Spring 5-18-2018

Deep learning analysis of limit order book

xin xu
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

Department of Mathematics

Deep Learning Analysis of Limit Order Book

by

Xin Xu

A thesis presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Master of Art

May 2018
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to sincerely appreciate my advisor, Pro. Figueroa-Lopez, for his guidance in the whole process of this thesis. He invested a great amount of time in working with me. I am also thankful for his patience when I encounter difficulties. He also helps me a lot in my whole master career.

In addition, I would like to thank my committee members, Pro. Lin and Pro. Ding to give me advices and suggestions on my thesis. I would like to thank my friend, Zoe.yu, for giving me some intuitive ideas and discussing with me on some challenge parts of this paper.

Last but not least, I would like to thank my parents, without their supports, I can not accomplish my master program. I love them.

Xin Xu

Washington University in St.Louis

May 2018

ABSTRACT

Deep Learning Analysis of Limit Order Book

by

Xu, Xin

A.M. in Statistics,

Washington University in St. Louis, 2018.

Professor Jose Figueroa-Lopez, Chair

In this paper, we build a deep neural network for modeling spatial structure in limit order book and make prediction for future best ask or best bid price based on ideas of [1]. We propose an intuitive data processing method to approximate the data is non-available for us based only on level I data that is more widely available. The model is based on the idea that there is local dependence for best ask or best bid price and sizes of related orders. First we use logistic regression to prove that this approach is reasonable. To show the advantages of deep neural network, we try different activation functions and compare the performances and program running time with other algorithms, such as logistic regression, kNN and random forest. And the deep neural network is the model that most suitable for limit order book. Besides this, the model contains an effective way to reduce overfitting problems. Also, this paper presents the limitations of our model and gives several methods to make improvements.

1. Introduction

People use limit order book to record and present the changes of stock prices in electronic exchanges. It contains all critical information of buy and sell orders in stocks market. A Limit order book consists of ask and bid limit orders with different sizes at all price levels. A limit order is an order to buy or sell certain number of shares at a certain price. A market order is the order that be executed immediately at current market prices. that Since the LOB records changes in milliseconds, we can get the information of supply and demand at any time. The best bid/ask price keeps changing, because there are buyers and sellers in and out the stock market continuously. Also, the limit order book contains hundreds of price levels which makes our data set has a huge dimensionality. The large amounts of data and high dimensionality make a big challenge for us to do statistical or machine learning analysis. In order to capture the nonlinear relation among the numbers, we try standard neural network in this paper, and our results show our neural network performs much better than other statisitcal or machine learning methods, such as logistic regression. There is a spatial structure in limit order books [1], which means that the sizes and best ask/bid price are highly correlated, and we will show some statistical evidence for this local dependence.

The reason we choose neural network as the method to build the model is that neural network can do the data driven job almost perfectly. Since in our problem, we do not know the specific forms or parameters of our model or distribution, but we do know there must

be some patterns of changes in limit order book. As a result, this is an unsupervised problem. By setting multiple layers and neurons with proper activation functions, a neural network can give us the probability distribution function that describes how best ask/ bid price changes. There are four important factors or features in a limit order book: best ask price, best bid price and their sizes. Intuitively, we believe the best ask price and bid price follows some conditional distribution by given their sizes. And our goal is to predict this distribution of the best ask or bid price at a future specific time spot.

Even though the neural network can capture the nonlinear relations among limit order book, it also has some disadvantages. For a classification standard neural network, the prediction values are totally dependent on the input response values, which means we can not predict any class that is not included in our input data set. This might make some constraints on our result, however, fortunately, the stock price hardly increases by more than hundreds of levels in a extremely short time. A generalized model will performs better [2].

1.1 What is a limit order book (LOB) and how it works?

In some stock markets, traders give their orders with a certain price and shares. Levels are the units that the stock price increasing or decreasing, ususally by \$0.01 or 1 cent. So it is a discrete variable. Size is the number of shares that one order contains For example, someone wants to sell 50 shares of Apple stock with price \$20.22, then the size of this ask order is 50. Also, ask means sell, bid means buy. The best ask price is the lowest sell price of the limit orders to sell in the LOB, the best bid price is the highest buy price of

the limit orders to buy in the LOB. If you check the limit order book, you will find a list that contains hundreds of orders with different price and shares.

The limit orders can be cancelled, orders at the best bid or ask prices can be executed, which may make the price change. If someone buys all the orders with best ask price, this will make the second-best order to be the best ask order at next time spot. Or if someone comes into the market and submit sell limit order with lower price than best ask price, then this new order will become the best ask order. So as to bid orders. Besides this, people might submit buy or sell limit orders at other price level but not the best price level, which can change the structure of the limit order book [3].

Figure 1 shows an example of limit order book at a time spot. In this limit order book, \$180.03 is the best ask price and 1100 is the size according to this price, and \$180.02 is the best bid price. You can find that the last 10 trades make the best ask price increase from \$180.01 to \$180.03. The market orders execute some or all of the orders at best ask or best bid price. It is clear to know the supply and demand in stock market. As a result, predicting the best ask or best price in future by given the current size and levels has great meaning for industry to do property investment or risk management in practice.

