DeepLOB: Deep Convolutional Neural Networks for Limit Order Books

Under The Guidance Of
Prof. Rajive Bhutani

Name: Oindrila Mukherjee
Roll no : Umds19003
Subject : Financial Analytics
Batch:2019-2021
Course: Mtech-DSA

Abstract

A deep learning approach has been adopted in this paper where a deep learning model is concocted to predict the price movement of the limit order book data of cash equities. Convolution neural networks are applied in this paradigm. Convolution filters are used to apprehend the spatial structures of LOB. Further , LSTM modules are used to encapsulate time dependencies. The dataset used in this state-of-art is the benchmark LOB dataset . The paradigm outperforms in this dataset . The dataset is divided into two parts one is a training set and the other a test set . Both the training and testing of the paradigm is performed in the same dataset . Optimiser is also used to check upon the optimisation of the algorithm and also ensured that the model runs with utmost accuracy.The ability to extract robust features which translate well to other instruments is an important property of our model which has many other applications.

Introduction

More than half of today's dynamic financial landscape uses digital LOB to record the trades. In traditional quote-driven marketplaces, traders can only buy or sell an asset at one of the prices , which were made public by market makers.But in recent years , traders now ,can directly view all resting limit orders in the limit order book of an exchange. Just because, limit orders are arranged into different levels based on their submitted prices, the evolution in time of a LOB showcases a multi-pronged issue with elements representing the numerous prices and order volumes/sizes at multiple levels of the LOB on both the buy and sell sides. LOB , itself is a labyrinthine dynamic framework with high dimensionality. Thus, inducing modelling complications that make traditional methods much difficult to cope up with. Paradigms of transmogrifying price sequences usually dominate over mathematical finance resulting in a range of Markov-like models with stochastic driving terms, such as the vector autoregressive model (VAR) or the autoregressive integrated moving average model (ARIMA) . These models, to avoid excessive parameter spaces, often rely on handcrafted features of the data. However, given the billions of electronic market quotes that are generated everyday, it is natural to employ more modern data-driven machine learning techniques to extract such features.In addition, limit order data, like any other financial time series data is notoriously non-stationary and dominated by stochastics. In particular, orders at deeper levels of the LOB are often placed and cancelled in anticipation of future price moves and are thus even more prone to noise. Other problems, such as auction and dark pools , also add additional difficulties, bringing ever more unobservability into the environment. The paper is followed with deep neural networks architecture that incorporates both convolutional layers as well as Long Short-Term Memory (LSTM) units to predict future stock price movements in large-scale high-frequency LOB data. The advantage of the concerned paradigm over previous research is that it has the ability to adapt for many stocks by extracting representative features from highly noisy data.In order to overcome the limitations of handcrafted features, the concept of   Inception Module is used to wrap convolutional and pooling layers together. The Inception Module helps to infer local interactions over different time horizons. The resulting feature maps are then passed into LSTM units which can capture dynamic temporal behaviour. The testing of the model on a publicly available LOB dataset, known as FI-2010 , and this method remarkably outperforms all existing state-of-the art algorithms. However, the FI-2010 dataset is only made up of 10 consecutive days of down-sampled pre-normalised data from a less liquid market.. The model delivers robust out-of-sample prediction accuracy across stocks over a test period of three months.Its observed that it projects not only the capability of the proposed paradigm to be able to extract robust features from order books, but also indicates the existence of universal features in the order book that modulate stock demand and price. The ability to transfer the model to new instruments opens up a number of possibilities that we consider for future work.

