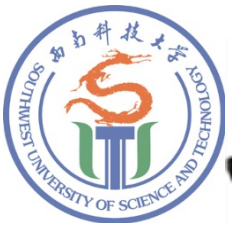


西南科技大学

Southwest University of Science and Technology

本科毕业设计（论文）

题目名称：高频金融交易中的价格变动预测建模及实现

学院名称	计算机科学与技术学院
专业名称	计算机科学与技术
学生姓名	李若昊
学号	5120180269
指导教师	董万利 讲师

二〇二二年六月

西南科技大学

本科毕业设计（论文）学术诚信声明

本人郑重声明：所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日期：　　年　月　日

西南科技大学

本科毕业设计（论文）版权使用授权书

本毕业设计（论文）作者同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南科技大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本毕业设计（论文）。

本论文属于

保密□，在__年解密后适用本授权书。

不保密□。

（请在以上方框内打“√”）

作者签名：　　指导教师签名：

日期：　年 月 日　　日期：　年 月 日

高频金融交易中的价格变动预测建模及实现

摘要：股指期货价格的波动是盈利的根本来源，人们通常通过对模式的学习来预测股指期货走势，调整投资策略。本文针对高频金融交易中的价格变动预测存在数据体量大且价格可变性和不确定性强等问题，建立了长短期记忆循环神经网络模型、卷积神经网络模型、长短期记忆循环神经网络与卷积神经网络的融合模型以及空洞卷积模型等四种模型，并对它们进行了参数择优。在经过对参数调优得到最优的模型结构及参数后，在 FI-2010 数据集上进行了实测，判断未来 10 个周期内股票价格的变化方向，预测精度达到近 90%，并对使用四种模型得到的预测效果进行了对比分析。另外，本文还进行了迁移学习、使用随机森林进行特征提取后再预测以及解释深度神经网络预测原理三项工作。

关键词：高频交易；股价预测；算法交易；时间序列；神经网络

Modelling of price movement forecasting in high frequency financial trading

Abstract: The volatility of security prices is a fundamental source of profitability and people often learn from patterns to predict security price movements and adjust their investment strategies. Based on the analysis of prediction methods suitable for this topic, the deep neural network was selected as the prediction method, and a long and short-term memory recurrent neural network model, a convolutional neural network model, a fusion model of long and short-term memory recurrent neural network and convolutional neural network, and a dilated convolutional model were developed for the stock price prediction problem in this thesis. After tuning the parameters to obtain the optimal model structure and parameters, this thesis uses the optimal model of each of the four models obtained from the tuning experiments to conduct a real-world test to determine the direction of change in the median stock price over the next 10 cycles. The results far exceeded the requirements of the assignment and a comparative analysis of the forecasting results obtained using the four models was subsequently carried out. Additional work on transfer learning, retraining and testing using random forests for feature extraction and explaining the principles of deep neural network prediction using convolutional neural networks as an example are also carried out.

Key words: High-frequency trading; stock price prediction; algorithmic trading; time series; Neural Network

目 录

第 1 章 绪论

1.1 研究背景和意义

股指期货价格的波动是盈利的根本来源，人们通常通过对模式的学习来预测股指期货走势，调整投资策略。如今的高频量化交易使得在几分之一秒内发生数以千计的交易成为现实，随之而来的是几乎无限的可以利用价格差来实时计算的机会。在这个时间粒度上，订单簿实际上已经提供了足够多的可以对未来价格做出较准确预测的信息。例如，买卖订单数量之间的不对称或订单数量、价格的突发变化都可能会对预测价格走势有帮助。可靠的预测结果对于希望规避逆向选择风险的交易者来说很重要，同时对于试图甄别非法交易活动的监管机构也将具有一定的参考意义。

1.2 国内外研究现状

作者使用 Scopus 分别检索了近四年引用数多和相关性强的文献，以寻求热度高、重要性强且适合于本研究课题的文章来作为接下来本文工作的路标。同时，下述文献也包含少数来自 arXiv-sanity 等网站上发布的还未经同行评议但有价值的文章。下面作者将按照时间顺序列出贴合于本文研究问题且文献作者具有一定影响力、期刊层次较高的研究成果。