1.2 Why neural network?

Deep learning is a very popular type of machine learning and it is built by neural network with multiple layers and numbers of neurons. In this paper we use it to create models that can learn to produce desired output for given input. One significant advantage is multi-layer neural networks can capture nonlinear relations among data and also very suitable for high-dimensional data, which can deal with limit order book perfectly. Of

SPDR S&P 500 ETF TR TR UNIT

Orders Accepted	Total Volume
1,153,586	7,689,062

	TOP OF BOOK		LAST 10 TRADES		
	SHARES	PRICE	TIME	PRICE	SHARES
ASKS ↑	11,000	180.07	14:42:13	180.03	100
	12,500	180.06	14:42:11	180.02	100
	12,900	180.05	14:42:11	180.01	100
	9,700	180.04	14:42:09	180.01	100
	1,100	**180.03**	14:42:09	180.01	200
BIDS ↓	6,400	**180.02**	14:42:08	180.01	100
	9,700	180.01	14:42:06	180.01	100
	9,600	180.00	14:42:06	180.01	100
	14,700	179.99	14:42:06	180.01	100
	11,500	179.98	14:42:06	180.01	100

Figure 1.1. LOB for SP 500 ETF

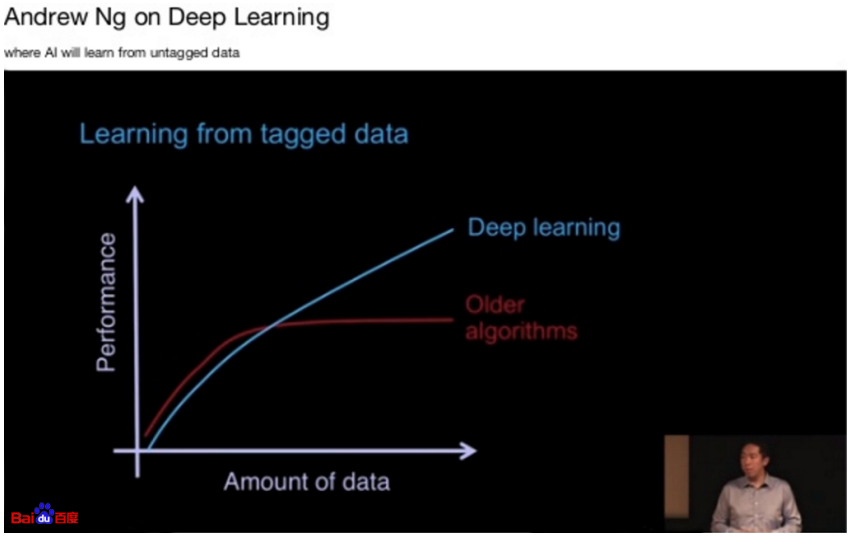

Figure 1.2. Algorithm's perfromance as data quantity increases

course, we can try other machine learning algorithms, logistic regression, support vector machine, random forest,etc. All of them can do both classification and regression. When applied to relatively low-dimensional predictive modeling tasks, deep neural networks do not perform significant accuracy over other algorithms, but as the data amount increase, the performance of deep neural network is much better than others. For example, the logistic regression can solve linear problem perfectly, but can not model nonlinear problem. Random forest can deal with large amount of data and prevent overfitting problem effectively, however, it separates the original data space into rectangulars by setting different thresholds which makes a restricted input function space. The SVM needs kernel transformation to make the data space linear separable by transforming the data into a higher dimension space and it pushes our model into curse of dimentionality. Also, for multi-class problems, SVM solves it by adding up many one-vs-rest classification models together, which is complicated and hard to accomplish. Andrew Ng describes the advantage of deep neural networks over traditional model types as follows

2. Data processing

Data processing is the hard and important part for machine learning. In some extents, it determines the performance of our models. A proper data set makes our learning meaningful and feasible. We download the historical limit order book for one day of 10 different stocks from NASDAQ website. Since the data is collected by milliseconds, there are more than 400 rows of data in each second and more than 1 million rows of data within 7 hours. Considered the computational ability of devices, we just fit the model by the data in 7 hours. It is totally enough to train the neural network and do a nice prediction.

Intuitively, we believe the best ask or best bid price in next state has a local dependence with the last state, which means the best ask or best bid price follows a conditional distribution given the size at last price level. It is easy to understand, since the larger size of current best order is, the lower probability it has that the best ask or best bid price will move to next state. The only thing that makes this happen is some market orders consumes all the volume of current order, otherwise, there always be some order at current lowest sell price or highest buy price [4]. Conditional on the best ask price will increase, $y > 0$, we can establish our model as follows:

$$P[Y = y|Y \geq y] = f(x_y), y > 0$$

If we want to consider the related orders between the best ask prices in current and future states, we should multiply all the probabilities of the orders with intermediate levels.

$$P[Y = y|Y > 0] = f(x_y) \prod_{y'=1}^{y-1} (1 - f(x_{y'})), y > 0$$

Y is the price in future or next time spot and $x_{y'}$ is the size of orders at level $y^{'}$. We will prove it in section 3 by using logistic regression to show the local dependence of limit order book. Besides the current state, the best ask or best bid price in the future should also depend on the limit orders between current best orders and future best orders. Since if we define the future as one second later, all the limit orders that the best ask/bid price skips should be considered. For example, consider the following state of the book at a given time t and $t + \triangle t$.

Ask/Bid	Price	Size	Ask/Bid	Price	Size
Ask	100.11	50	Ask	100.24	100
Ask	100.04	30	Ask	100.22	30
Ask	100.03	50	Ask	100.12	70
Ask	100.00	100	Ask	100.11	50
Bid	99.99	70	Bid	99.96	60
Bid	99.96	70	Bid	99.80	50
Bid	99.94	65	Bid	99.79	20

Table 2.1
Limit order book at time t and t+δt

The best ask price at time t is \$100.00 and best bid price is \$99.99. At time $t + \triangle t$, the best ask price and best bid price are \$100.11 and \$99.96. Let's consider ask limit

orders first, from \$100 to \$100.11, there are two limit orders at price \$100.03 and \$100.04. The best ask price accomplishes this increments, only if market orders consume those two limit orders between them. And in practice, we do not know how many intermediate limit orders among best ask and best bid prices. As a result, to fit the model, we need to record all the state-to-state formats at every time spot that we pick. In Table.1 we can also find there are many zero levels, the levels have zero ask or bid size, such as level \$100.05. To simplify our job, we only record the orders that are 50 levels from current state, no matter it is nonzero levels or zero levels. Because the stock price increases 50 levels in one second is almost impossible. That is why our neural network is a truncated model. As a result, for each nonzero intermediate level, we should report the size of it. To clarify the terminology, the number of shares at level l is the size at level l.

