dependence or not. The two interaction patterns can be altered to generate several different patterns, and will be illustrated with more details later.

(3) After the outcome variable generated, causal SNPs are removed from the simulated dataset to mimic the case that causal SNPs may not be sampled in real data. We expect the two models have the ability to detect causal SNPs' existence even the real causal SNPs are not genotyped and presented in the data set (as Figure 7b).

(4) Data sets with different sizes (100, 200, 500 and 1000) are sampled from COGEND data using the schema in previous step. During the sampling, we try to make the outcome to be balanced, i.e. half of patients are Nicotine dependent and half are not.

(5) For each dataset generated, a permutation test is performed to compare whether the fitted DNN model has significantly lower value for error function. If so, we think there are causal SNPs existing in the two blocks.

(6) Repeat the above 5 steps for 200 times and count the number of times we correctly detect the existence of causal SNPs N, then calculate the power as: $N/200$.

(7) Random Forest model is also applied on the same data set and calculate the power of detecting the causal SNPs.

2.4.2 Simulation Settings

Two sets of studies with different interactive patterns are conducted. We name them as "2+1" and "2+2". In the "2+1" pattern, there are two causal SNPs which have interactive effect on the outcome from the first LD block and one causal SNPs which has marginal effect on the outcome from the second LD block. In the "2+2" pattern, there are two causal SNPs in the first LD block and also has two causal SNPs in the second LD block, with the same interaction pattern as that in "2+1". Detailed settings are given in the following two sections.

2.4.2.1 Pattern "2+1"

Causal SNPs from block 1 are set to be SNP 50 and SNP 65. Causal SNP from block 2 is SNP X, and we have tried different positions for SNP X: 99, 97, 95, 113, 94, 90, 74~79, 81~84, 86 or 91. The block level information are defined as follows:

$$block\ 1 = \begin{cases} 1, & if\ SNP\ 50 = 1\ or\ 2\ and\ SNP\ 65 = 0\ or\ 1 \\ 0, & otherwise \end{cases} \tag{2.4.1}$$

$$block\ 2 = \begin{cases} 1, & if\ SNP\ X = 2 \\ 0, & otherwise \end{cases} \tag{2.4.2}$$

Then the phenotype is determined as:

$$P(phenotype = 1) = \begin{cases} \beta, & if\ block\ 1 = 1\ and\ block\ 2 = 1 \\ \epsilon, & otherwise \end{cases}$$

Phenotype = 1 means the patient is Nicotine dependent. So in this simulation study, when block 1 = 1 and block 2 = 1 are true, they have synergistic effect on causing Nicotine dependence. Parameter β is called the penetrance and represents the proportion of genotype causes for Nicotine dependence. Different values for $\beta: 0.3, 0.4, 0.5, 0.6, 0.7$ will be tested in the simulation study. Parameter ϵ is set to be 0.01 as a random cause for Nicotine dependence, indicating that even a patient does not have the genetic cause, he or she still has a small probability to be Nicotine dependent.

2.4.2.2 Pattern "2+2"

The only difference of "2+2" pattern from previous pattern is that: block 2 has two causal SNPs – SNP 80 and SNP X. We also have tried different positions for SNP X: 99, 97, 95, 113, 94, 90, 74~79, 81~84, 86 and 91. The block level information are defined as:

$$block\ 1 = \begin{cases} 1, & if\ SNP\ 50 = 1\ or\ 2\ and\ SNP\ 65 = 0\ or\ 1 \\ 0, & otherwise \end{cases} \tag{2.4.3}$$

$$block\ 2 = \begin{cases} 1, & if\ SNP\ 80 = 0\ or\ 1\ and\ SNP\ X = 0\ or\ 1 \\ 0, & otherwise \end{cases} \tag{2.4.4}$$

The other parts are exactly the same with "2+1" pattern:

$$P(phenotype = 1) = \begin{cases} \beta, & if\ block\ 1 = 1\ and\ block\ 2 = 1 \\ \epsilon, & otherwise \end{cases}$$

With $\beta = 0.3, 0.4, 0.5, 0.6\ or\ \ 0.7$ and $\epsilon = 0.01$.

2.4.3 Model Structure for Simulation Studies

The model structure used in our simulation studies is a little different from Figure 9, because here in the simulation studies, there are no environmental factors. We would like to formulate our double layer neural network models in more details here, with model structure shown in Figure 11.

From Section 2.4.1, block 1 contains 18 SNPs and block 2 contains 40. And after determining the causal SNPs and phenotypes, we eliminate the causal SNPs from our

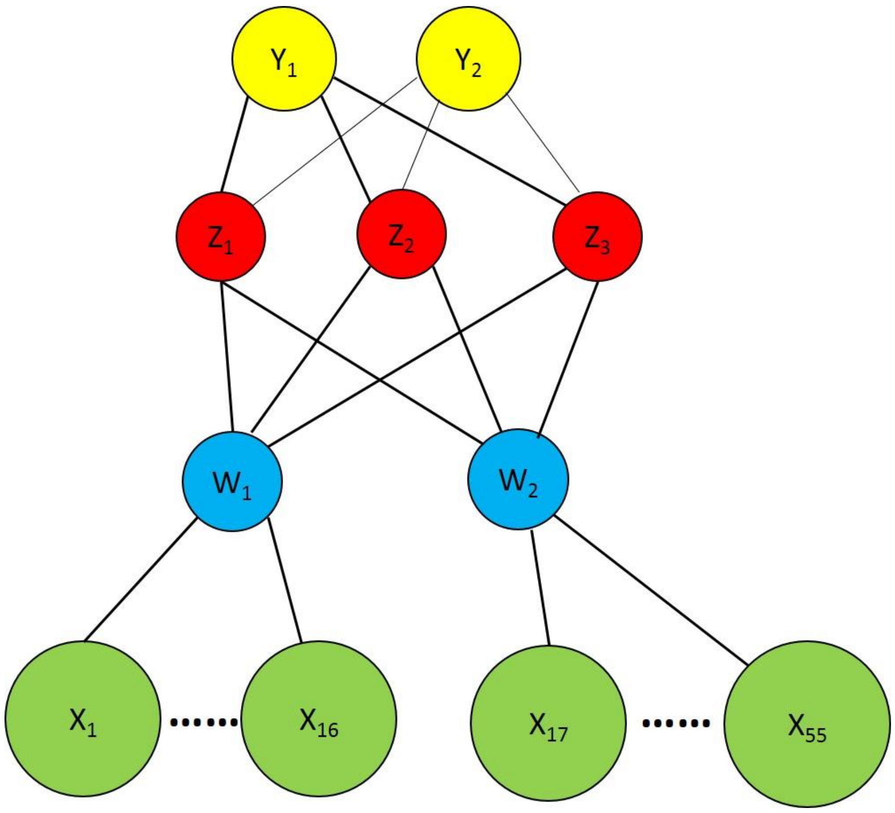


Figure 11. Neural network structure for simulation studies.
There are two blocks – SNP groups in the input layer so that the structure has two neurons in the first hidden layer. Then three neurons in the second layer. It has two outcome neurons because the outcome variable is binary.

generated data set, which means finally block 1 contains 16 SNPs, block 2 contains 39 SNPs for "2+1" pattern and 38 SNPs for "2+2" pattern. The two patterns make little difference in model structure, so without loss of generality, we use “2+1” pattern as our example. For simplicity, X_1 to X_{16} represent SNPs from block 1 and X_{17} to X_{55} represent SNPs from block 2. Each of the X has discrete value: 0, 1 or 2 for genotype aa, Aa and AA respectively.

X variables are the input neurons and are linked to the first hidden layer and then to the second hidden layer through the following formulas:

$$W_1 = \sigma\left(\sum_{p=1}^{16} \alpha_p * X_p\right) \; and \; W_2 = \sigma\left(\sum_{p=17}^{55} \alpha_p * X_p\right)$$

$$Z_m = \sigma\left(\sum_{n=1}^{2} \beta_{mn} * W_n\right) \; with \; m = 1, 2, 3$$

Then the Z neuros are connected with the two outcome neurons through:

$$T_k = \sum_{m=1}^{3} \theta_{km} * Z_m \; with \; k = 1, 2$$

$$Y_1 = \frac{e^{T_1}}{e^{T_1} + e^{T_2}} \; and \; Y_2 = \frac{e^{T_2}}{e^{T_1} + e^{T_2}}$$

Here Y_1 and Y_2 represent the probability of phenotype = 0 (not being Nicotine dependent) and phenotype = 1 (being Nicotine dependent). If $Y_2 > 0.5$, we predict that the corresponding patient is more likely to be Nicotine dependent.

2.4.4 Simulation Studies Results

For each choice of SNP X, we have one figure showing the powers calculated for DNN and Random Forest models using different sample sizes (1000, 500, 200 and 100) and at different penetrance. Figure 12 and Figure 13 shows the simulation results for pattern “2+1” and pattern “2+2” correspondingly.

Each sub-figure has y label named with a SNP number which represent the SNP X in our simulation settings. Different colors in one sub-figure are for different sample sizes: red for 1000, green for 500, blue for 200 and black for 100. Solid lines are for DNN models and dashed lines are for random forest models. Each point on the line represents the power for specific model detecting the existence of causal SNPs at each penetrance value.

All settings show that larger penetrance and larger sample sizes lead to better powers, as expected. For larger sample sizes (1000 or 500), DNN models always have better performance than Random Forests (implemented using Random Jungle software [60] [41]) in terms of power. For small sample sizes (200 or 100), the two models may have better performance in different choices for SNP X.

We also find that for pattern “2+1”, both models almost have little powers in detecting the significance for X = 90 and 113. For the pattern “2+2”, both models have no powers for X = 74, 77, 78, 94, 97, 82 and 86. We suspect this phenomenon is related to the characteristics of the specific SNPs – whether the causal SNPs have high correlations with other SNPs within the second LD block and even the correlations with SNPs from the first LD block. Also we notice that the “problematic” SNPs for pattern “2+1” and “2+2” are very different and pattern “2+1” has only two of such kind of SNPs which might because the marginal effect is easier to be detected than the interactive effect. The above suspicions need more simulation studies to validate.