Xingyu Zhou 等人在 2018 年 2 月 13 日的文章[1]中采用了长短期记忆循环神经网络（Long Short-Term Memory, LSTM）和卷积神经网络（Convolutional neural networks, CNNs）来进行对抗性训练以预测股票市场。作者注意到，他们在向模型输入训练集和测试集的时候采用了滑动窗口的手法，这启发了作者在进行本文后续所述的实验时也可以使用类似的手段。Bruno Miranda Henrique 等人在 2018 年 4 月 20 日的文章[2]中采用了支持向量回归（Support Vector Regression, SVR）技术预测了间隔时间为一分钟的股价。这篇文章的作者通过把他们的模型与有效市场假说提出的随机游走模型进行比较，得出了 SVR 确实拥有预测能力的结论，且同时还指出如果可以周期性更新这个模型的话，效果会更好。Faisal Qureshi 等人在 2018 年 12 月 20 日的文章[3]中评估了多个机器学习方法中的分类器，探索了不同的分类器在预测股票短期价格变化方面的潜力。他们通过实验发现随机森林表现最佳。

Zihao Zhang 等人在 2019 年 3 月 25 日的文章[4]中利用了 CNN 和 Inception 模块来自动提取特征，并使用 LSTM 单元来捕获输入之间的时间依赖性。他们提出的这种方法在为期三个月的实测中准确度一直很高且预测效果稳定。值得注意的是，这篇文章的

作者还对自己提出的模型进行了敏感度分析，定位出了模型中贡献准确度最大的部分。这激发了作者在后续的实验中也希望挑选恰当的方法，如显著图（Saliency map）、类激活图（Class Activation Map）或反褶积（Deconvolution）来解释本文所提出的模型的输出原理的想法。

AH Bukhari 等人在 2020 年 3 月 23 日的文章[5]中提出了一个基于自回归分数积分移动平均（Autoregressive Fractionally Integrated Moving Average, ARFIMA）模型和 ARFIMA 残差的 LSTM 模型的“混合 ARFIMA-LSTM”模型，来预测金融市场突发的变化。这个模型克服了神经网络过拟合的问题，且从实验结果来看优于单独的差分整合移动平均自回归（Autoregressive Integrated Moving Average , ARIMA）模型、ARFIMA 模型和广义回归神经网络（General regression neural network , GRNN）。在均方根误差指标上，其准确度与传统的预测模型相比提高了有约 80%。我们知道，在利用 LSTM 预测普通股市时，考虑到股市类型的因素，观测点会比较少，这导致了即便采用正则化也极可能出现过拟合的问题。作者希望在本文所述的实验若遇到相似的问题，可以借用这篇文章的经验得到启示。Adamantios Ntakaris 等人在 2020 年 5 月 19 日的文章[6]中提供了三组共 270 个适合高频交易（High-frequency trading , HFT）的手工特征，并且将它们合并到了 FI-2010 数据集中，考察了它们在短期中间价格变动方向预测问题中的有效性。实验结果表明，添加这篇文章所描述的手工特征后，可以提高分类器的性能。Zineb Lanbouri 等人在 2020 年 8 月 6 日的文章[7]中基于高频历史价格数据和自行计算的技术指标，用 LSTM 预测了标准普尔 500 指数中单只标的未来 1、5、10 分钟内的价格。文章的作者通过实验证明了他们提出的方法可以在不使用技术指标的情况下来提前 10 分钟、5 分钟或 1 分钟有效地预测收盘价。此外作者还指出，他们认为 LSTM 在这个方面问题上非常有应用前景，且有意改进他们的模型以实现在线预测。YuChen Tu 在其 2020 年 8 月 9 日的硕士学位论文[8]中探索了含滞后特征的前馈神经网络（Feedforward neural network , FNN）、不含滞后特征的 FNN、含 LSTM 层的多对一循环神经网络（Recurrent neural networks，RNNs）和含 LSTM 层的多对多 RNN 中更适合预测未来中间价变化方向的网络结构。作者通过大量的参数调整和模型训练，发现含 LSTM 层的多对多 RNN 的性能优于其他三个模型结构。此外，作者还发现了 RNN 的性能并不会随着时间步长（time step）的增加而线性增加。本文根据这篇文章带来的启发，设置了实验中欲搭建的模型类别。Yuechun Gu 等人在 2020 年 10 月 19 日的文章[9]中先用实验证明了传统的广义自回归条件异方差模型（Generalized AutoRegressive Conditional