However, to collect this kind of data set requires huge efforts. Since the data we can get easily is the time series data only for best ask and best bid price and their sizes, we can use the event frequency data set which records the detailed information for each transaction to derive all the limit order books at any time spot. But this is a hard work and there are no efficient ways to do this job. Thus we try to mimic this real limit order book by a best price limit order book. Look at table 2.2.

Let's still consider the best ask price, the method is that we record the first best ask price of each second and add 1 level (\$0.01) to this price at one time. Then searching the nonzero levels with this new price and recording their sizes. After this, we take the average of them to be the size at this level. If there are zero levels, just record the size as zero at those levels. We do this iteration untill adding 50 levels to the first best ask price. And repeat this works for each second, because we set $\triangle t$ to be 1 second. For example,

Time	Best ask price	Size at best ask price	Best bid price	Size at best bid price
32400	100.01	80	99.22	100
32400	100.01	50	99.24	50
32400	100.03	70	99.23	70
...	...	...	...	...
32401	98.24	60	97.98	80
32401	98.27	100	97.97	75

Table 2.2
Limit order book of best price

the first best ask price at time 32400 seconds is \$100.01. We record this price and add \$0.01 to it, then searching the price \$100.02 in the same second. Suppose we find several nonzero \$100.02 levels, and we take the average of all the sizes of these levels, recording this average size as the size at level \$100.02 in the true panel limit order book at t. Also, the response value y is the level difference between the price at t and $t + \triangle t$. Thus, part of the final data set is shown in table 2.3.

y	s0	s1	s2	s3	s4	s5	...	s47	s48	s49	s50	s51
7	2695	0	0	0	2	1464	...	0	0	0	0	0
-3	500	0	0	0	0	0	...	0	0	0	0	0
0	400	224	200	0	0	300	...	0	0	0	0	0
-1	474	0	0	0	0	0	...	0	0	0	0	0
0	300	528	0	0	0	0	...	0	0	0	0	0
1	100	1275	0	0	0	0	...	0	0	0	0	0

Table 2.3
Final data set

In this table, s0 is the size of best ask price at time t. s1 to s51 are the sizes of the rest levels following the best ask price at time t, to mimic the data in third columns of Table1. $y = 7$ means the future best ask price is \$0.07 higher than current best ask price. It is reasonable to see a lot of zero levels in this data set, especially when level increases to the price that \$0.5 more than the best ask price. Since the larger the ask price is, the lower influence to best ask price. Why we can use the best price limit order book to mimic the true panel limit order book? Because if the best ask or best bid price keep changing, the new best ask or best bid price must be picked from the orders that have already existed

in the limit order book. No matter the new best price order is the orders that next to the old best price order, or the orders that skip some levels, it depends on the amount of volume that market orders consume. To reduce the errors, we take the average. For each stock, there will be more than fifty thousands samples to train and test the deep neural network model. And we randomly choose 75% of the samples to be the train data set and rest 25% to be the test data set. Our prediction case is: Predicting the distribution of best ask or best bid price at $t + \triangle t$ by given state at t, e.g $P(Y = y|X = x)$.

3. Local dependence

In section 2, we showed an intuitive idea that in a limit order book, there are nonlinear relations among best ask or best bid price and sizes of different levels. Also, we propose that the distribution of future best ask or best bid prices largely depends on the size of the best ask price (or best bid price), and has a relation with the sizes of orders that are close to the best ask orders. The influence of nearby orders will gradually decrease as the levels of nearby orders increase, or best orders are less relevant with further orders which have higher levels {abergel2013mathematical. This phenomenon is called local dependence. In section 3.1 we will use standard neural network to show the nonlinear relations in limit order book. In section 3.2 we will use logistic regression to show the described local dependence.

3.1 Nonlinear relation in limit order book

In this paper, our purpose is to use deep neural network to capture the nonlinear relations in limit order book and this relation helps us to predict the conditional distribution about future prices by given current state's information. We will use an example to show there does exist this nonlinear relation [1]. The probability that the best ask price will increase is highly related with best ask sizes and best bid sizes. It is obvious in reality, since the larger the best ask size, the harder that best ask price reach the next level. Some market orders have to consume all the liquity of current best ask orders, thus

more likely Y equals to y. A large best bid size means the demand of stocks in market is huge, thus the more likely Y will move to levels higher than y. Let's define Y as the price in future and y is current price.

To achieve this, we only pick the nonnegative samples in Table 3 and the features are corresponding best ask and best bid sizes. So the probability we try to estimate is $P(Y > y|Y \geq y, x_y)$, where Y is the best ask price in future for 1-second horizon, y is the current best ask price and x_y is current best bid and ask sizes. In the data set, we use 1 to present increasing movement and 0 to present no movement. Then the data is given in table 3.1.

Movement	Best ask size	Best bid size
1	2695	100
0	400	200
0	300	8200
1	100	6400
...	...	...

Table 3.1
Non-decreasing samples with best ask and best bid sizes

There are more than 13000 samples to train this 4-layer neural network and the model is fitted by using best ask size and best bid size. The probability of increasing movement is shown below in Figure 3.1.

This figure shows a significant concave surface which means there are strong nonlinear relations among the best ask price movement, and the best ask and best bid sizes. As

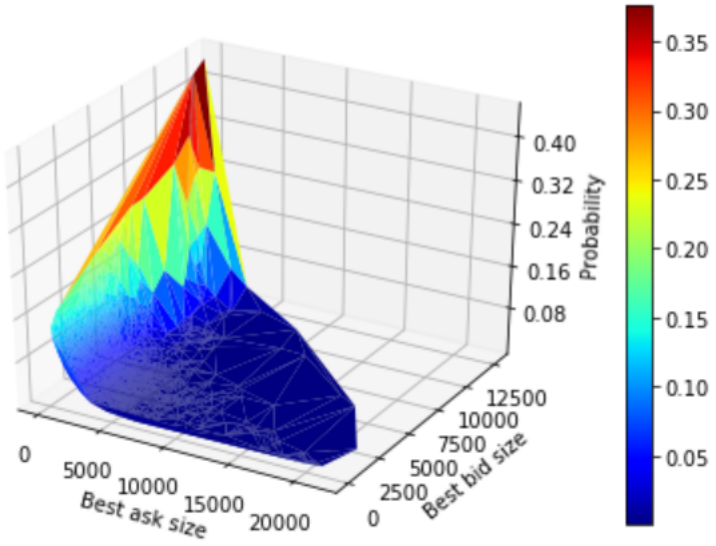

Figure 3.1. Probability that the best ask price increases for a 1-second horizon.