Background

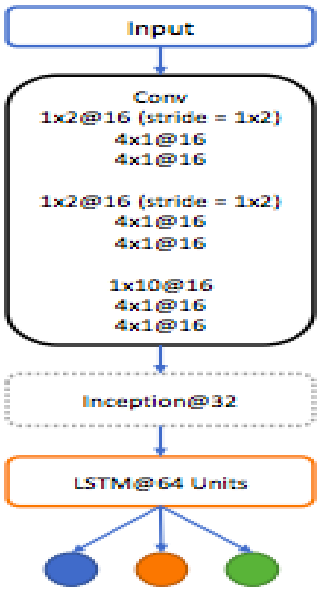
Predictability of stock markets is already in the whirlpool of several research with a long history in the field of financial literature . Although there have been umpteen opinions regarding the efficiency of markets, out of which most of it are   accepted widely. Studies show that financial markets are to some extent predictable . Two major classes of work which attempt to forecast financial time-series are, broadly speaking, statistical parametric models and data-driven machine learning approaches . Traditional statistical methods generally assume that the time-series under study are generated from a parametric process . There is, however, agreement that stock returns behave in more complex ways, typically highly nonlinearly . Machine learning techniques are able to capture such arbitrary nonlinear relationships with little, or no, prior knowledge regarding the input data . Recently, there has been a surge of interest to predict limit order book data by using machine learning algorithms . Among many machine learning techniques, pre-processing or feature extraction is often performed as financial time-series data is highly stochastic. Generic feature extraction approaches have been implemented, such as the Principal Component Analysis (PCA) and the Linear Discriminant Analysis (LDA) in the work of . However these extraction methods are static pre-processing steps, which are not optimised to maximise the overall objective of the model that observes them. In the work of , the Bag-of-Features model (BoF) is expressed as a neural layer and the model is trained end-to-end using the backpropagation algorithm, leading to notably better results on the FI-2010 dataset . These works suggest the importance of a data driven approach to extract representative features from a large amount of data. In our work, we advocate the end-to-end training and show that the deep neural network by itself not only leads to even better results but also transfers well to new instruments (not part of the training set) - indicating the ability of networks to extract "universal" features from the raw data. Arguably, one of the key contributions of modern deep learning is the addition of feature extraction and representation as part of the learned model. The Convolutional Neural Network (CNN) is a prime example, in which information extraction, in the form of filter banks, is automatically tuned to the utility function that the entire network aims to optimise. CNNs have been successfully applied to various application domains, for example, object tracking , object-detection and segmentation . However, there have been but a few published works that adopt CNNs to analyse financial microstructure data and the existing CNN architectures are rather unsophisticated and lack thorough investigation. Just like when moving from "AlexNet" to "VGGNet" , we show that a careful design of network architecture can lead to better results compared with all existing methods. The Long Short-Term Memory (LSTM) was originally proposed to solve the vanishing gradients problem of recurrent neural networks, and has been largely used in applications such as language modelling and sequence to sequence learning . Unlike CNNs which are less widely applied in financial markets, the LSTM has been popular in recent years, all utilising LSTMs to analyse financial data. In particular, uses limit order data from 1000 stocks to test a four layer LSTM model. Their results show a stable out-of-sample prediction accuracy across time, indicating the potential benefits of deep learning methods. To the best of our knowledge, there is no work that combines CNNs with LSTMs to predict stock price movements and this is the first extensive study to apply a nested CNN-LSTM model to raw market data. In particular, the usage of the Inception Model in this context is novel and is essential in inferring the optimal “decay rates” of the extracted features.

Dataset

FI2010 benchmark dataset is used in this state -of-the-art . It is divided into two parts . The first part being the training part which consists of 362400 rows and 149 columns and the second part ,the testing part consists of rows of 31937 rows and 149 columns . The first 40 columns of the FI-2010 dataset are 10 levels ask and bid information for a limit order book and we only use these 40 features in our network.

Model-architecture

The detail of the network architecture, which encompasses three main building blocks: standard convolutional layers, an Inception Module and a LSTM layer, as shown in Figuredown below. The main idea of using CNNs and Inception Modules is to automate the process of feature extraction as it is often difficult in financial applications since financial data is notoriously noisy with a low signal-to-noise ratio. Technical indicators such as MACD and the Relative Strength Index are included as inputs and preprocessing mechanisms such as principal component analysis (PCA) are often used to transform raw inputs. However, none of these processes is trivial, they make tacit assumptions and further, it is questionable if financial data can be well-described with parametric models with fixed parameters. In this work, the only requirement is the history of LOB prices and sizes as inputs to the algorithm. Weights are learned during inference and features, learned from a large training set, are data-adaptive, removing the above constraints. A LSTM layer is then used to capture additional time dependencies among the resulting features. It is noted that very short time dependencies are already captured in the convolutional layer which takes “space-time images” of the LOB as inputs.