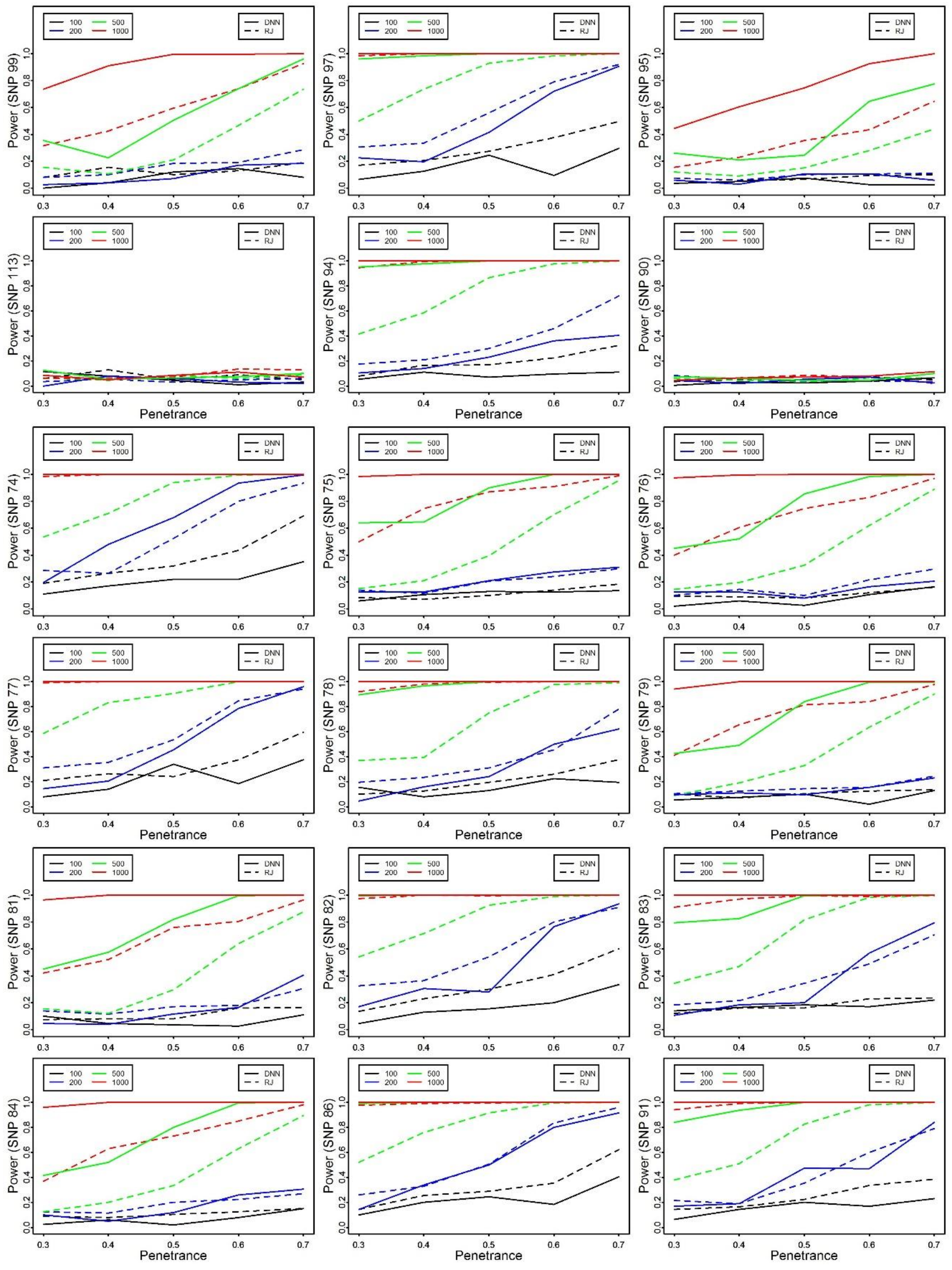


Figure 12. Simulation study results for pattern "2+1".
Each line represents power curve along with penetrance (x-axis) for one specific method (solid line for DNN and dashed line for Random Forests) at a specific sample size (four colors).

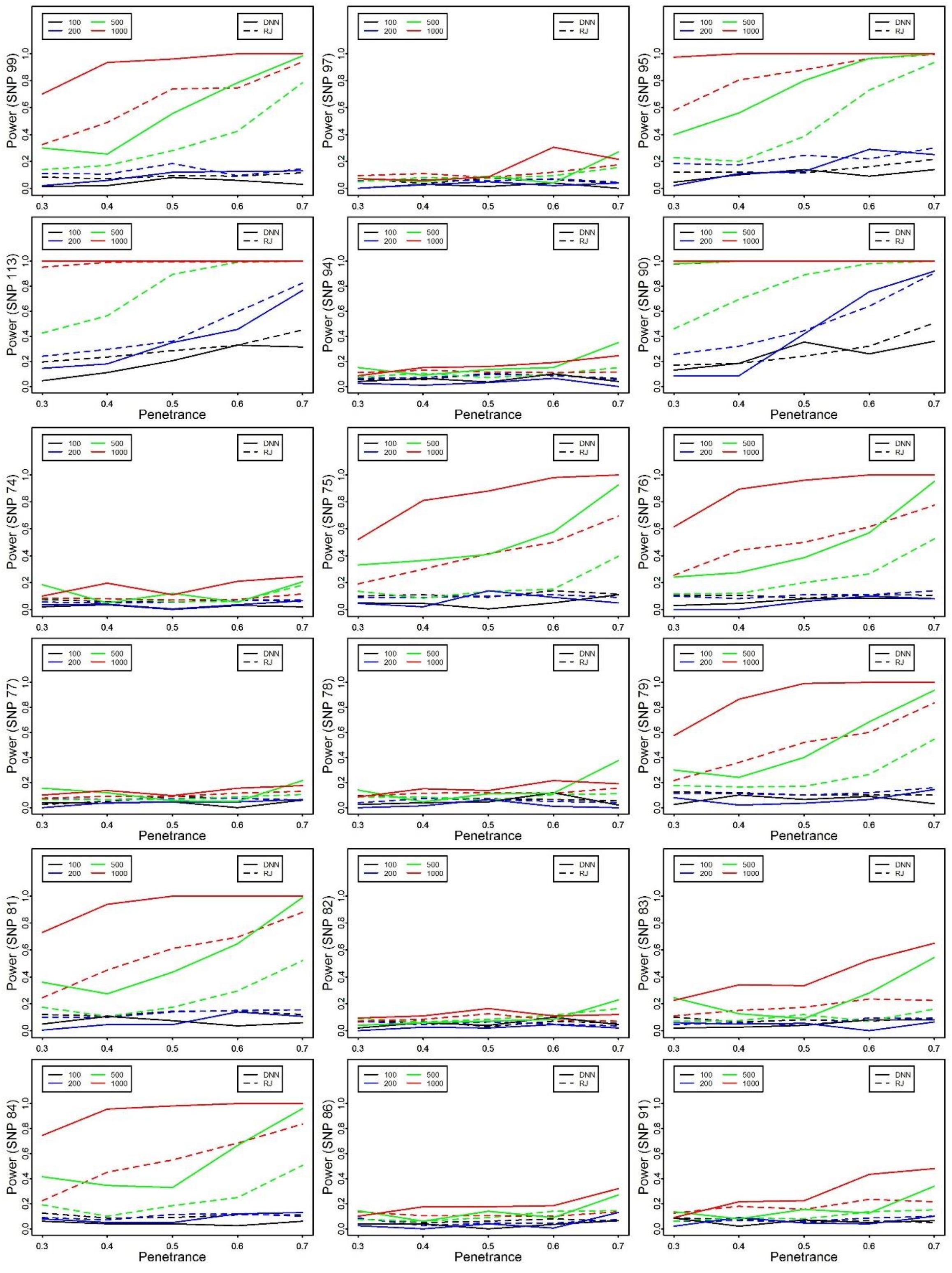


Figure 13. Simulation study results for pattern "2+2".

Each line represents power curve along with penetrance (x-axis) for one specific method (solid line for DNN and dashed line for Random Forests) at a specific sample size (four colors).

2.4.5 Real Data Analysis

The goal of this analysis is to identify which LD blocks may contribute significantly to the phenotype – the blocks have the largest impact on whether the patient will be Nicotine dependent or not.

2.4.5.1 Model Structure and Parameter Decreasing Method

Out DNN structure for this real data analysis is shown in Figure 14, similar to that in Figure 9, and the parameters are set as: $P = 216$ because COGEND data has totally 216 SNP loci to analyze; $N = 17$ because our preliminary analysis results (in Figure 10) show that the 216 SNPs could be divided into 17 LD blocks; $N_1 = 3$ because COGEND data has 3 environmental factors: age, gender and ethnicity. So we have 20 ($= N + N_1$) neurons in the first hidden layer; $M = 6$ (6 neurons in the second hidden layer) based on our trials and the number of neurons in the first hidden layer; $K = 2$ because the

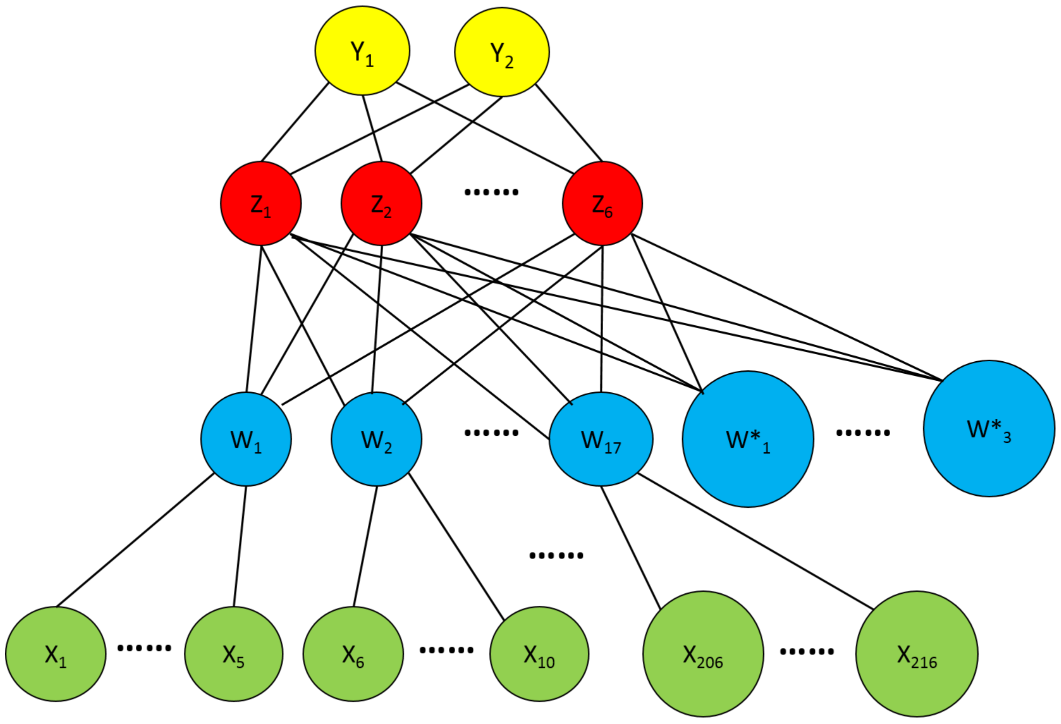


Figure 14. DNN structure for COGEND data analysis
216 input variables (SNPs) are locally connected to 17 W neurons based on results shown in Figure 10. W neurons with 3 W^* neurons (3 environmental factors included in COGEND data) are fully connected to Z neurons.

phenotype is a binary variable (Nicotine dependent or not); the penalization parameter λ is pre-specified to be 0.5 based on our trials.

The relationships between each two layers are shown in Formula (2.3.1) – (2.3.3) and parameter estimation process (using back-propagation algorithm) is shown in Formula (2.3.4) – (2.3.9). To reduce the local minimal problem of back-propagation algorithm, we try 50 randomly selected initial values for parameter set Ω. Each initial value will converge to a local minimal (might be the global minimal) of the error function $R_p(\Omega)$ and the estimation corresponding to the smallest $R_p(\Omega)$ value will be chosen as our final parameter estimation.

PDM (as introduced in Section 1.3.4) is applied on the above DNN model to select the most important LD blocks (the W neuron in Figure 14): (1) Initially, we have the full model as shown in Figure 14 and the corresponding parameter estimations for $R(\Omega)$ (Note that $R(\Omega)$ value is different from $R_p(\Omega)$ and shows how well our model is fitted to the data, refer to Formula (1.3.2) and (1.3.3)). Then we try 17 reduced DNN models – each with one W neuron and its related X neurons being excluded. The reduced DNN models would usually have a higher $R(\Omega)$ value and the one with least $R(\Omega)$ value shows that the excluded W neuron and related X neurons have the least impact on the outcome variable. So the W neuron (also the corresponding X neurons) with least $R(\Omega)$ value is removed to obtain a model, which will serve as full model in the next step. (2) In the next step, we try another 16 reduced DNN models to filter out the W neuron that has the least impact on the outcome. Eliminating that W neuron (also the corresponding X neurons) leads to our model after the second elimination. (3) Repeat the process in (2) until there are two W neurons left in the model, which represent the two LD blocks having the most impact on the outcome.