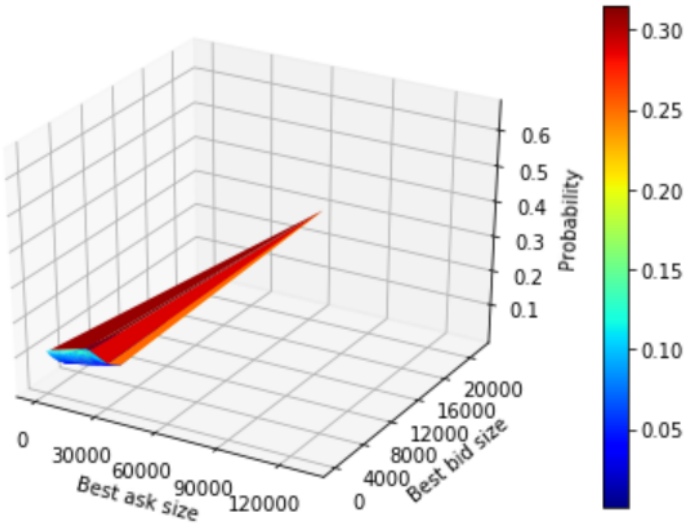

Figure 3.2. Probability that the best ask price decreases for a 1-second horizon.

the best ask size increases, the probability that the best ask price increases is decreasing, since the supply of the stocks is increasing. As the best bid size increases, the probability that the best ask price increases is increasing, since the demand of the stocks is increasing. This graph does not have a smooth surface and cover the whole grids area because we do not have enough data points to plot. But the colors are changing gradually and there are no bumps or outliers which means a good fitting. We do not have enough data for decreasing so the graph of decreasing is not significant.

3.2 Logistic regression for local dependence

In section 3.1 we have shown there are nonlinear relation in limit order book. Now let's prove the limit order book also exhibit a feature called local structure. Local dependence means the influence of nearby orders will gradually decrease as the level increases. Let Y be the best ask price at $t + \triangle t$, time horizon is also 1 second, and y be the current best ask price at t. As what we have proved in section 3.1, the probability of event $Y > y | Y \geq y$ largely depends on the best ask size at level y and might be less dependent on the sizes of other levels. We guess this dependence will decrease as the levels increases. So we modify our Table 3 to be Table 5 in below and 1 represents increasing, 0 represents no-changes. Logistic regression is a method that describes the relationship between the dependent variable (the movement) and a set of independent variables (sizes at different levels). It generates the coefficients of a formula to predict a logit transformation of the probability of presence of the characteristic of interest. In this problem, the model is

$$P(Y > y | Y \geq y) = (1 + \exp(b + \theta \cdot factors))^{-1}$$

Movement	s0	s1	s2	s3	s4	s5	...
1	2695	0	0	0	2	1464.731	...
0	400	224	200	0	0	300	...
0	300	528.99	0	0	0	0	...
1	100	1275	0	0	0	0	...
0	1400	0	0	0	0	0	...

Table 3.2
Non-decreasing samples with different sizes

```
Call:
glm(formula = response ~ ., family = binomial(link = "logit"),
    data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-6.1785  -0.2370  -0.2288  -0.2190   5.2275

Coefficients: (45 not defined because of singularities)
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.442e+00  1.192e-01 -28.887  < 2e-16 ***
Z.0         -8.913e-02  4.183e-05  -2.131   0.0131 *
z            1.157e-03  5.053e-05  22.900  < 2e-16 ***
z.1          1.603e-03  2.940e-04   5.454 4.94e-08 ***
z.2          3.876e-03  9.299e-01   0.042   0.9668
z.3          5.526e+01  4.418e+04   0.001   0.9990
z.4         -5.890e-02  5.910e+01  -0.001   0.9992
z.5                 NA         NA      NA       NA
z.6                 NA         NA      NA       NA
z.7                 NA         NA      NA       NA
z.8                 NA         NA      NA       NA
z.9                 NA         NA      NA       NA
z.10                NA         NA      NA       NA
z.11                NA         NA      NA       NA
z.12                NA         NA      NA       NA
z.13                NA         NA      NA       NA
```

Figure 3.3. Summary for logistic regression

$$g(p(x)) = \ln(\frac{p(x)}{1-p(x)}) = b + \beta_0 s0 + \beta_1 s1 + ... + \beta_{49} s49 + \beta_{50} s50$$

Factors are sizes at different levels, such as s0, s1,...., s50. The coefficients of the model reflects the influence of sizes at different levels, the levels from 0 to 50. And we are fitting a model to show the relation between sizes and increasing of best ask price. The result is shown in Figure 3.3.

From the output of our model we can find there are so many NA, because in our data set, a lot of zero-levels orders appear if levels increase to more than \$0.1. And we also can find that the coefficients at level 3, level 4 and level 5 are not significant, since the p values almost get to 1. The coefficients at level 0, level 1 and level 2 are significant. It means the sizes at those levels are small relative to the increasing of best ask price. And if we plot the coefficients, it is shown in Figure 3.4.