a) Convolutional Layer

Recent development of electronic trading algorithms often submit and cancel vast numbers of limit orders over short periods of time as part of their trading strategies . These actions often take place deep in a LOB and it is seen that more than 90% of orders end in cancellation rather than matching, therefore practitioners consider levels further away from best bid and ask levels to be less useful in any LOB. In addition, the work suggests that the best ask and best bid (L1-Ask and L1-Bid) contribute most to the price discovery and the contribution of all other levels is considerably less, estimated at as little as 20%. As a result, it would be otiose to feed all level information to a neural network as levels deep in a LOB are less useful and can potentially even be misleading. Naturally, we can smooth these signals by summarising the information contained in deeper levels. We note that convolution filters used in any CNN architecture are discrete convolutions, or finite impulse response (FIR) filters, from the viewpoint of signal processing. FIR filters are popular smoothing techniques for denoising target signals and they are simple to implement and work with.We can write any FIR filter in the following form:
y(n) = X M k=0 bkx(n − k)
 where the output signal y(n) at any time is a weighted sum of a finite number of past values of the input signal x(n). The filter order is denoted as M and bk is the filter coefficient. In a convolutional neural network, the coefficients of the filter kernel are not obtained via a statistical objective from traditional signal filtration theory, but are left as degrees of freedom which the network infers so as to extremise its value function at output. The details of the first convolutional layer inevitably need some consideration. As convolutional layers operate a small kernel to “scan” through input data, the layout of limit order book information is vital. Let's recall that we take the most 100 recent updates of an order book to form a single input and there are 40 features per time stamp, so the size of a single input is (100 × 40). We organise the 40 features as following:

{p (i) a (t), v(i) a (t), p (i) b (t), v (i) b (t)} n=10 i=1
where i denotes the i-th level of a limit order book. The size of our first convolutional filter is (1 × 2) with stride of (1 × 2). The first layer essentially summarises information between price and volume {p (i) , v(i)} at each order book level. The usage of stride is necessary here as an important property of convolutional layers is parameter sharing. This property is attractive as less parameters are estimated, largely avoiding overfitting problems. However, without strides, we would apply the same parameters to {p (i) , v(i)} and {v (i) , p(i+1)}. In other words, p (i) and v (i) would share the same parameters because the kernel filter moves by one step, which is obviously wrong as price and volume form different dynamic behaviors. Because the first layer only captures information at each order book level, we would expect representative features to be extracted when integrating information across multiple order book levels. We can do this by utilising another convolutional layer with filter size (1 × 2) and stride (1 × 2). The resulting feature maps actually form the micro-price .

p micro price = Ip(1) a + (1 − I)p (1) b I = v (1) b v (1) a + v (1) b .

The weight I is called the imbalance. The micro-price is an important indicator as it considers volumes on bid and ask side, and the imbalance between bid and ask size is a very strong indicator of the next price move. This feature of imbalances has been reported by a variety of researchers . Unlike the micro-price where only the first order book level is considered, we utilise convolutions to form micro prices for all levels of a LOB so the resulting features maps are of size (100, 10) after two layers with strides. Finally, we integrate all information by using a large filter of size (1×10) and the dimension of our feature maps before the Inception Module is (100, 1). We apply zero padding to every convolutional layer so the time dimension of our inputs does not change and Leaky Rectified Linear Units (Leaky-ReLU) are used as activation functions. The hyper-parameter (the small gradient when the unit is not active) of the Leaky-ReLU is set to 0.01, evaluated by grid search on the validation set. Another important property of convolution is that of equivariance to translation . Specifically, a function f(x) is equivariant to a function g if f(g(x)) = g(f(x)). For example, suppose that there exists a main classification feature m located at (xm, ym) of an image I(x, y). If we shift every pixel of I one unit to the right, we get a new image I 0 where I 0 (x, y) = I(x − 1, y). We can still obtain the main classification feature m0 in I 0 and m = m0 , while the location of m0 will be at (xm0 , ym0) = (xm −1, ym). This is important to time-series data, because convolution can find universal features that are decisive to final outputs. In our case, suppose a feature that studies imbalance is obtained at time t. If the same event happens later at time t 0 in the input, the exact feature can be extracted later at t 0 . We do not use any pooling layer except in the Inception Modules. Although pooling layers help us find representations invariant to translations of the input, the smoothing nature of pooling can cause under-fitting. Common pooling layers are designed for image processing tasks, and they are most powerful when we only care if certain features exist in the inputs instead of where they exist . Time-series data has different characteristics from images and the location of representative features is important. Our experiences show that pooling layers in the convolutional layer, at least, cause under-fitting problems to the LOB data. However, we think pooling is important and new pooling methods should be designed to process time-series data as it is a promising solution to extract invariant features.