Heteroskedasticity , GARCH）比简单的 LSTM 效果好得多，接着用从模型对比中得到的启发针对 LSTM 在此领域的应用提出了些许建议，包括应该使用自回归 RNN、将一些技术指标看作协方差一起作为模型输入，以及将目标看作价格本身的回归问题而非二元分类问题会更合适等。作者也展示了将 LSTM 做了上述改进之后，其性能有着大幅提升，且可以优于 GARCH。

Konark Yadav 等人于 2021 年 5 月 18 日的文章[10]中提出了一个基于 FastRNN 的实时股价预测模型，它可以提供比除 ARIMA 和 FBProphet 外的所有模型更快的预测速度。并且，该模型还在均方根误差（Root Mean Square Error , RMSE）指标上比 LSTM 表现得更好。Muye Wang 在其 2021 年 6 月 16 日的博士学位论文[11]中使用了一个基于 RNN 的方法估计了成交时间的分布，又基于动态限价订单簿（Limit Order Book, LOB）使用强化学习预测了价格。文章的作者发现，模型对单一股票学习到的模式可以泛化到对其他股票的预测里。这激发了作者在本文的实验中希望尝试迁移学习的想法。Yue Yang 等人在 2021 年 6 月 21 日的文章[12]中利用 XGBoost 模型和 LightGBM（Light Gradient Boosting Machine）模型实现了对股价的预测。其实验结果表明，XGBoost 和 LightGBM 的组合模型比它们二者中任何一个单一模型以及其它神经网络模型都具有更好的预测效果。Liang Zeng 等人在 2021 年 7 月 26 日的文章[13]中针对价格走势预测问题提出了一个新的结构 LARA（Locality-Aware Attention and Adaptive Refined Labeling）。他们通过 LA-Attention 来提取可能盈利的样本，解决了低信噪比和金融数据高随机性带来的问题。该模型的性能在实测中优于传统时间序列分析方法和一些基于机器学习的方法。作者通过大量的实验证明 LARA 确实捕捉到了更有价值的交易点。我们知道，股价预测问题的难点之一就在于股价数据低信噪比较低，这篇文章给本文处理数据的低信噪比问题带来了思路。Xuerui Lv 和 Li Zhang 在他们 2021 年 8 月 7 日的论文[14]中针对预测 LOB 的价格变动问题提出了一个可以更好地学习 LOB 中时间特征的 SRGRU（Stacked Residual Gated Recurrent Unit）网络。作者通过在 FI-2010 数据集上进行大量实验，证明了 SRGRU 在准确度和 F-score 指标上均优于 DeepLOB（Deep Convolutional Neural Networks for Limit Order Books）模型，且与后者相比，前者平均执行时间更短、参数更少且泛化能力更强。

目前，我国内地对于高频交易的监管基本上是空白的，它在内地资本市场中被看作是一个神秘的舶来品。本文的目的之一即是希望通过对世界其他地方已有的研究成果进行探索，进而给我国监管机构一些启发。

1.3 本文研究工作

本文所述实验的首要目的为使用本文经过 3.3 节模型调优后得到的模型，预测单只股票在未来数个周期内的中间价格变化方向，且预测效果不低于数据论文[15]中给出的基线准确率及 F1 分数。为达成目标，作者首先在本文 3.2 节中构建了四种预测模型，而后使用某五只股票连续六个交易日的十级限价订单簿数据对它们在 3.3 节中进行了大量调参；在得到调优后的价格变化方向分类器后，作者使用它们完成了预测任务。在规定任务之外，本文还进行了迁移学习、进行特征选取再实训及实测，以及对在使用 3.2 节提出的模型预测时预测的原理进行了探索。本文对每一次实验得到的结果都进行了分析和讨论。

1.4 本文组织结构

本文具体安排如下：

第 1 章，绪论。主要介绍了高频交易中股价预测的研究背景和意义，同时针对其国内外发展现状进行了简单的总结。最后说明了本文涉及的主要研究工作和文章组织结构。

第 2 章，相关理论和关键技术简介。本章主要介绍了作者在第三章建立模型结构时所采用的两种深度神经网络——卷积神经网络和循环神经网络的基本概念，以及第四章探索神经网络模型预测原理时将用到的理论方法。