2.4.5.2 Real Data Analysis Results

Figure 15 shows the analysis results. Each point on the curve represents one fitted model with x-axis value showing the number of eliminated LD blocks. And the value above each point means which LD block is excluded in this step. For example, the point at x-axis value 0 means the full model which has all 17 LD blocks information. The point at x-axis value 1 means the reduced model which has 16 LD blocks as the 5th LD block is removed due to having the least $R(\Omega)$ value. The point at x-axis value 2 means the reduced model which has 15 LD blocks as the 5th and 11th LD blocks are removed. As fewer LD blocks are included in the model, the $R(\Omega)$ value almost goes up as we expected. Several unexpected drop down may because we do not reach the global minimal point for the previous model or both models. Increasing the number of trials for initial values may help reduce this phenomenon.

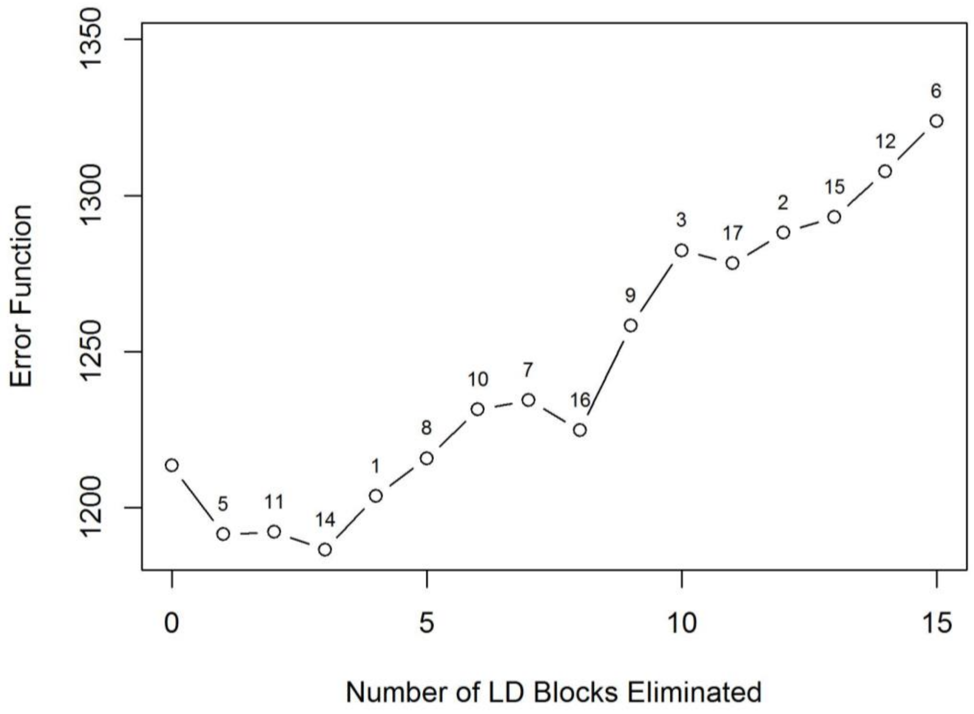


Figure 15. Eliminated LD blocks with corresponding error function values.
Y axis is $R(\Omega)$ values for fitted DNN model. Each point on the curve represents one DNN models with a number above the point shown the eliminated LD block comparing to the previous point's model. After 15 eliminations, LD block 4 and 13 remain in the final model.

The remaining LD blocks are the 4th and 13th LD blocks which are believed to have the most impact on Nicotine dependence. The 4th LD block consists of SNPs from gene CHRNG on chromosome 2 and the 13th LD block consists of SNPs from gene IREB2, AGPHD1, PSMA4, CHRNA5, CHRNA3 and CHRNB4 on chromosome 15. It's also worth to notice that the 6th LD block has the third most impact on Nicotine dependence based on our results in Figure 15 (the point at x-axis value equal to 15). This LD block consists of only one SNP loci (rs2276560) on chromosome 2, and this SNP belongs to no genes. Comparing that the 4th LD block has 6 SNPs and the 13th LD block has 72 SNPs, we think at single SNP level, SNP rs2276560 might have the most impact on Nicotine dependence.

2.5 Discussions and Future Work

Our studies have demonstrated that our DNN model can have better power in detecting causal SNPs than the Random Forest. For future works, more thorough simulation studies should be carried out to further examine the properties of the DNN:

- More methods need to be included for comparison, such as MDR, BEAM and penalized logistic regression [50]. PLINK is another popularly accepted software in GWAS, providing the analysis on GWAS data through linear or logistic regression and also other traditional statistics and tests applicable in GWAS [53].
- In the simulation studies, we observe that both DNN and Random Forests have very little powers when causal SNPs are in some specific positions (summarized in Section 2.4.3.3). We suspect that it is because different causal SNPs may have different correlations with other SNPs nearby. More simulation studies need to be designed to address this question.
- Our simulation studies only focus on two LD blocks data set, which is much smaller than common data sets in GWAS. Like the COGEND data set has already contain 17 blocks. So designing a more general framework to detect the existence or even the positions of causal SNPs by utilizing our designed DNN model structures would be vital in the future. Among the two tasks, determining the positions of causal SNPs is more difficult and also more meaningful to GWAS. Since our DNN model summarizes LD block information in the first hidden layer (as in Figure 9), we may apply the parameter decreasing method (Section 1.3.4) to select among all neurons in the first hidden layer (LD blocks). The last neurons remaining in the model would have highest possibilities to contain the causal SNPs.

- For the real data analysis, PDM is currently considered as the best practice for factor selection in our DNN model frame. However, the procedure requires the fitted DNN model to be stable, that is, repeating the parameter estimation process will give similar model estimations and stable error function values. This may be an issue for the DNN model, as the global minimal of the error function is usually not achievable, which may be the reason that in Figure 15, there are some drop-downs of $R(\Omega)$ values when one LD block is eliminated. So a more stable procedure, such as averaging several DNN fittings, needs to be developed in the future.
- Another aspect about real data analysis is the computational time for testing among the whole genome. For our DNN model to test on COGEND data (216 SNPs from 2022 individuals with only 1 trial for initial values in parameter estimation) using parameter decreasing method, it takes less than one hour. MDR suffers a lot for testing higher interactive effects and BEAM is reported to analyze among 96,932 SNPs from 146 individuals for about 5 hours [72]. So to further evaluate the applicability of our proposed method, we also need to compare the computational burdens among widely applied methods.

CHAPTER 3 HIGH FREQUENCY TRADING USING NEURAL NETWORK MODELS

3.1 Review of Neural Network Models in Finance

The predicting power of neural networks has been noticed by financial industry for quite a long time, and has been used successfully for price forecasting problems. [17] provided a good review for applications of NN models in stock markets for researches published during 2005 to 2010. Apart from the researches summarized in [17], some new frameworks of NN have been proposed recently, such as fuzzy NN [14], partially connected NN [4], hybrid of non-linear ICA and NN [11], Legendre NN with random time strength function [40], hybrid system built upon recurrent NN structure [55] and constructive NN models with specific training algorithm [28]. In these studies, the neural network based approaches have been compared with many other traditional baseline methods including multiple regression, ARIMA model, and the results supported the better performance of neural network.

Some researches have also compared neural network models with other machine learning models in stock price prediction: [30] compared single-layer NN with SVM in stock prediction, and concluded that NN was better than SVM. [51] compared four machine learning models applied in stock prediction: Random forests, SVM, NN (single layer) and naive-Bayes. Random forests was claimed to have the best performance and naive-Bayes was the worst.

By reviewing the listed studies, we notice that recent applications of neural networks in finance mainly focused on further studies of newly designed architectures and hybrid systems, some of which were quite successful. Along this line of research, in this thesis, we will also propose a new neural network structure for stock market prediction. For this purpose, proper input variables are very important. Based on the literatures, there are mainly four types of variables used in the prediction models, which are discussed as follows:

1. Historical price and volume. Historical prices and volumes are the raw market data. The historical prices recorded a stock's price movement in the past, and volume

shows how active a stock was, reflecting the force of demand/supply behind the price movement. The rationales to include the historical data as inputs are based on the assumptions that (1) in financial markets, history may repeat itself, and (2) observations in a time series are not independent so past data can provide information to the future. It is usually difficult to determine how many past time points should be included, that is, short history may lower the prediction accuracy due to the information loss, while long series would include too much noises. Usually 10-20 historical time points should be enough.

2. Technical indicators: Technical indicators are mathematical transformations of the raw price and volume data, often including up and down volume, advance/decline data and other inputs. These indicators help preprocess the raw data into clearer formats. For example, it would be easier to assess whether a stock is trending using certain technical indicators such as moving averages, relative strength index, and MACD (moving average convergence/divergence). Technical indicators should also be included, as a complementary to the raw data.
3. Economic indicators: Economic indicators measure the economic activity, and are the summary of current economic performance. Macroeconomic environment likely impacts most of the stocks in the same way, that is, the majority of stocks advance in bull markets and decline in bear market. Therefore, it is usually useful to include some of the economic indicators to gauge the overall market status.
4. Periodical signals: It has been long observed that markets experience cycles and there are periodical signals that repeatedly occur as patterns in price movement. For example, The January effect hypothesizes that there is a seasonal anomaly in the financial market where stocks' prices increase in the month of January more than in other months. This type of calendar effect may create an opportunity for investors and may be included as model inputs.

Some of these inputs will be included in our model later.

3.2 Forecasting in Financial Data

3.2.1 Challenges in Low Frequency Data

Forecasting is such a major problem in financial markets, it has been intensively studies in the past decades with low-frequency data (daily, monthly, and yearly) [35] [31] [5], and the major group of methods is based on technical analysis that formulates predictions using the time series of prices [6] [49] [27] [66] [19].

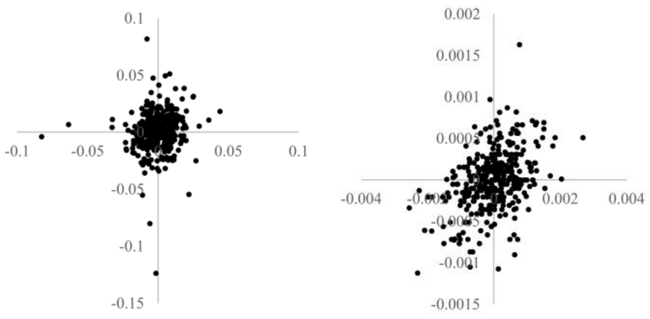
There are also some challenges exist in traditional forecasting models on financial data. For instance, for distributions of returns and volatilities, while they significantly affect the performance of econometric models, they are typically very difficult to be determined and inconsistent forecasting may arise from different assumptions. Gaussian models was popular for its simplification, but it is rarely satisfied and has poor fitting of real-world data [1]. Other distributions discussed in recent work include stable Levy distribution [67] [7], t-distribution [68], and power law distributions [63] [64]. No matter what distributions were used, these assumptions are usually compromises to the real-world data and thus biased. Another important issue is how to measure correlations in financial markets, as different methods affect the performances of forecasting models quite significantly. Pearson correlation has been widely used for its low computational cost; however, it correctly measures the association between variables only when the samples are normally distributed. Although other correlation measures, such as Kendall's $\boldsymbol{\tau}$ [47] [73] and Spearman correlation [44], have been used in non-Gaussian cases, their computational costs are usually higher and it is unclear that they are not optimal either.