It is apperant that as the level goes up, the coefficients, or the influence of nearby orders are decreasing. The numerical values of coefficients are increasing since we fit a

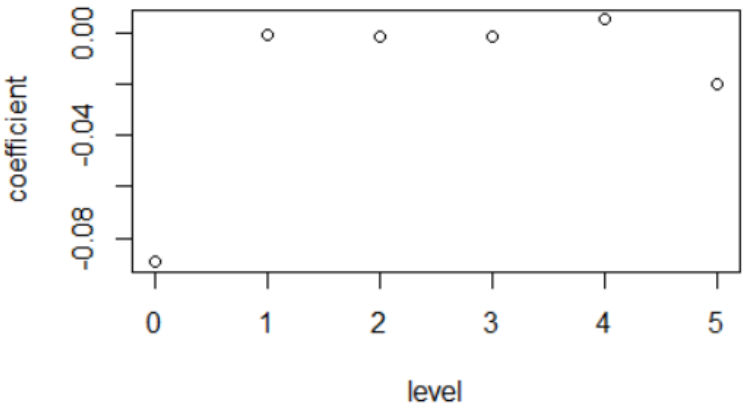

Figure 3.4. Coeffecients of different levels

model to describe best ask price increasing. The size at y has the largest coefficient. To reach level Y, market orders have to consume all the orders at level 0 or level y. And the sizes of levels $y' > y$ are also relative to the distribution of future best ask price. As a result, we get the conclusion that there is local spatial structure in limit order book and the dependence has negative relation with the distance from other limit ask orders to current ask orders.

4. Neural network for prediction

The main idea of this paper is transforming a distribution prediction problem to a standard classification problem. The output of classification neural network is a sequence of probabilities which are generated by an activation function in the output neurons, called softmax function. And this is why we can make a common result for both problems [5]. In order to model this distribution, first we discretize the response space into a discrete space, $R^d = r_{-d/2}, ...r_{-2}, r_{-1}, 0, r_1, r_2, ...$ The limit order book data can perfectly match this space, since the increments of levels are fixed by \$0.01. Also, softmax function will produce a probability distribution of all the levels and we pick the level which has the largest probability among all possible values to be the predicted level, $\hat{Y}$.

In section 4.1, we will review the basic theory of neural network, including the components of a neural network and how to do backpropagation with stochastic gradient descent, also with the early stopping method in cross-validation to reduce overfitting problem. Section 4.2 fits the model with different activation functions and evaluates how our model works by given confusion matrix [6]. Section 4.3 compares our neural network model with other algorithms such as randomforest, kNN and multinomial logistic regression.

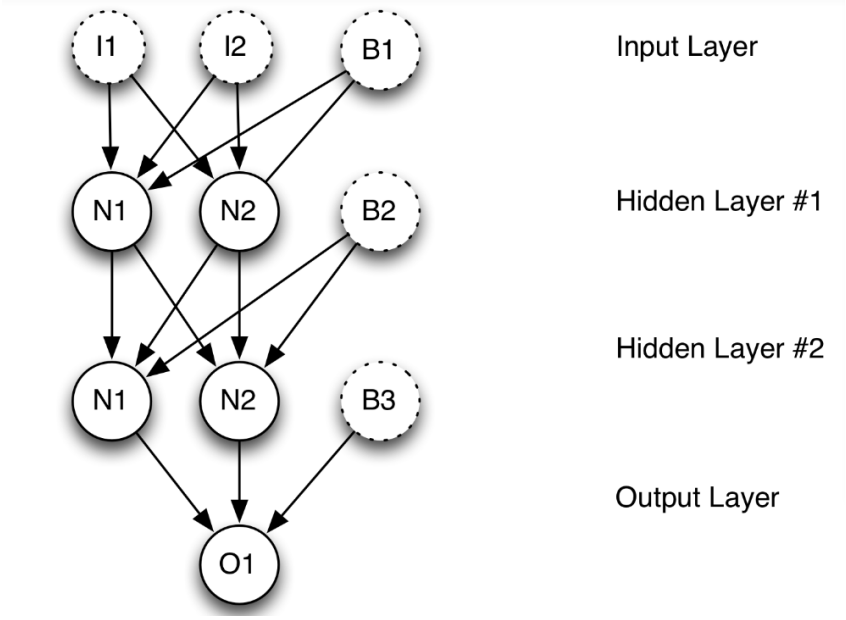

Figure 4.1. Structure of neural network

4.1 Theory of neural network

Neural network is one of the most popular machine learning models in nowadays and we call the models which have more than four layers deep learning [7]. It accepts input and produce output. The basic components of a neural network are layers, neurons and activation functions. Layers include input layer, hidden layers and output layer. The input layer does nothing just send values to conneted neurons in hidden layer. Hidden layers and output layer do some processing jobs. And each layer contains different number of neurons and there are activation functions in each of the neurons. The reason we use neural network to capture the nonlinear relations in limit order book is we can choose nonlinear activation funcitons in hidden layer to process the weights and values. Figure 4.1 is a simple neural networks with two hidden layers.

The input layer contains n neurons that corresponding to the dimension of independent factors, $s_1, s_2, ..., s_{n-1}, s_n$. And the number of hidden layers and their neurons are decided empirically. Usually the performance of the model will increase as the number of layers and neurons increases. However, large number of layers might take a long training time and cause overfitting problem. Overfitting means our network starts to remember the data so performs better and better but not generalized for new data.

A weight vector W_l connects layer to layer and there are bias neuron $b_l$for each hidden layer. The purpose of training the model by iterations is to find the paramters that can best fit our data i.e., the loss function is minimized or log-likelihood function is maximized. The parameters are $\theta = (W_1, W_2, ..., W_L, b_1, ..., b_L)$. Let x be the input data, g is the activation function, then the output of l-th layer is

$$f_{l,\theta}(x) = g(W_l f_{l-1,\theta}(x) + b_l), l = 1, 2, ...L$$

The most popular loss function is

$$L(\theta) = \frac{1}{2}\sum(y - \hat{y})^2$$

For classification problem, we use the log-likelihood function

$$L(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{y\in Y} 1_{y=y_n} log(f_{l,\theta}(x))$$

Activation functions can be chosen from tanh, rectified linear units (ReLU), sigmoid, linear. The activation function for output layer should be softmax in classification problems. The ReLU is

$$\phi(x) = max(0, x)$$

The sigmoid is

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

And the softmax function is calculated by

$$\phi(x) = \frac{e^{z_i}}{\sum_{j=1}^{d_L} e^{z_j}}$$

It is easy to find the softmax function is actually a form of discrete probability. The numerator is the exponential of the output of last layer, i-th neuron. The denominator is the sum of exponential of the output of last layer, all the neurons. And we represent this proportion as probability.