第 3 章，预测模型的建立及对初步建立的模型的调优。这一章为第四章的最终预测任务进行了前期准备。作者首先在 3.1 节中分析了究竟使用何种预测方法实现本文的研究任务最为妥当，随后在 3.2 节中根据选择的方法，建立了四种模型。在 3.3 节引入了模型调优及后续实测均会使用到的预测效果评估指标后，在 3.4 节对模型参数进行了大量的调试工作。

第 4 章，实测结果及额外工作。在第 3 章的基础下，本文使用调优后的模型在第 4.1 节中阐述了实训及实测过程，并展示了实测结果。随后，本文还在 4.2～4.4 节中做了大量的额外工作，包括迁移学习、特征选取及重测，还有对深度神经网络的预测原理用卷积神经网络为例加以解释。

1.5 本章小结

本章对研究背景及其意义、国内外相关研究、本文将进行的工作及文章结构进行了阐述。

第 2 章 关键技术介绍

2.1 卷积神经网络及其反向传播

卷积神经网络，是一种结构大体上如图 2-1 所示的神经网络。较经典的卷积神经网络通常包含卷积层、池化层和全连接层。

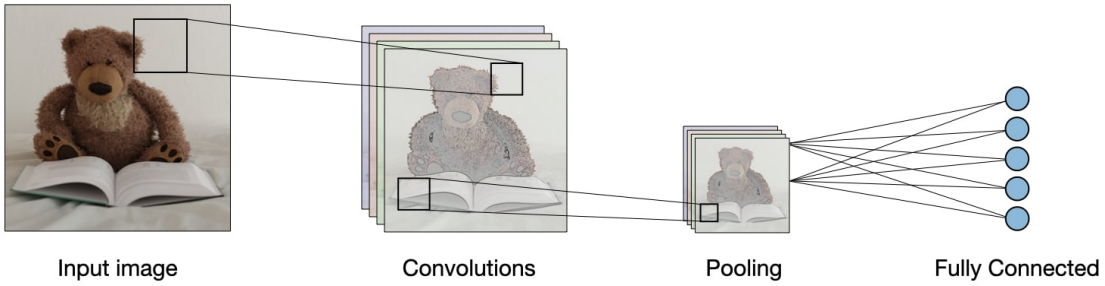


图 2-1 一种典型的 CNN 的大致结构图[1]

图 2-1 中的卷积层（Convolution layers）使用滤波器（filter）来对所输入数据的各个通道进行扫描，同时进行卷积操作。卷积操作后输出的结果称为特征图（feature map）。其中，滤波器是一个蕴含权重数字的矩阵，它的大小（如图 2-1 中图片内的方框所示）以及扫描步长（stride）是使用者常常调节的，滤波器大小及扫描步长也被称为超参数。卷积层有如下两个优点：参数共享，及稀疏连接。这些特性使得滤波器上需要被学习的参数比全连接层少得多，同时也为卷积神经网络带来了擅长捕捉平移不变性的特点。

池化层（Pooling layers）通常紧随在卷积层后出现，它的作用是对其获得的输入进行下采样，从而减少输入的大小，加快计算速度。同时这也会强化输入中一定区域内比较突出的特征，使侦测更加精准。常见的池化方式有两种，分别为最大池化（max pooling）和平均池化 （average pooling）。其中最大池化产生的效果是在某个特征在某个滤波器中的某个地方被侦测到做一标记；当然如果某个特征在某个区域中并没有被侦测到，则不标记。

全连接层（Fully Connected layers）一般在网络的末尾出现，数据在传入它之前需要首先被拉平（flatten），随后这些拉平后的数据将被映射到全连接层所有的神经元之上。

[1] 本章使用的图片均来源于 Afshine Amidi 和 Shervine Amidi 所著的 VIP Cheatsheets for CS 230.