3.2.2 Challenges in High Frequency Data

Recently, high-frequency data and the related high frequency trading [15] strategies have opened new ways for research and practice in finance, especially for problems in stock market forecasting. While some of the methods in low frequency data are still applicable to the high-frequency setting, most are usually limited in handling the more complex data structure associated with them. And among the limited published researches, most of them have used minute level data to test model performances [2].

Apart from the challenges mentioned in Section 3.2.1, high frequency data has other important properties which may impact model performances dramatically.

Comparing to low-frequency data, correlations between stocks in high-frequency data are usually non-ignorable, and therefore the dependence structure among stocks is more important in explaining the market movement in high-frequency data. Figure 16 shows a comparison of correlation between the returns of IBM Inc. and Apple Inc. in daily data and 5-min data. Figure 16(a) plots the overall relationship in daily returns from January 1, 2013 to June 1, 2013, and Figure 16(b) shows the relationship of them in 5-min data (June 3, 2013). Obviously, the dependence between the two stocks is much more significant in the 5-minute data than that in the daily data. Although identifying dynamic dependence in high-frequency data between two stocks is not difficult, modeling the dependence structure among the whole stock market and using them for market forecasting is complicated. However, it is an important problem to solve, as from the perspectives of market policymakers, regulators and private investors, there is a great interest in being able to forecast and simulate the market movements.



(a) Daily Data (b) 5-min Data

Figure 16. Correlations in different time scale.

Picture a and b show the correlations between Apple and IBM in two different time scales – daily level and 5 minute level. The 5 minute level data shows a much stronger correlation.

3.3 Forecasting High Frequency Data Using Neural Networks

In this study, we propose a new framework for market forecasting with high-dimensional high-frequency stock data , and focus on designing a novel double layer NN (DNN) structure that does not require specific distributions assumptions and also avoids the explicit calculation of correlations. The main ideas can be briefly summarized as follows:

- For a target stock, we first examine its dependence with the rest of stocks in the market as well as their volatility information. We try to capture these important information among the high-dimension data as the initial steps.
- We further incorporate these information with different types of technical indicators (TIs). The TIs at the levels of individual stocks and market are included into different layers of the neural network.
- For the TIs, we focus on the moving average (MA), which indicates the trend and momentum of a stock, and the advance/decline (AD) lines that show the strength of a current trend and its likelihood of reversing.

In the following, our proposed model will be described in details.

3.3.1 Overview of the Framework

Our overall goal is to forecast the price movement of individual stocks within one exchange, i.e. S&P 500, based solely on their historical data. Let $S_{t,j}$ denote the close price of stock j $(i = 1, \dots, J)$ for the 5-minute interval at time t $(t = 1, \dots, T)$, and let $X_{t,j} = \frac{S_{t+1,j} - S_{t,j}}{S_{t,j}}$ be the relative change from time t to $t+1$. The goal here is to predict $X_{t+1,j}$ – the close price for next 5-minute interval based on $S_{i,j}$ $(i = 1, .., t; j = 1, \dots, J)$.

To make the one-step prediction on the return $X_{t+1,j}$ for a specific stock j, we construct a double layer neural network (DNN) model structure as shown in Figure 17. Our design is quite different from the most commonly used NN structures in financial forecasting, as it allows us the flexibility of incorporating input variables with different features into the model in a hierarchical way. As shown in later sections, this design can not only improve the model performance, but also dramatically reduce the number of parameters, leading to faster implementation.

The bottom layer consists of two types of inputs: The $Xs = \{X_{t,j'}: j' = 1, \dots, j-1, j+1, \dots, 500\}$ and the $Qs = \{Q_{k,j}: k = t-9, \dots, t\}$, where index j is for the stock of interest.

The Xs include the price information for the rest 499 stocks in S&P 500, excluding stock j, within the current 5-minute interval at time t, and the Qs include information of the intra-interval proportions for stock j in 10 most recent 5-minute intervals. Local connections from the input layer to the first hidden layer (W layer, Figure 17) are applied, i.e., one hidden neuron gather information only from a group of consequent input neurons. The Xs are categorized into ten groups according to the predefined sectors in S&P 500, and Q part is also categorized into ten groups, one for each time interval. In total, there are $N = 20$ hidden neurons (the Ws) in the first-hidden layer. The relationship between the Ws and the Xs and Qs are expressed as follows:

$$W_n = \sigma\left(\sum_{p=s_n}^{e_n} \alpha_{np} * X_p\right), n = 1, \dots, 10 \tag{3.3.1}$$

$$W_n = \sigma\left(\sum_{p=s_n}^{e_n} \alpha_{np} * Q_p\right), n = 11, \dots, N \tag{3.3.2}$$

where $\sigma(x) = \frac{1-e^{-x}}{1+e^{-x}}$ is the activation function, s_n and e_n are the starting and ending indexes for n^{th}group of input variables.

In addition to the 20 Ws, another 15 inputs neurons, corresponding to different exponential moving averages for stock j, are added into the first hidden layer to connect with five hidden neurons (the Zs) in the second hidden layer. The relationship between the Zs and the Ws and $EMAs$ can be expressed as follows:

$$Z_m = \sigma\left(\sum_{n=1}^{20} \beta_{mn} * W_n + \sum_{n\prime=1}^{15} \beta_{mn\prime} * EMA_{n\prime}\right) \tag{3.3.3}$$

Finally, three more input neurons, corresponding to the advanced-decline information in S&P 500 (see the next section for details), are added into the second hidden layer with the Zs, which are jointly connected to the output Y. The relationship between the the Y and the Zs and ADs can be expressed as follows:

$$g(Y) = \sum_{m=1}^{5} \theta_{km} * Z_m + \sum_{m\prime=1}^{3} \theta_{km\prime} * AD_{m\prime} \tag{3.3.4}$$

where $g(\cdot)$is the link function. Here, since $Y = X_{t+1,j}$ is continuous, g can simply take an identity function, i.e. $g(x) = x$. However, this model structure can be readily extended to binary or multi-categorical outputs, where g can take a soft-max function, i.e. $g(x) = \frac{e^x}{1+e^x}$.

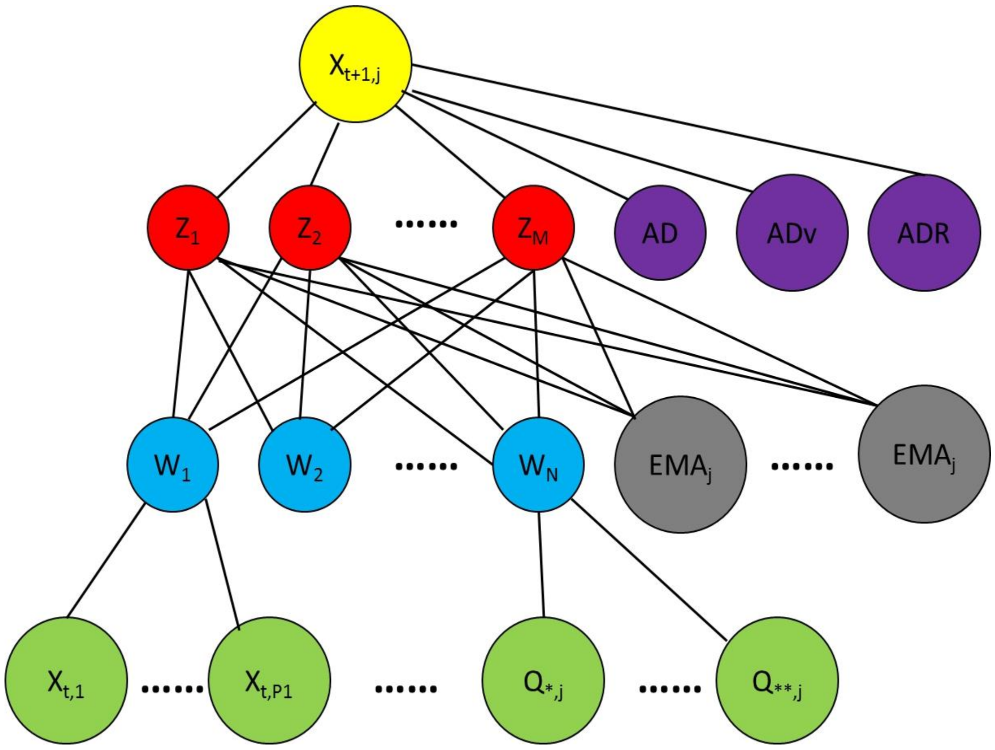


Figure 17. The Double Layer Neural Network Framework.

Inputs in the bottome layers are summarized into hidden variables Ws, which then subsequently interacts with the input EMAs and form the second-layer hidden variables Zs. The Z variables and market indicators serve as the final input for forecasting.

3.3.2 Generating Input Variables

The input variables consist of several technical indicators for a specific stock of interest and also a few market indicators that gauge the overall upward/downward market trend.

3.3.2.1 Intra-interval Proportions

For a particular 5-minute interval at time t, the relationship between its open, high, low and close prices may provide meaningful information regarding to the price movement, particularly for short-term momentum movements. For example, a close located nearby the low or nearby the high may suggest different market sentiment that is useful in the prediction. Three intra-interval proportions, $Q_{t,j} = \left(\frac{a_{t,j}}{d_{t,j}}, \frac{b_{t,j}}{d_{t,j}}, \frac{c_{t,j}}{d_{t,j}}\right)$, are used to summarize the information for a time interval t,where $a_{t,j} = High - Open$; $b_{t,j} = Open - Close$; $c_{t,j} = Close - Low$ and $d_{t,j} = High - Low$.

3.3.2.2 Exponential Moving Average (EMA)

Although the original price series $(S_{t,j})$ and its corresponding EMAs carry essentially the same information, we favor EMAs as input variables as they are smoother and could generate more stable performance in the perdition. The EMAs are calculated with the following formula and m controls the decaying rate:

$$EMA_t(m, S_{t,j}) = \alpha S_{t,j} + (1-\alpha)EMA_{t-1}(m, S_{t,j}) \qquad (3.3.5)$$

where $t \geq m$ and $\alpha = \frac{2}{m+1}$. We use $m = 1 - 10,\ 20, 30, 40, 50, 100$ as inputs. Notice that $m = 1$ leads to $\alpha = 1$ which means $EMA_t(1, S_{t,j}) = S_{t,j}$.

3.3.2.3 Advance/Decline (AD), Advance/Decline volume (ADv) and Advance/Decline Ratio (ADR) indicators

These indicators are used to measure the market breadth. The AD indicator counts the net number of advancing stocks within the 5-minute interval at time t, which is the number of advancing stocks less the number of declining stocks. ADv is the net volume of advancing stocks within a time interval t, which is the volume of advancing stocks less the volume of declining stocks. The ADR is calculated by dividing the volume of advancing stocks by the volume of declining stocks.