b) Inception Module

We note that all filters of a standard convolutional layer have fixed size. If, for example, we employ filters of size (4 × 1), we capture local interactions amongst data over four time steps. However, we can capture dynamic behaviours over multiple timescales by using Inception Modules to wrap several convolutions together. We find that this offers a performance improvement to the resultant model. The idea of the Inception Module can be also considered as using different moving averages in technical analysis. Practitioners often use moving averages with different decay weights to observe time-series momentum . If a large decay weight is adopted, we get a smoother time-series that well represents the long-term trend, but we could miss small variations that are important in high-frequency data. In practice, it is a daunting task to set the right decay weights. Instead, we can use Inception Modules and the weights are then learned during back-propagation. In our case, we split the input into a small set of lower dimensional representations by using 1 × 1 convolutions, transform the representations by a set of filters, here 3 × 1 and 5 × 1, and then merge the outputs. A max-pooling layer is used inside the Inception Module, with stride 1 and zero padding. "Inception@32" represents one module and indicates all convolutional layers have 32 filters in this module. The 1 × 1 convolutions form the Network-in-Network approach. Instead of applying a simple convolution to our data, the Network-in-Network method uses a small neural network to capture nonlinear properties of our data. We find this method to be effective and it gives us an improvement on prediction accuracy.

c) LSTM Module

In general, a fully connected layer is used to classify the input data. However, all inputs to the fully connected layer are assumed independent of each other unless multiple fully connected layers are used. Due to the usage of Inception Module in our work, we have a large number of features at the end. Just using one fully connected layer with 64 units would result in more than 630,000 parameters to be estimated, not to mention multiple layers. In order to capture temporal relationships that exist in the extracted features, we replace the fully connected layers with LSTM units. The activation of a LSTM unit is fed back to itself and the memory of past activations is kept with a separate set of weights, so the temporal dynamics of our features can be modelled. We use 64 LSTM units in our work, resulting in about 60,000 parameters, leading to 10 times fewer parameters to be estimated. The last output layer uses a softmax activation function and hence the final output elements represent the probability of each price movement class at each time step.

Experiment Settings

We apply the same architecture to all our experiments in this section and the proposed model is denoted as DeepLOB. We learn the parameters by minimising the categorical cross entropy loss. The RMSprop, is utilised and we set the parameter "epsilon" to 1 and the learning rate to 0.01. The learning is stopped when validation accuracy does not improve for 20 more epochs. This is about 50 epochs for the FI-2010 dataset and 10 epochs for the test part of the dataset.

Result

An epoch of 50 is used in this experiment . However after 10 epochs the metric of accuracy however has almost constant . For each iteration and its respective epochs the elevation of the accuracy is like a slope . The accuracy rate epoch 1 attained minimum accuracy of all . It's around 64 percentile . With gradual iteration the accuracy in the training set ended up to 80 percentile. However the values were nearly same after the twentieth epoch.

Conclusion

In this paper, we introduce the first hybrid deep neural network to predict stock price movements using high frequency limit order data. Unlike traditional hand-crafted models, where features are carefully designed, we utilise a CNN and an Inception Module to automate feature extraction and use LSTM units to capture time dependencies.In a recent extension of this work we have modified the DeepLOB model to use Bayesian neural networks . This allows to provide uncertainty measures on the network's outputs which for example can be used to upsize positions .In subsequent continuations of this work we would like to investigate more detailed trading strategies, using Reinforcement Learning, which are based on the feature extraction performed by DeepLOB.

Reference

[1] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods," Journal of Forecasting, vol. 37, no. 8, pp. 852– 866, 2018.

[2] C. A. Parlour and D. J. Seppi, "Limit order markets: A survey," Handbook of financial intermediation and banking, vol. 5, pp. 63–95, 2008.

[3] I. Rosu et al., "Liquidity and information in order driven markets," Tech. Rep., 2010.

[4] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," Modeling Financial Time Series with S-PLUSR , pp. 385–429, 2006.

[5] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the ARIMA model," in Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference on. IEEE, 2014, pp. 106– 112