训练一个这样的网络即是指使模型在如图 2-2 和图 2-3 所示的反向传播中依靠输入数据自动学习滤波器矩阵中的权重，从而得到一个可以实现自动侦测需要其侦测的特征的滤波器。

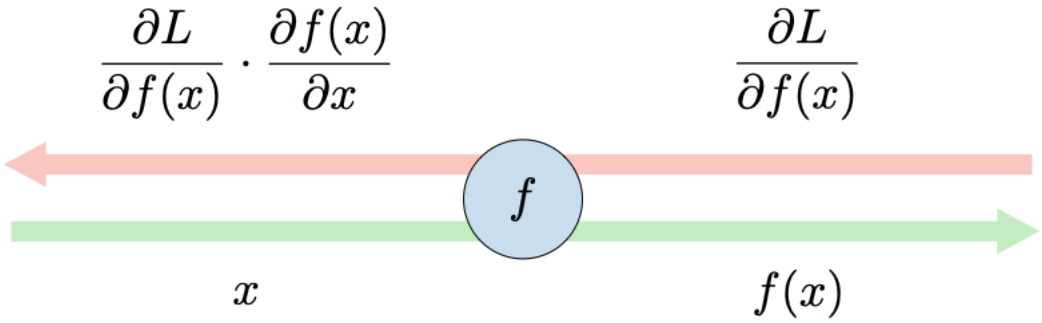


图 2-2 反向传播

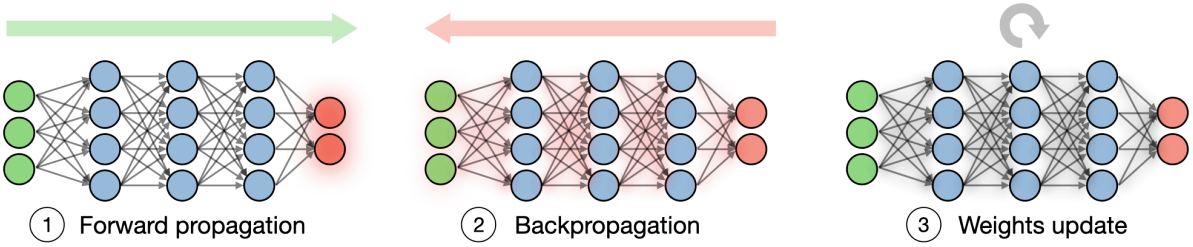


图 2-3 训练过程

学习方法如公式 2-1 所示：

$$W \leftarrow W - \alpha \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial W} \qquad (式\ 2-1)$$

式 2-1 中：

W: 为权重矩阵；

α: 为学习速率（learning rate）；

$\mathcal{L}$: 为损失函数。

2.2 循环神经网络及其反向传播

循环神经网络等序列模型近年来已经被广泛地应用到了许多领域中，如语言识别和自然语言处理问题等。所有典型的循环神经网络结构上均有一个优点，即可以处理任意长的输入且模型大小不会随输入大小的增加而增加。此外，循环神经网络可以像 2.1 节中提到的卷积神经网络一样，在输入的不同位置上共享已经学习到的特征。结合本文的研究目标来讲，即如果中价上升（下降、不变）出现在了输入的某一个位置上，且网络已经认识到了这样的输入模式属于中价上升（下降、不变），那么今后它将可

以自动地在输入的其他位置上自动地发现这样的特征，识别到其同样属于中价上升（下降、不变）。

图 2-4 是一个传统的的单向多对多循环神经网络结构：

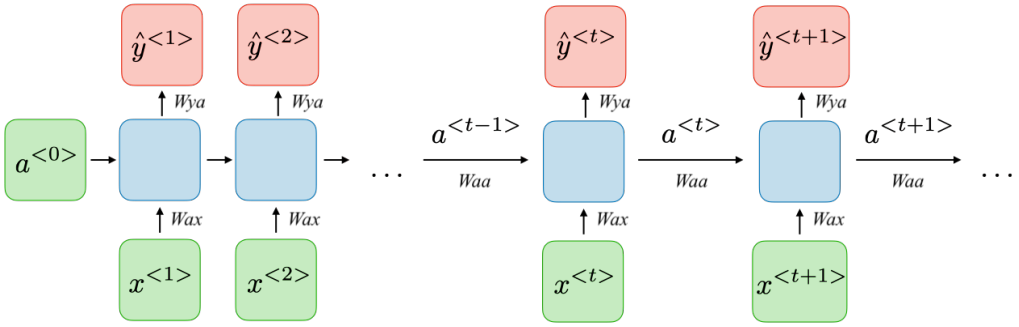


图 2-4 一种传统的单向 RNN 的正向传播图

对于每一时间步 t，激活函数输出值 $a^{<t>}$和当前时间步的预估值 $\hat{y}^{<t>}$表示如下：

$$a^{<t>} = g_1(W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a) \qquad (式\ 2-2)$$