3.3.3 Sector Analysis

Since the price movements of different stocks are correlated, we try to incorporate these correlations into our DNN model structure to borrow information from other stocks in predicting the stock of interest. The pair-wise correlation among all S&P 500 stocks has been estimated and plotted in the Figure 18, based on returns in the first 2000 training intervals in the tested periods, which can be approximately viewed as independent samples. Our analyses on the ten S&P 500 sectors, including Consumer Discretionary, Consumer Staples, Energy, Financials, Health Care, Industrials, Materials, Information Technology, Telecommunication Services and Utilities, indicates that stocks within one sector tend to move in the same direction, and the correlations are largely within sectors. For example, stocks in Energy or Utilities have very litter correlations with stocks outsides these sectors. To reduce the number of parameters in the model, which

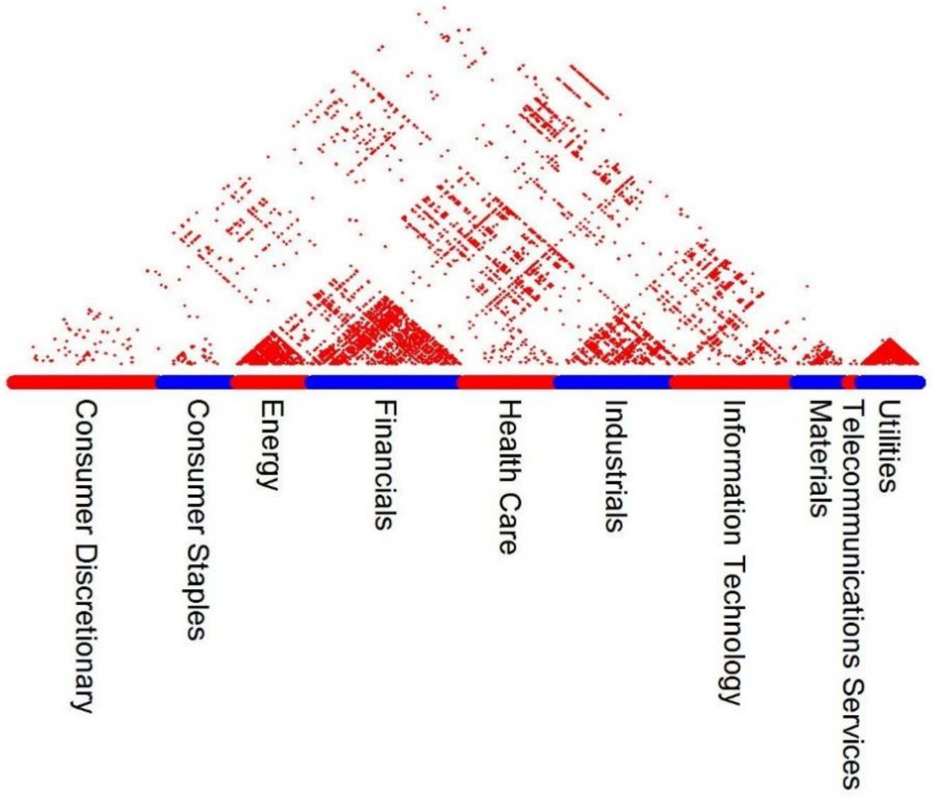


Figure 18. Correlation analysis for stocks in S&P 500 sectors

The pairwise correlations among 500 stocks, which are grouped according to their sectors, are calculated. The redness indicates the correlation strength.

subsequently alleviate computing burdens, we decide to summarize the section information first (the bottom input layer in Figure 17) and then use them to interact with EMAs of a specific stock for forecasting.

3.3.4 Target Function and Regularization

The complete parameter set, Ω, consists of all the links in the DNN model structure. For example, $\Omega = \{\alpha_{np}, \beta_{mn}, \beta_{mn'}, \theta_{km}$ and $\theta_{km'}: n = 1, \dots, N; n' = 1, \dots, 10;\ p = 1, \dots, P; k = 1, \dots, K; m = 1, \dots, M;\ m' = 1, \dots, 3\}$.

For the regression problem, the target function can be simply set to be the sum-of-squared errors:

$$R(\Omega) = \sum_{i=1}^{I} (y_i - \hat{y}_i)^2 \tag{3.3.6}$$

y_i is the actual return for ith training data point and $\hat{y}_i$is the predicted return based on neural network model.

Since the number of parameters in Ω is typically large, it is very easy to over-fit the network model and in those situations, a global minimizer of $R(\Omega)$ is not desired. Usually, early stopping rules may be placed to obtain local minimizer, which means the model is only trained for some cycles and stopped well before it approaches the global minimum. Alternatively, regularization may be added through some penalty terms, which is how we implement the algorithm here.

A L_2-penalty can be used in the network models, and correspondingly the target function becomes

$$R_p(\Omega) = R(\Omega) + \lambda J(\Omega) = R(\Omega) + \lambda ||\Omega||^2 \tag{3.3.7}$$

where $R(\Omega)$ is the original error function, $J(\Omega)$ is the penalty term, with $\lambda \geq 0$ being a tuning parameter (as introduced in Section 1.3.2). Larger values of λ will tend to shrink the weights toward zero. To choose an appropriate value for the hyper-parameter λ, cross-validation (CV) should be applied. However, in practice we found a pre-specified penalization coefficient λ of 0.5 works well. Therefore, to save the computing time on CV, we prefix λ to be 0.5 in our implementation.

3.3.5 Learning Algorithm

Minimizing the penalized target function in the above is a non-linear, non-convex problem, and numerical algorithms need to be applied. The gradient descent algorithm is applied here (as introduced in Section 1.3.3.1). Let $\eta \in \Omega$ represent a parameter in neural network models. The target function is minimized in an iterative process:

$$\eta_{r+1} = \eta_r - \gamma_r * \left.\frac{\partial R_p}{\partial \eta}\right|_{\eta=\eta_r} \tag{3.3.8}$$

where γ_r is the learning rate at step r which is chosen to be: $\gamma_r = 1/r$. The process stops when $\left|\frac{\eta_{r+1}-\eta_r}{\eta_r}\right| < 10^{-6}$.

The implementation of the algorithm is realized using the C language, which was compiled into a dynamic link file and loaded into R version 3.1.2 to carry out the subsequent analyses.

3.3.6 The Ensemble NN and Back-testing

In practice, we found that predications based on one single NN had some stability issues, that is, for the same period of data, if we ran NN multiple times, the prediction results may vary dramatically. The reason may lie in two-fold: 1) given the high dimensionality of the parameter space, the random start point of the algorithm may yield a big difference; 2) since the prediction pattern in financial market is well-known to be low, different local minima from the target functions may perform quite differently. Therefore, to increase the stability of our NN model, for each training data set, 1000 NN models will be trained and the top 50% performers will be selected for prediction. This resembles the idea of ensembles of methods, which led to the following dynamic back testing process.

At time t, the 5-minute intervals at time point from $t-2100$ to $t-101$ form the training data set. 1000 neural network models are built upon randomly selected 1000 time points from the training data set and validated on the validation data set consisting of 5-minute intervals at time points from $t-100$ to $t-1$. Each NN model has a profit curve over the 100 time points and has a ranking at each point showing how many models (curves) are better than this particular model at each point. The summation of these rankings for all 100 time points shows how well this particular model can profit on the validation data set. Fifty percent of the top models are chosen as a "committee" and this "committee" is applied to predict one step ahead returns along the next 100 5-minute intervals at time points ($t+1$ to $t+100$), which means the "committee" is updated every 100 time points. Figure 19 illustrates this dynamic training process.

Because of this "moving" design, our model needs to be updated for each new time point, which leads to very high computation burden. To make the model computationally feasible, we update the "committee" every 100 time points. This is not unreasonable as the fitted model is generated based on the committee and would only be affected when reasonable training data have been replaced.

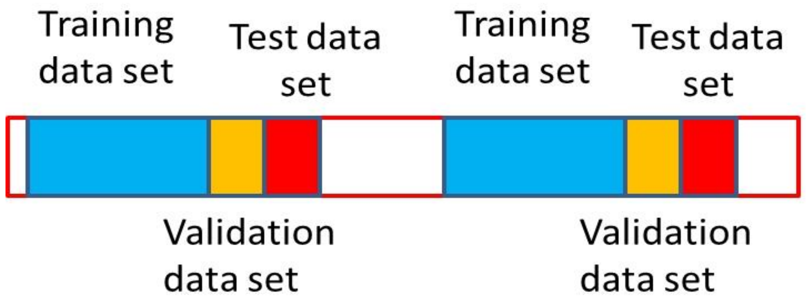


Figure 19. Dynamic back-testing process
The most recent intervals are divided into training (Blue) and validation (Yellow) periods. The NN models are trained with the training data and then their performances are evaluated with the validation data. The top NN models are then combined to predict the movement in the test period (Red).

3.4 Benchmark Methods and Evaluation Metrics

3.4.1 Benchmark Methods

3.4.1.1 ARMA-GARCH

The first baseline method we use to compare with our framework is ARMA(1,1)-GARCH(1,1), a well-known approach for financial time series. For the time series of return for stock j,

$$X_{t,j} = \phi_{0,j} + \phi_{1,j}X_{t-1,j} + \epsilon_{t,j} + \varphi_{1,j}\epsilon_{t-1,j} \tag{3.4.1}$$

where the residuals

$$\epsilon_{t,j} = \sigma_{t,j}u_{t,j} \tag{3.4.2}$$

and the time-dependent variance

$$\sigma_{t,j}^2 = \alpha_{0,j} + \alpha_{1,j}\sigma_{t-1,j}^2 + \beta_{1,j}\epsilon_{t-1,j}^2 \tag{3.4.3}$$

The unknown parameters $\{\phi_0, \phi_1, \varphi_1, \alpha_0, \alpha_1, \beta_1\}$ are estimated via maximum likelihood method. Notice that unit innovation $u_{t,j}$ is a strong white noise process which can be estimated by $\hat{u}_{t,j} = \frac{\hat{\epsilon}_{t,j}}{\hat{\sigma}_{t,j}}$.

3.4.1.2 ARMAX-GARCH

The second baseline approach is an extension of ARMA(1,1)-GARCH(1,1). For the given time series $X_{t,j}$, ARMAX-GARCH model considers endogenous variable $X_{t,j}$ and a set of exogenous variables $Y_{1t,j}, Y_{2t,j}, \ldots, Y_{1t,j}$. Then, the original model is modified as:

$$X_{t,j} = \phi_{0,j} + \phi_{1,j}X_{t-1,j} + \sum_{i=1}^{n} \mu_{i,t-1,j}Y_{i,t-1} + \epsilon_{t,j} + \varphi_{1,j}\epsilon_{t-1,j} \tag{3.4.4}$$

where n is the number of exogenous variables. In our experiments, we include input variables of intra-interval proportions, EMA and A/D in our double neural network.

3.4.1.3 The Regular Neural Network Model

The third baseline approach is the single hidden layer neural network, which is the simplest but probably also the most widely used NN structure. It consists of three layers: the bottom input layer, the middle hidden layer and the top output layer. Every neuron in the hidden layer is fully connected with all other neurons in the input and output layers. This model includes the same input variables as ARMAX-GARCH. The function "nnet" in the standard R package "nnet" was used to implement this regular NN model.

3.4.2 Evaluation Metrics

For certain stock j at each time intervals $t,$ a prediction would be made for the next interval $t+1$ based on a specific model $m.$ If the model prediction is consistent with the true price movement, a profit is yielded; otherwise a loss incurs. Let us denote the return for model m at interval t as $r_{m,t,j} = sgn(\hat{y}_{m,t,j} y_{t,j}) * |y_{t,j}|$. $y_{t,j} = X_{t+1,j}$ is the real increase rate during next 5 minute and $\hat{y}_{m,t,j}$ is the predicted increase rate from model m during next 5 minute. Assume the total number of intervals being tested is T_0. We used the following criteria to evaluate/compare the performance of different models.