The training of neural network is always combined with Gradient Descent. It is the most general and effective way for optimization over complex functions, especially when there is no closed form for solutions. The basic idea of gradient descent is finding local minimum of the surface of loss function by taking derivatives. To achieve this, we keep making small steps downwards to this surface. And following the direction of the gradient, ∇w.

$$\nabla w = (\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, ... \frac{\partial L}{\partial w_d})$$

However, the gradient descent method only can find local minimum but not the global minimum. Because if the loss surface is not concave, our gradient will stop at the first local minimum. To avoid this, we use stochastic gradient descent(SGD) and usually it is embraced in backpropagation. SDG picks a certain number of samples, called batch, at one time to train the model. After getting the local minimum, it records the result and pick another batch to run again. After a large iterations, our gradient will find a minimum spot of our loss surface which makes us satisfied. This is the global minimum we want. We call the iteration epoch in deep learning. In each iteration, we update the parameters by using the following updating rules

$$\theta_t = \theta_{t-1} - v_t$$

$$v_t = \eta \nabla_{\theta_{t-1}} L(\theta_{t-1})$$

Where t is the index of iterations. ∇ is the gradient, L is the loss function and η is the learning rate to scale the gradient. Setting a proper learning rate is important, since a too low learning rate will usually converge to a good solution but the process will be very slow. Too high learning rate will either fail to converge or converge to a minimum that have high error. Now let's use loss function (4) and sigmoid activation function (6) to briefly show how to do backpropagation. To simplify, we only consider a 1 hidden layer neural network.

Step 1, Initialize the neural network, we set some random parameters as initial values of w.

Step 2, Input the first batch and get the predicted value from output layer. Then compute the error $y - \hat{y}$.

Step 3, Compute the gradient of loss function L with respect to the w backward from output layer to input layer. Let's consider w_{ij}, the weight from hidden node h_i to output node $\hat{y}_j$. We also use those symbols to be the value of their neurons. u is the intermediate input value of hidden node, K is the number of nodes in previous hidden layer,

$$u_j = \sum_{i=1}^{K} w_{ij} h_i$$

First we need to compute

$$\nabla w_{ij} = \frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial u_j} \frac{\partial u_j}{\partial w_{ij}}.$$

If we consider the specific forms of L and u, we have

$$\frac{\partial L}{\partial \hat{y}_j} = y_j - \hat{y}_j$$

$$\frac{\partial \hat{y}_j}{\partial u_j} = \hat{y}_j(1 - \hat{y}_j)$$

$$\frac{\partial u_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_{i=1}^{K} w_{ij} h_i = h_i$$

Then we rewrite it

$$\nabla w_{ij} = \frac{\partial L}{\partial w_{ij}} = (y_j - \hat{y_j})\hat{y}(1 - \hat{y})h_i$$

Now we consider the weights w_{ki} from input layer node to hidden layer node.

$$\nabla w_{ki} = \frac{\partial L}{\partial w_{ki}} = \sum_{j=1}^{M} (\frac{\partial L}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial u_j} \frac{\partial u_j}{\partial h_i}) \frac{\partial h_i}{\partial u_i} \frac{\partial u_i}{\partial w_{ki}}$$

$$h_i = f(u_i) = \frac{1}{1 + e^{-u_i}}$$

$$u_i = \sum_{k=1}^{d} w_{ki} x_d$$

Add them up

$$\nabla w_{ki} = \frac{\partial L}{\partial w_{ki}} = h_i(1 - h_i)x_k \sum_{j=1}^{M} (y_j - \hat{y}_j)\hat{y}(1 - \hat{y})w_{ij}.$$

Step 4, update the weights

$$w_{ij} = w_{ij} - \eta \nabla w_{ij}$$

$$w_{ki} = w_{ki} - \eta \nabla w_{ki}$$

Step5, repeate step 2 to 4 by trying a new batch of data, untill it converge to the error that we accept. The batch is picked randomly.

In this paper, we are training a four-hidden-layer network and each hidden layer contains 20 nodes. Also we use log-likelihood maximum principle instead of loss function. Thus the formula proves will be complicated. Fortunately, we do not have to use those general theory in detail when we fit our model.

4.2 Model Training

To implement our model, we use the packages named tensorflow and sequential in python. Those packages are convenient to establish a neural network with any number of layers and neurons, also with any kinds of activation functions. Also setting the earlystopping function and batchsize in these packages.

We use 4 layer neural network to train our model. The more layers it has, the more complex nonlinear relation it can capture. However, as we said at the first of this section, large number of layers and neurons might cause overfitting and long training time. Thus, our model has four hidden layers and each hidden layer has 20 neurons. The input layer has 51 neurons, because it depends on how many features or factors in our data set. We choose 3 different activation functions (tanh, sigmoid, relu) in hidden layers to find out which function has the shortest running time and highest accuracy. The initial parameters are randomly initialized from normal distribution. And the bathsize is 32 for each iteration of stochastic gradient descent. In machine learning, overfitting is a tough problem for any algorithms. To reduce the influence of overfitting, we use cross validation to record the error rate for each epoch. In each epoch, the data is divided into two data set, training set and testing set. Training set is 95% of all the samples and testing set is 5% of the samples. Both of them are separated randomly every time. Figure 4.2 shows how overfitting happens.