$$\hat{y}^{<t>} = g_2(W_{ya} a^{<t>} + b_y) \qquad (式\ 2-3)$$

式 2-2 及 2-3 中：

W_{ax}: 表示从每一个 $x^{<t>}$连接到隐藏层的一组参数；

W_{aa}: 表示从每一个 $a^{<t-1>}$连接到下一个隐藏层的一组参数；

W_{ya}: 表示从每个隐藏层连接到 $y^{<t>}$的一组参数；

b_a,b_y: 表示偏差值；

g_1, g_2: 表示相同或不同的激活函数。

从图 2-4 和公式 2-2、公式 2-3 中可以注意到，当循环神经网络继续读取数据中的第二部分 $x^{<2>}$时，除去只用 $x^{<2>}$作为了输出 $\hat{y}_2$ 的输入外，还将第一时间步计算的结果也作为了输入。也就是说，循环神经网络实现了在给定第一时间步计算结果的情况下，计算第二时间步的预测结果这项功能。同样地，以此类推，可以看到在估计 $\hat{y}_3$ 时，网络也不仅使用了 $x^{<3>}$的信息，还使用了 $x^{<1>}$和 $x^{<2>}$的信息来辅助计算。这意味着，该种类型的网络在计算时能够考虑到过往的历史信息。

但在实际应用的过程中，类似图 2-4 所示的传统循环神经网络存在着一定的缺陷。因为它的输出 $\hat{y}$ 的局部依赖性过于强。仔细观察可以看到，图 2-4 中$\hat{y}$的值主要依赖靠近其本身的其它几个值，从而较难被输入序列中较早（即图中靠左）的部分影响到。为了规避这个问题，作者对于图 2-4 中的每一个 RNN 单元都采用了带有特殊门控的 LSTM 循环单元，该单元的结构如图 2-5 所示。

图 2-5 中：

$c^{<t>}$: 为记忆单元，且 $c^{<t>}= \Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$；

$\tilde{c}^{<t>}$: 为可能的 $c^{<t>}$的替代值，且$\tilde{c}^{<t>}= tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$；