(1) Absolute return:

$$r_{m,j} = \prod_{t=1}^{T_0} (1 + r_{m,t,j}) \tag{3.4.5}$$

This is equivalent to a strategy of buying or short selling a fixed number of shares according to the model prediction on the next time point $t+1$.

(2) Sharpe Ratio:

$$SP_{m,j} = \frac{\bar{r}_{m,j}}{s_{m,j}} \tag{3.4.6}$$

Where the average return $\bar{r}_{m,j} = \frac{\sum_{t=1}^{T_0} r_{m,t,j}}{T_0}$ and the standard deviation $s_{m,j} = \sqrt{\frac{1}{T_0 - 1} \sum_{t=1}^{T_0} (r_{m,t,j} - \bar{r}_{m,j})^2}$. Sharpe ratio quantifies the risk efficiency of a model performance.

(3) Prediction Accuracy:

$$A_{m,j} = \frac{\sum_{t=1}^{T_0} I(r_{m,t,j} > 0)}{T_0} \tag{3.4.7}$$

Prediction accuracy is calculated by simply the ratio of the total number of correct predictions on the direction of the price movement over the total of tested intervals.

3.5 Overall Performance

We downloaded the data for the 500 stocks in the Standard & Poor’s 500, ranging from January 1, 2013 to May 31, 2013. There are 104 trading days and each day contains 78 5-minute intervals, which makes 8112 time points (observations) in total. The raw data include open, high, low and close prices and also volume for each stock at each time point. Table 2 presented the basic statistics for 500 stocks’ trading volumes and their market capitalizations on May 31, 2013.

Our DNN method together with the other three benchmark methods were applied on 100 stocks with largest capital in S&P 500 to compare their performances.

	Mean	**Standard Deviation**	**Median**	**Min**	**Max**
Volume	11.8	60.2	3.9	0	24951.1
Market cap	32.1	51.2	15.3	3.0	422.1

Table 2. Descriptive statistics for 500 stocks’ trading volumes in 5 minute level (unit: thousands shares) and market capitalizations on May 31, 2013 (unit: billions USD).

3.5.1 Forecasting Power

Prediction accuracy was used as the main metric for the comparisons between the four methods. We also explored prediction accuracies on time points with large predicted returns. The rational is that returns close to zeros may mainly represent market noises and therefore harder to predict, and models may perform better in those time intervals with larger returns.

The new returns were defined as:

$$r^*_{m,t,j} = \begin{cases} r_{m,t,j}, & when\ |\hat{y}_{m,t,j}| > C_{m,j} \\ 0, & when\ |\hat{y}_{m,t,j}| \le C_{m,j} \end{cases} \tag{3.5.1}$$

$m = 1, \dots, 4$ represented four methods: DNN, SNN (regular NN model), ARMA-GARCH and ARMAX-GARCH. $j = 1, \dots, 100$ represented the 100 stocks involved in the study. t was the time point. The new accuracy rate for method m and stock j was defined as:

$$A^*_{m,j} = \frac{\sum_{t=1}^{T_0} I(r^*_{m,t,j} > 0)}{\sum_{t=1}^{T_0} I(r^*_{m,t,j} \ne 0)} \tag{3.5.2}$$

In our study, the accuracy rates were calculated in each 100 time points, that is, for each method and each stock with 2000 testing time points, there would be 20 accuracy rates. We also defined the proportion of transactions as:

$$P_{m,j} = \frac{\sum_{t=1}^{T_0} I(r^*_{m,t,j} \ne 0)}{T_0} \tag{3.5.3}$$

$C_{m,j}$'s are pre-selected to ensure all $P_{m,j}$ have the same value.

Annualized Sharpe ratios were also calculated: SP_m in previous section was the Sharpe ratio for 5 minute level. The annualized Sharpe ratio, according to its definition, should be $\sqrt{n} * SP_m$ with n being the number of 5 minute intervals in one year.

All four methods are applied on all 100 stocks and each stock has at most 2000 testing points divided into 20 intervals. So for each method, each stock, each interval, there is one prediction accuracy rate calculated based on the pre-defined $C_{m,j}$'s which share the same proportion of transaction. Then we repeat the previous studies for different proportions of transactions: 90%, 70%, 50%, 30%, 10%. Figure 20 summarizes all the accuracy rates with the corresponding 95% confidence intervals. Each point and the confidence interval in Figure 20 represents 2000 = 100 (stocks) * 20 (accuracy rates per stock) accuracy rates. From Figure 20, we conclude the following two statements:

(1) As the proportion of transaction decrease, all four models have increased performance in terms of prediction accuracy rate, which means larger predicted values does show better prediction on whether the stock would increase or decrease.

(2) Our designed method based on DNN models outperforms the other three benchmark methods especially when the proportion of transactions are below 50% - the confidence intervals show that the differences are significant. This suggests that larger predicted values from our method have more probability on predicting the right trend for stocks in the next 5 minute.

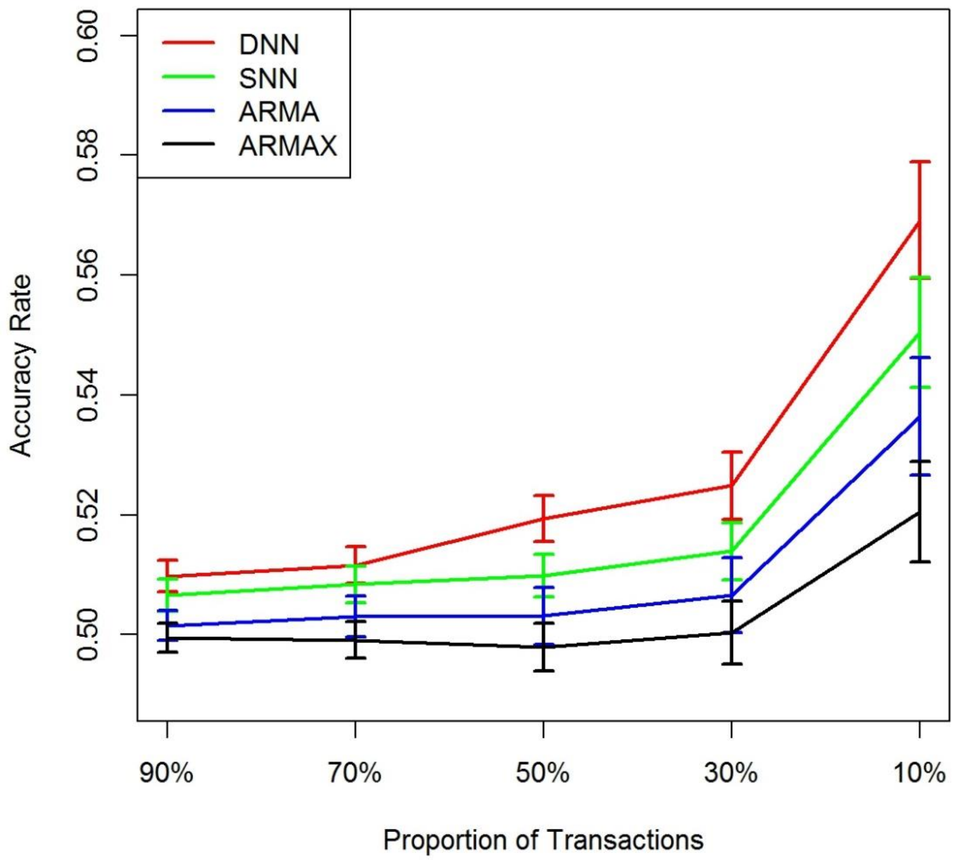


Figure 20. Accuracy rates and corresponding 95% confidence intervals for DNN, SNN, ARMA-GARCH and ARMAX-GARCH models applied on 100 leading stocks

Each point in the figure represents the averaged prediction accuracy rate for one method applied in all 100 stocks using one set of $C_{m,j}$ sharing the same proportion of transactions. DNN model always outperforms to the other models and the differences are significant.

Table 3 presented the annualized Sharpe ratios for four methods at corresponding proportions of transactions. Medians with 25%, 75% quantiles of the 100 stocks were calculated and shown in the table. From Table 3, we observed that DNN outperformed to the other three methods in 4 out of 5 proportion of transactions: 90%, 70%, 50% and 30% in terms of annualized Sharpe ratio. And DNN reached the highest median annualized Sharpe ratio at the cut off values making 50% of the transactions.

Method	**Proportion of transactions**				
	90%	**70%**	**50%**	**30%**	**10%**
DNN	1.50 (-1.07, 4.11)	1.29 (-1.24, 4.33)	1.70 (-0.50, 4.55)	1.55 (-0.73, 4.05)	0.89 (-1.10, 3.00)
SNN	0.92 (-0.60, 3.05)	1.08 (-1.17, 3.69)	1.20 (-1.09, 3.10)	1.23 (-0.83, 3.51)	1.66 (-0.14, 3.39)
ARMA	0.63 (-1.21, 1.47)	0.16 (-1.40, 1.53)	0.23 (-1.35, 1.88)	0.14 (-1.63, 1.63)	0.48 (-1.51, 1.80)
ARMAX	-0.44 (-1.83, 1.31)	-0.25 (-1.75, 0.86)	-0.55 (-2.13, 0.93)	0.06 (-1.97, 1.48)	0.27 (-1.92, 1.93)

Table 3. Annualized Sharpe ratios for four methods at different proportions of transactions.

Each cell presented the median with 25%, 75% quantiles of annualized Sharpe ratios among the 100 stocks with corresponding method and proportion of transactions applied.

3.5.2 Computations

Efficient R programs have been developed to implement our proposed model. Roughly, one DNN fitting takes approximately 0.8 seconds in a regular workstation (Intel Xeon E5-2620 @ 2.00GHz), which means that fitting 1000 DNNs at one step would take about 13.3 minutes. If this model were to be applied to support real-life trading, we can afford to update the model every three 5-minute intervals. For data with higher frequencies and also with multiple stocks, we can use clusters to parallelize the process to speed up the model updating.

3.5.3 Case Studies

The absolute returns were calculated for all 100 stocks at each time point as Formula (3.4.5). Both DNN method and ARMA-GARCH method outperformed to the other methods on 30 stocks out of 100 stocks. As two case studies, Figure 21 and Figure 22 showed the absolute return curves for all four methods applied on Accenture Plc (ACN) and Baker Hughes Incorporated (BHI). Each figure contained five curves, with each representing one strategy. Each curve showed the portfolio value at each time point if we invested 1 dollars at time point 1, applying the corresponding strategy. The blue curve was for buy-and-hold strategy and actually showed the trend of market price. The other four curves represented strategies utilizing the four methods: red for DNN, green for SNN, brown for ARMA-GARCH and black for ARMAX-GARCH. But notice that these portfolio values did not consider the transaction costs.