After training for a long time, our model will remember the data and perform better and better. The error rate of training set keeps decreasing while the error of validation set will decrease first but increase after some time. The early stopping method stops the training when the error of validation starts increasing by recording error rate of each

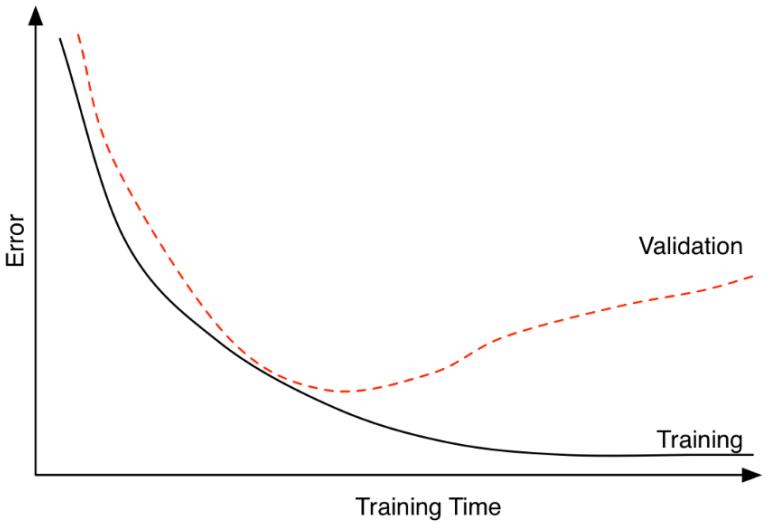

Figure 4.2. Error rate for validation and training

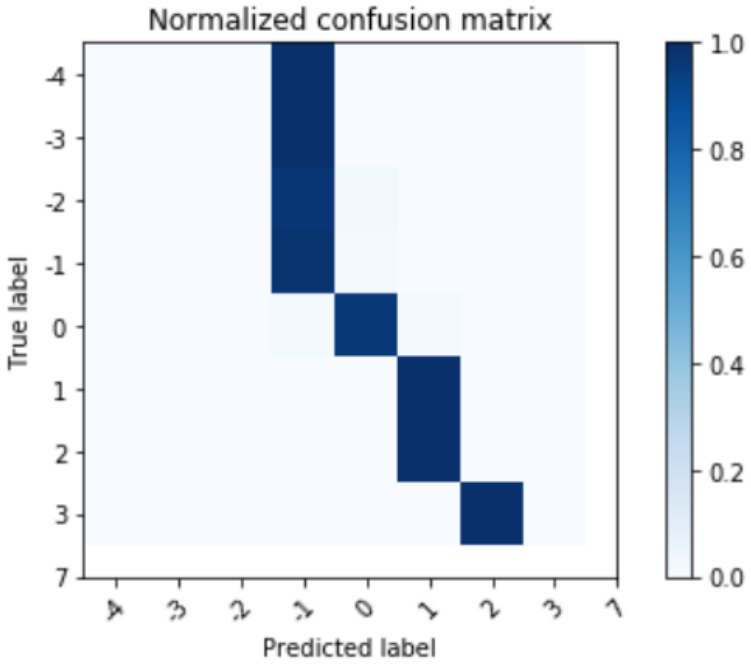

Figure 4.3. Normalized confusion matrix for activation functions tanh

epoch. We train our model by both best ask price and best bid price data but only present the normalized confusion matrix of best ask price. We run each kind of model for ten times and the confusion matrix does not change too much.

The confusion matrix is a common visualization for classification problems. It counts the number of predicting points that compared by predicted labels and corresponding true labels. The normalized confusion matrix is just normalized the numbers for each true label. Thus the more color squares show on diagonal, the better model is. And the color squares on nondiagonal areas mean that there are wrong prediction on certain labels. The color represents the degree of proportion. From those 3 figures we can find neural networks with sigmoid and relu have done a good job. Even though the prediction for levels -2, -3 and -4 performs bad, because we do not have enough data to cover all different levels. Also, the model only could predict levels from -4 to 3 and level 7 but missed all