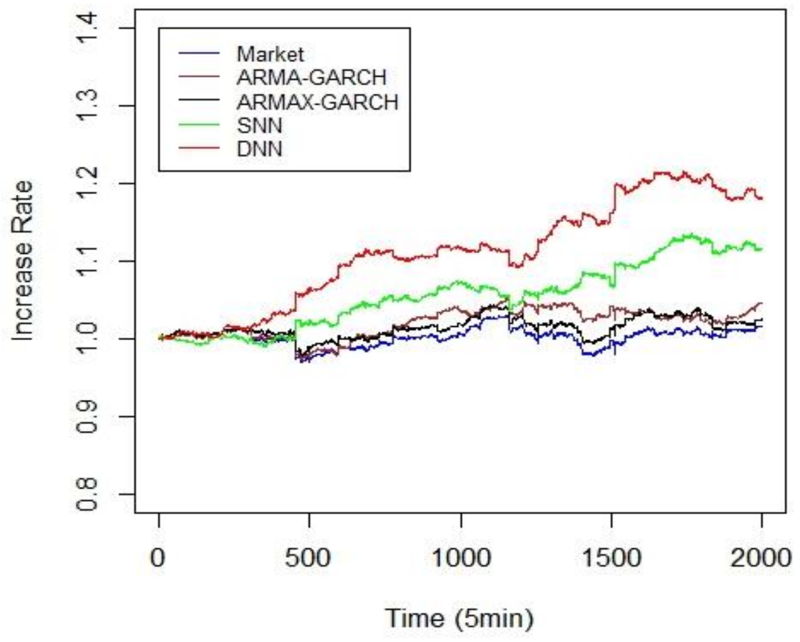


Figure 21. Absolute return curves for all methods applied on ACN
For ACN, DNN (red line) performed the best among four methods. The 2000 time points profit would be around 20% for DNN method.

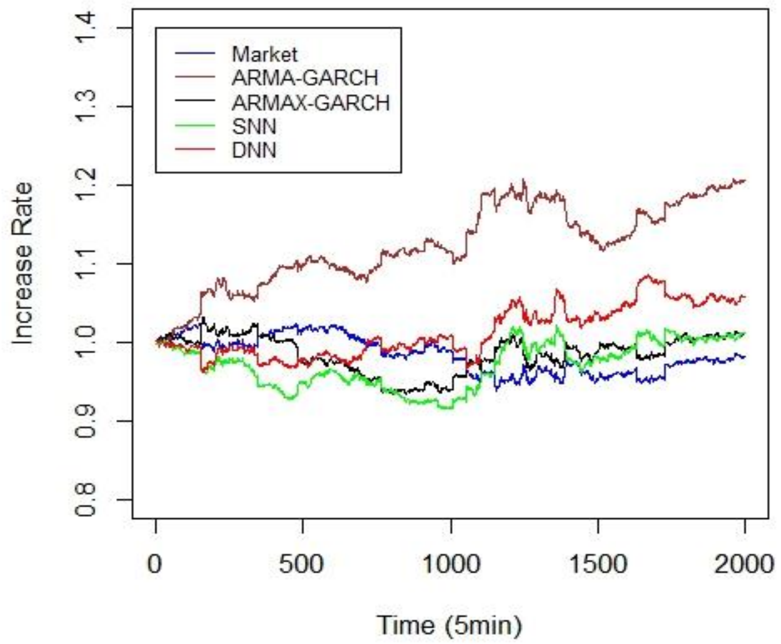


Figure 22. Absolute return curves for all methods applied on BHI
ARMA-GARCH (brown line) performed the best among the four methods. The 2000 time points profit would be around 20% for ARMA-GARCH method.

3.6 Conclusions and Future Work

Our method differs from the others and provides new information in two aspects. Firstly, we proposed a novel multi-layer NN framework with local connections to handle the large correlations existing only in high frequency stock data, which improves the prediction performance, while most related works that are designed for low-frequency data did not consider intra-market correlations. Secondly, we investigated and benchmarked the performance of our newly proposed model, together with a regular NN model and two classical econometric models for high-frequency forecasting. It is possible that large amount of transaction costs might dramatically reduce the profits in high-frequency trading, however, our model can still help in providing effective solutions. For example, focusing on predictions with large returns may be used to design useful trading strategies as it would not only reduce the number of transactions but also increase the prediction accuracy rates.

For the future work, we plan to study the performance of our model for data with even higher frequency. We will study whether the prediction accuracy rates would remain the same for data at minute or even second level. For example, if for minute-level data, the prediction accuracy rates remain similar or just have minor decrease, then the framework based on our DNN models can make nearly 5 times of the profit as using 5-minute level data within the same period of trading days. It is also possible that a new NN structure might be necessary according to the characteristics in data with higher frequencies.

Another possible improvement for our framework is to establish a dynamic system for the DNN models. The system should allow 1000 DNN models (as introduced in Section 3.3.6) to be updated as needed. We need to study not only when to update but also what faction of the 1000 DNN models will be updated. It is definitely not optimal to update all 1000 DNN models at every time point, because it is quite time consuming and if there is only one record changed in the training data set, the DNN models should be similar as previous set of models.

In the aspect of model structure, we think it may be worth to try on recurrent neural networks that are designed specifically for time series data. The crucial part is how to integrate the other environmental inputs (those technical indicators introduced in Section 3.3.2) into the models and determine which part, the time series information or the environmental inputs, should have more effect on the outcomes and how to realize this importance through structure designs.

For comparison, it is also worth to include random forests or other machine learning models. Random forests model is expected to have much less computational burden, but how well it can performance in this high frequency scale remains unknown. It is also possible that if random forests perform similarly with double layer neural network, we can make ensemble predictors based on both models.

CHAPTER 4 FUTURE WORK ON NEURAL NETWORKS

4.1 Development of New Structures

Recently, more and more researchers, especially those focusing on the mathematical models of neural networks, have been changing their interests to designing novel structures. For example, in 2006, Hinton and his collaborators [23] proposed the so-called "deep belief nets" model, and in 2007, they further introduced how a many-layered neural network model could be trained effectively and gain higher power [22]. Based on these work, nowadays deep learning has been developed and become a popular area in machine learning.

Along with the growth of deep learning, some other creative and efficient neural network structures that did not receive much attention before have also been brought back for further studies. Convolutional neural networks and recurrent neural networks might be the two most accepted structures. Convolutional neural networks have been applied by Yann LeCun in his hand-written numbers recognition [38] [37]. The major characteristics of this structure is to use the shared weight (the convolution operations) in convolutional layers, which can both reduce the required number of parameters and improves performance, especially in image recognition where the convolution operations are meaningful. Recurrent neural networks are also a set of neural network models developed long time before and it is a structure designed specifically for time series like data.

Combining the above ideas can provide broad choices for the structures of neural networks. As we believe that the most promising way to apply neural networks is to design data-oriented structures, for the S&P 500 stock price predictions, both convolutional neural networks and recurrent neural networks might find their way to help improve the performance. For GWA study, the data is less structured than stock price data set, but may still be suitable for convolutional structures. Although we can pre-determine the types of structures based on our intuition and motivations, more structure related parameters still need many trials or systematic tuning process to determine their best values. At some aspects, designing or choosing a suitable structure for neural networks is more like an art instead of scientific research.

4.2 Development of New Training Algorithms

Another aspect of neural network models that draw researchers' attentions is the training algorithm. Back-propagation is usually the first choice because of its stability and comparably faster convergence speed. There are also some variations of back-propagation algorithm like, such as using the second order derivatives (Hessian matrix) or Bayesian framework [48] to implement the algorithm. The biggest drawback for back-propagation is its generalizability – each new structure may need significant change on the iterative formulas and thus need to rewrite the corresponding codes. So the other two algorithms (simulated annealing and genetic algorithm) might be more suitable when searching among different types of structures.

In essence, the training algorithm is simply a nonlinear optimization (nonlinear programming) problem, which has long been a difficulty in computational area. Till now, no accepted algorithms can guarantee that the global minimal point for a general nonlinear optimization problem can be found. However, for different types of nonlinear optimization problems, there are a lot of researches focusing on improving the existed general algorithm based on the specialty of problems. Back-propagation with only the first derivatives is an implementation of gradient descent method and is an extension of Newton-Raphson method if including the Hessian matrix. Some other specific algorithms for neural network could be found in [45] [18] [61]. It might be worth to try a hybrid system of back-propagation and simulated annealing algorithms because back-propagation is good at local convergence and simulated annealing enable the searching to jump out of local minimal point [43].

4.3 Other Distributions for Dependent Variables

Continuous and multi-categorical outcomes are the two most common data types in statistics and machine learning, but there also exist other interesting outcome data type like count data (modeled by Poisson or Negative Binomial distribution) and censored data (survival analysis). Among those popular machine learning models like SVM and Random Forests, neural networks might be the easiest one which could be extended to problems with the other types of outcome variables. There are very little researches tackling this topic. The most important step is to determine how to organize the outcome neurons to well represent the outcome variables and how to transfer from the last hidden layer to the outcome layer (see Figure 3). Since the outcome variables may not represent information in the same scale, designing a proper error function would also be very critical. For example for censored data, it might have one binary outcome neuron represents censoring or not, another continuous outcome neuron representing the follow-up time. Then how to define the error function which can be minimized to fit the two outcome neurons simultaneously? It is a promising direction for neural networks related researches, and if the idea works, the application area of neural networks could be further extended.

REFERENCE

[1] T. G. ANDERSEN, T. BOLLERSLEV, F. X. DIEBOLD and H. EBENS, *The distribution of realized stock return volatility*, Journal of financial economics, 61 (2001), pp. 43-76.

[2] K. BRETNEY and Z. COBURN, *High frequency trading with an artificial neural network*, Mimeo, 2008.

[3] R. M. CANTOR, K. LANGE and J. S. SINSHEIMER, *Prioritizing GWAS results: a review of statistical methods and recommendations for their application*, The American Journal of Human Genetics, 86 (2010), pp. 6-22.

[4] P.-C. CHANG, D.-D. WANG and C.-L. ZHOU, *A novel model by evolving partially connected neural network for stock price trend forecasting*, Expert Systems with Applications, 39 (2012), pp. 611-620.

[5] G. M. CHEN, M. FIRTH and O. M. RUI, *The dynamic relation between stock returns, trading volume, and volatility*, Financial Review, 36 (2001), pp. 153-174.

[6] T.-L. CHEN, C.-H. CHENG and H. J. TEOH, *Fuzzy time-series based on Fibonacci sequence for stock price forecasting*, Physica A: Statistical Mechanics and its Applications, 380 (2007), pp. 377-390.

[7] T. C. CHIANG, H.-C. YU and M.-C. WU, *Statistical properties, dynamic conditional correlation, scaling analysis of high-frequency intraday stock returns: Evidence from Dow-Jones and nasdaq indices*, Physica A, 388 (2009), pp. 1555-1570.

[8] D. C. CIREŞAN, U. MEIER, L. M. GAMBARDELLA and J. SCHMIDHUBER, *Convolutional neural network committees for handwritten character classification*, *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, IEEE, 2011, pp. 1135-1139.

[9] J. CLARK, I. KOPRINSKA and J. POON, *A neural network based approach to automated e-mail classification*, *null*, IEEE, 2003, pp. 702.

[10] H. J. CORDELL, *Detecting gene–gene interactions that underlie human diseases*, Nature Reviews Genetics, 10 (2009), pp. 392-404.

[11] W. DAI, J.-Y. WU and C.-J. LU, *Combining nonlinear independent component analysis and neural network for the prediction of Asian stock market indexes*, Expert Systems with Applications, 39 (2012), pp. 4444-4452.

[12] F. D. DE FREITAS and A. R. DE ALMEIDA, *Portfolio selection with predicted returns using neural networks*, *IASTED International Conference on Artificial Intelligence and Applications*, 2001, pp. 99-103.

[13] W. N. FRANKEL and N. J. SCHORK, *Who's afraid of epistasis?*, Nature genetics, 14 (1996), pp. 371-373.

[14] M. GHIASSI, J. SKINNER and D. ZIMBRA, *Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network*, Expert Systems with Applications: An International Journal, 40 (2013), pp. 6266-6282.

[15] P. GOMBER, B. ARNDT, M. LUTAT and T. UHLE, *High-frequency trading*, Available at SSRN 1858626 (2011).

[16] N. GRADOJEVIC, R. GENÇAY and D. KUKOLJ, *Option pricing with modular neural networks*, Neural Networks, IEEE Transactions on, 20 (2009), pp. 626-637.

[17] E. GURESEN, G. KAYAKUTLU and T. U. DAIM, *Using artificial neural network models in stock market index prediction*, Expert Systems with Applications, 38 (2011), pp. 10389-10397.

[18] M. T. HAGAN and M. B. MENHAJ, *Training feedforward networks with the Marquardt algorithm*, Neural Networks, IEEE Transactions on, 5 (1994), pp. 989-993.

[19] M. R. HASSAN, B. NATH and M. KIRLEY, *A fusion model of HMM, ANN and GA for stock market forecasting*, Expert Systems with Applications, 33 (2007), pp. 171-180.

[20] T. HASTIE, R. TIBSHIRANI, J. FRIEDMAN, T. HASTIE, J. FRIEDMAN and R. TIBSHIRANI, *The elements of statistical learning*, Springer, 2009.

[21] J. HERRERO, A. VALENCIA and J. DOPAZO, *A hierarchical unsupervised growing neural network for clustering gene expression patterns*, Bioinformatics, 17 (2001), pp. 126-136.

[22] G. E. HINTON, *Learning multiple layers of representation*, Trends in cognitive sciences, 11 (2007), pp. 428-434.

[23] G. E. HINTON, S. OSINDERO and Y.-W. TEH, *A fast learning algorithm for deep belief nets*, Neural computation, 18 (2006), pp. 1527-1554.

[24] J. N. HIRSCHHORN and M. J. DALY, *Genome-wide association studies for common diseases and complex traits*, Nature Reviews Genetics, 6 (2005), pp. 95-108.

[25] K. HORNIK, *Approximation capabilities of multilayer feedforward networks*, Neural networks, 4 (1991), pp. 251-257.

[26] K. HORNIK, M. STINCHCOMBE and H. WHITE, *Multilayer feedforward networks are universal approximators*, Neural networks, 2 (1989), pp. 359-366.

[27] S.-H. HSU, J. P.-A. HSIEH, T.-C. CHIH and K.-C. HSU, *A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression*, Expert Systems with Applications, 36 (2009), pp. 7947-7951.

[28] R. S. JOAO, T. F. GUIDONI, J. R. BERTINI, M. NICOLETTI and A. O. ARTERO, *Stock Closing Price Forecasting Using Ensembles of Constructive Neural Networks, Intelligent Systems (BRACIS), 2014 Brazilian Conference on*, IEEE, 2014, pp. 109-114.

[29] L. K. JONES, *A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training*, The annals of Statistics (1992), pp. 608-613.

[30] Y. KARA, M. ACAR BOYACIOGLU and Ö. K. BAYKAN, *Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange*, Expert systems with Applications, 38 (2011), pp. 5311-5319.

[31] J. M. KARPOFF, *The relation between price changes and trading volume: A survey*, Journal of Financial and quantitative Analysis, 22 (1987), pp. 109-126.

[32] T. KOHONEN, *An introduction to neural computing*, Neural networks, 1 (1988), pp. 3-16.

[33] M. A. KRAMER, *Nonlinear principal component analysis using autoassociative neural networks*, AIChE journal, 37 (1991), pp. 233-243.

[34] D. KRIESEL, *A brief introduction to neural networks*, Retrieved August, 15 (2007), pp. 2011.

[35] C. G. LAMOUREUX and W. D. LASTRAPES, *Heteroskedasticity in stock return data: volume versus GARCH effects*, The Journal of Finance, 45 (1990), pp. 221-229.

[36] S. LAWRENCE, C. L. GILES, A. C. TSOI and A. D. BACK, *Face recognition: A convolutional neural-network approach*, Neural Networks, IEEE Transactions on, 8 (1997), pp. 98-113.

[37] B. B. LE CUN, J. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD and L. D. JACKEL, *Handwritten digit recognition with a back-propagation network*, *Advances in neural information processing systems*, Citeseer, 1990.

[38] Y. LECUN, *Generalization and network design strategies*, Connections in Perspective. North-Holland, Amsterdam (1989), pp. 143-55.

[39] M. D. LI, *Identifying susceptibility loci for nicotine dependence: 2008 update based on recent genome-wide linkage analyses*, Human genetics, 123 (2008), pp. 119-131.

[40] F. LIU and J. WANG, *Fluctuation prediction of stock market index by Legendre neural network with random time strength function*, Neurocomputing, 83 (2012), pp. 12-21.

[41] J. MALLEY, J. KRUPPA, A. DASGUPTA, K. MALLEY and A. ZIEGLER, *Probability machines: consistent probability estimation using nonparametric learning machines*, Methods of information in medicine, 51 (2012), pp. 74.

[42] W. S. MCCULLOCH and W. PITTS, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics, 5 (1943), pp. 115-133.

[43] P. D. MCNELIS, *Neural networks in finance: gaining predictive edge in the market*, Elsevier Acad. Press, 2005.

[44] M. B. MIKHAIL, B. R. WALTHER and R. H. WILLIS, *Does forecast accuracy matter to security analysts?*, The Accounting Review, 74 (1999), pp. 185-200.

[45] M. F. MØLLER, *A scaled conjugate gradient algorithm for fast supervised learning*, Neural networks, 6 (1993), pp. 525-533.

[46] H. MUTOH, N. HAMAJIMA, K. TAJIMA, T. KOBAYASHI and H. HONDA, *Exhaustive exploring using Artificial Neural Network for identification of SNPs combination related to Helicobacter pylori infection susceptibility*, Chem-Bio Informatics Journal, 5 (2005), pp. 15-26.

[47] R. NEWSON, *Parameters behind "nonparametric" statistics: Kendall's tau, Somers' D and median differences*, Stata Journal, 2 (2002), pp. 45-64.

[48] H. OKUT, D. GIANOLA, G. J. ROSA and K. A. WEIGEL, *Prediction of body mass index in mice using dense molecular markers and a regularized neural network*, Genetics research, 93 (2011), pp. 189-201.

[49] P.-F. PAI and C.-S. LIN, *A hybrid ARIMA and support vector machines model in stock price forecasting*, Omega, 33 (2005), pp. 497-505.

[50] M. Y. PARK and T. HASTIE, *Penalized logistic regression for detecting gene interactions*, Biostatistics, 9 (2008), pp. 30-50.

[51] J. PATEL, S. SHAH, P. THAKKAR and K. KOTECHA, *Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques*, Expert Systems with Applications, 42 (2015), pp. 259-268.

[52] W. PITTS and W. S. MCCULLOCH, *How we know universals the perception of auditory and visual forms*, The Bulletin of mathematical biophysics, 9 (1947), pp. 127-147.

[53] S. PURCELL, B. NEALE, K. TODD-BROWN, L. THOMAS, M. A. FERREIRA, D. BENDER, J. MALLER, P. SKLAR, P. I. DE BAKKER and M. J. DALY, *PLINK: a tool set for whole-genome association and population-based linkage analyses*, The American Journal of Human Genetics, 81 (2007), pp. 559-575.

[54] E. F. PUTRA and R. KOSALA, *Application of artificial neural networks to predict intraday trading signals*, (2011).

[55] A. M. RATHER, A. AGARWAL and V. SASTRY, *Recurrent neural network and a hybrid model for prediction of stock returns*, Expert Systems with Applications, 42 (2015), pp. 3234-3241.

[56] M. D. RITCHIE, L. W. HAHN and J. H. MOORE, *Power of multifactor dimensionality reduction for detecting gene ‐ gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity*, Genetic epidemiology, 24 (2003), pp. 150-157.

[57] M. D. RITCHIE, L. W. HAHN, N. ROODI, L. R. BAILEY, W. D. DUPONT, F. F. PARL and J. H. MOORE, *Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer*, The American Journal of Human Genetics, 69 (2001), pp. 138-147.

[58] D. RUMELHART, G. HINTON and R. WILLIAMS, *Learning internal representations by error propagation In: DE Rumelhart and JL McClelland, editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, 1986.

[59] D. E. RUMELHART, G. E. HINTONT and R. J. WILLIAMS, *Learning representations by back-propagating errors*, NATURE, 323 (1986), pp. 9.

[60] D. F. SCHWARZ, I. R. KÖNIG and A. ZIEGLER, *On safari to Random Jungle: a fast implementation of Random Forests for high-dimensional data*, Bioinformatics, 26 (2010), pp. 1752-1758.

[61] Y. SHANG and B. W. WAH, *Global optimization for neural network training*, Computer, 29 (1996), pp. 45-54.

[62] P. M. SHEA and V. LIN, *Detection of explosives in checked airline baggage using an artificial neural system*, *Neural Networks, 1989. IJCNN., International Joint Conference on*, IEEE, 1989, pp. 31-34.

[63] A. A. TODA, *The double power law in income distribution: Explanations and evidence*, Journal of Economic Behavior & Organization, 84 (2012), pp. 364-381.

[64] A. A. TODA and K. J. WALSH, *The double power law in consumption and implications for testing Euler equations*, Available at SSRN 2319454 (2014).

[65] Y. TOMITA, S. TOMIDA, Y. SUZUKI, T. SHIRAKAWA, T. KOBAYASHI and H. HONDA, *Artificial Neural Network Model for Prediction of Childhood Allergic Asthma*

Using Single Nucleotide Polymorphism Data, Genome Informatics, 14 (2003), pp. 593-594.

[66] C.-F. TSAI and Y.-C. HSIAO, *Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches*, Decision Support Systems, 50 (2010), pp. 258-269.

[67] J. VOIT, *The statistical mechanics of financial markets*, Springer Science & Business Media, 2013.

[68] M. L. WEITZMAN, *Subjective expectations and asset-return puzzles*, The American Economic Review (2007), pp. 1102-1130.

[69] B. WIDROW, D. E. RUMELHART and M. A. LEHR, *Neural networks: Applications in industry, business and science*, Communications of the ACM, 37 (1994), pp. 93-105.

[70] C. YANG, Z. HE, X. WAN, Q. YANG, H. XUE and W. YU, *SNPHarvester: a filtering-based approach for detecting epistatic interactions in genome-wide association studies*, Bioinformatics, 25 (2009), pp. 504-511.

[71] H. ZHANG and G. BONNEY, *Use of classification trees for association studies*, Genetic epidemiology, 19 (2000), pp. 323-332.

[72] Y. ZHANG and J. S. LIU, *Bayesian inference of epistatic interactions in case-control studies*, Nature genetics, 39 (2007), pp. 1167-1173.

[73] W. ZHOU, K. XIAO and F. SONG, *Dynamic rank correlation computing for financial risk analysis*, *Knowledge Science, Engineering and Management*, Springer, 2011, pp. 269-280.

[74] Z.-H. ZHOU, Y. JIANG, Y.-B. YANG and S.-F. CHEN, *Lung cancer cell identification based on artificial neural network ensembles*, Artificial Intelligence in Medicine, 24 (2002), pp. 25-36.