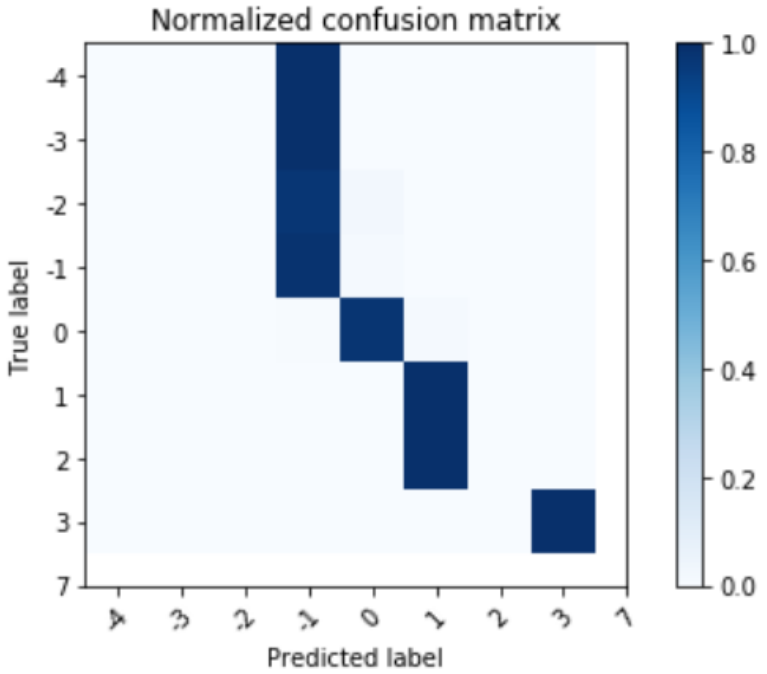

Figure 4.4. Normalized confusion matrix for activation functions sigmoid

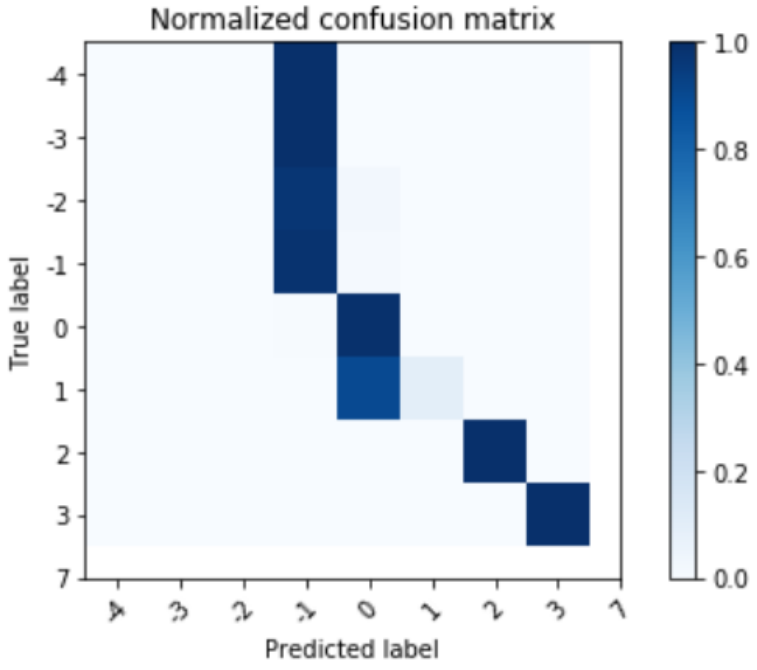

Figure 4.5. Normalized confusion matrix for activation functions relu

other labels. Since the standard neural network generalizes over the input space but does not generalize over output space, which means the class we can predict totally depends on the response values that in our original data set. We could fit the model by larger amount of data that might cover all the levels, or use a generalized neural network [2]. The data we use is from a short time which means in real stock markets, the stock price might be in an increasing trend. So we do not have not enough data for decreasing events. Of course our model can not capture the accurate information of best ask price decreasing. One effective way to improve this is using a longer period data, such as six month or 1 year. Then the data will contain all kinds of variation of stock prices.

In the model with sigmoid activation function, it has a seriously wrong prediction for level 2. The model totally missed the prediction of level 2 and predict level 1 as level 2. The model with relu activation function only has a little wrong predition for level 1. As a result, the best model should be the one with relu as activation function. Besides this, we show the accuracy and training time with the other three algorithms (kNN, random forest and multinomial logistic regression) in next section.

4.3 Other algorithms and comparison

There are many other algorithms which can make prediction for future best ask or best bid price. And in this section we will compare our deep neural network with multinomial logistic regression, kNN and randomforest. Let's briefly explain how these three algorithms work.

kNN is a non-parametric method to do classification or regression. The method is straight forward and take the train set as model. It computes the k nearest data points and decides the label or class by vote majority. We tried k equals to 1, 3 and 5.

Randomforest is the method that combines many decision trees. Decision tree makes classification by choosing a threshold which largely reduce the entropy of the data. It is good at avoiding overfitting, since it introduces uncertainty into the model by taking proportional samples and features to train the model but not using all the data for each time.The accuracy and training time for all above models are shown below The training

Algorithm	tanh Neural Net	sigmoid Neural Net	relu Neural Net	RF
Accuracy	0.9570226	0.95517038	0.957383	0.93031066
Training time (Seconds)	10.9674423	11.2387965	10.0457821	1.398726
Algorithm	RF	1kNN	3kNN	5kNN
Accuracy	0.93031066	0.9185558	0.9401063	0.9437447
Training time (Seconds)	1.398726	0.875943	0.893245	0.8948721

Table 4.1
Accuracy and running times of all 7 models

time of kNN and random forest is much less than Neural network. Since the models are much simpler than deep learning model. All of these models performs well. As k increases from 1 to 5, the accuracy of kNN also increases but needs a longer training time. The accuracy of three different neural networks are really close but relu has a shorter training time. Since the function of relu is $max(0, x)$, much easier to apply than sigmoid and

tanh. Both of them are nonlinear functions but sigmoid and tanh have more complex structure. Overall, neural network beats other algorithms in accuracy.

5. Conclusion

This paper developes a deep neural network to capture the nonlinear relations among limit order book and make a prediction of probability distribution of future best ask or best bid price. There are nonlinear relations between the best ask or best bid price and sizes of nearby orders, especially the current state orders' sizes. The influence of nearby orders will gradually decrease as the level increases. The probability of increasing or decreasing of stock prices largely depends on supplying and demands. If the sizes of best ask orders are large, then it means there is a high supply of stocks in market, and the probability of stock price increasing is low. The performance of deep neural network is much better than logisitic regression, kNN and random forest. Because our model has deep structures and several nonlinear activation functions to get more information. The deep neural network model has a high accuracy to predict the future best order prices which means it capture a rich information from limit order book. Other algorithms can not deal with a nonlinear problem perfetctly. The earlystopping method effectively stops the overfitting problem.

We also have two main problems which might make our model not perfect. One is the data that available to us. Collection of all the limit order books by seconds in a long period is so difficult. Then we have to use best limit order books to mimic the original limit order book. It must causes some errors. And the devices of computing determine that we can not fit the model by huge amounts of data. And the data in

a short period performs significant trending component, which means the stock price increasing or decreasing data might be not enough. In this paper, we do not have too many data that represent decreasing of best ask or best bid price. The other problem is that our model is not generalized on space of response values. It means we can not get a distribution covers a complete range of levels [1].

Despite this, our model shows high accuracy and really suitable for prediction applications in real stocks market.

REFERENCES

[1] Justin Sirignano. Deep learning for limit order books. 2016.

[2] Aristidis Likas. Probability density estimation using artificial neural networks. *Computer physics communications*, 135(2):167–175, 2001.

[3] Xin Guo, Adrien De Larrard, and Zhao Ruan. Optimal placement in a limit order book: an analytical approach. *Mathematics and Financial Economics*, 11(2):189–213, 2017.

[4] Marco Avellaneda, Josh Reed, and Sasha Stoikov. Forecasting prices from level-i quotes in the presence of hidden liquidity. *Algorithmic Finance*, 1(1):35–43, 2011.

[5] Matthew Dixon. Sequence classification of the limit order book using recurrent neural networks. *Journal of Computational Science*, 2017.

[6] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *Business Informatics (CBI), 2017 IEEE 19th Conference on*, volume 1, pages 7–12. IEEE, 2017.

[7] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.