$a^{<t>}$: 为激活值，且 $a^{<t>}= \Gamma_o \star c^{<t>}$；

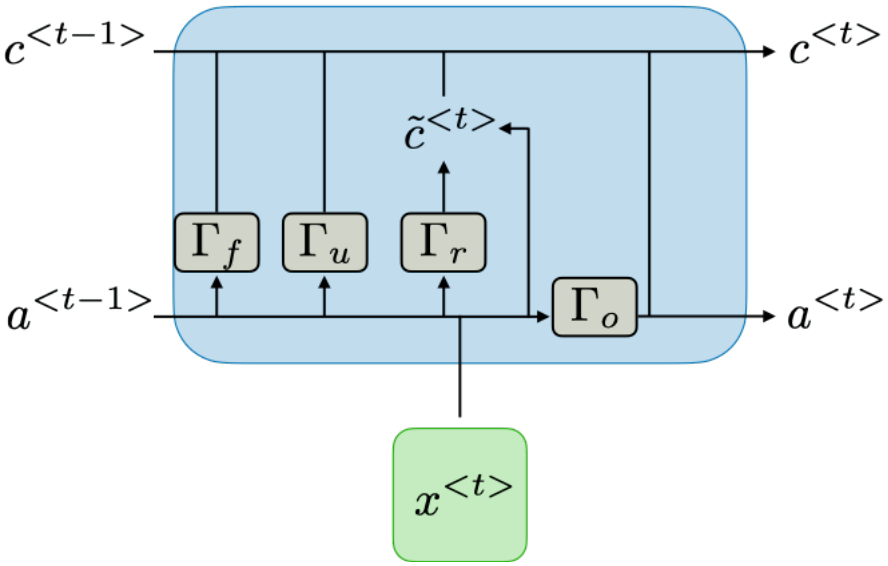


图 2-5 LSTM 单元结构图

Γ_u: 该门负责判断是否要用$\tilde{c}^{<t>}$替代 $c^{<t>}$；

Γ_r: 该门定义了 $a^{<t-1>}$与计算$\tilde{c}^{<t>}$的相关性；

Γ_f: 该门负责控制是否保留 $c^{<t-1>}$且将其加到$\tilde{c}^{<t>}$上（类似于指数加权移动平均）；

Γ_o: 该门负责控制要输出记忆单元的多少；

$\star$: 表示向量间的逐元素乘法。

上述所有具有特殊用途的门均有着形式上相同的计算式：

$$\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b) \qquad (式\ 2-4)$$

式 2-4 中：

σ: 为 sigmoid 函数；

W, U, b: 均为参数，且因门不同而不同。

此外，依照本文研究问题的需要，应使循环神经网络处理完整个输入序列后再输出 $\hat{y}$。因此，实际实验中将采用与图 2-4 类似，但形式上为多（输入）对一（输出）的如图 2-6 所示的网络结构。

图 2-6 中 T_x表示输入序列的长度，且此处 $Tx > 1$。

循环神经网络的反向传播在每一个时间点都将被执行。对于某一时间步 T，损失函数$\mathcal{L}$相对于权重矩阵 W 的导函数在式 2-5 中被定义：

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W}\Big|_{(t)} \qquad (式\ 2-5)$$

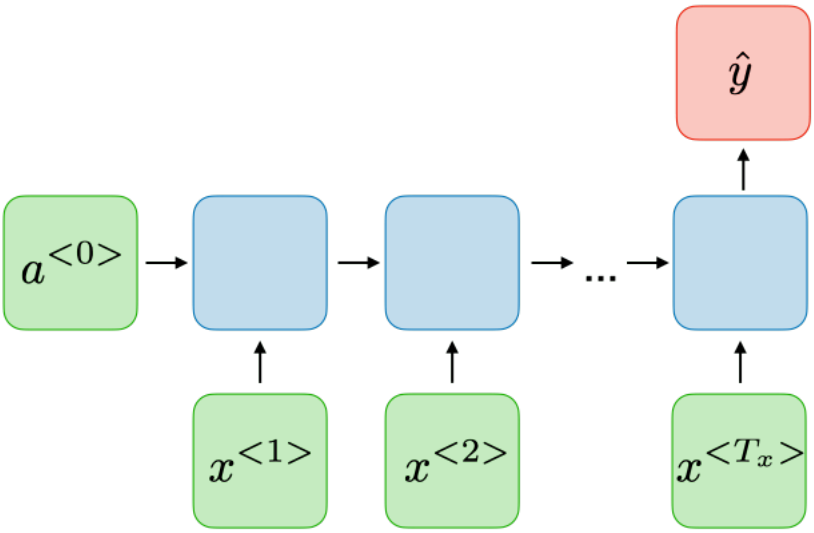


图 2-6 多对一的 RNN 结构

2.3 显著图

目前许多网络的改进都是基于反复试错的。大多数时候，人们会尝试各种超参数，然后用验证集来调试模型，判断模型的效果是否有提升。这样的做法实际上存在一个问题，即结果的可解释性较差。也就是说，人们不容易知道为什么以某种方式调整模型，其效果就会随之变化。显著图恰是一种方便向他人快速解释模型是否正在正确地分类的一种方法。为便于解释原理，此处举一个例子：现有许多张仅含单只小动物的图片，某个模型欲对所输入的图片中包含的小动物的种类进行分类。要利用显著图考察此模型是否正在正确地进行分类，需要做的事情就是将式 2-6

$$\frac{\partial\ score\ of\ some\ animal(x)}{\partial\ x} \qquad (式\ 2-6)$$

中所定义的“模型给出的某个分类的得分（一般是在被 SoftMax 层过滤前的得数）的导数”逆向传播（back propagation）至输入，察看究竟是输入中的哪一部分使得模型认为这张图片里含有的是某种特定动物，从而模型才会给这种动物打分最高。式 2-6 用自然语言描述出来即：输入中的哪一个像素 x 仅有微小的变动即会对分类得分产生最大的影响。特别地，对于图片分类任务，将会得到一个矩阵——Pixel Score Map。这个矩阵中绝对值大的数字就代表着它们各自对应的像素点是对分类得分有较大影响的像

素点。值得一提的是，该方法也常被应用于从图片中分割出仅有这只动物的部分的任务上，可以通过这个方法来探索某个分类结果究竟依赖于输入中的哪些部分。当然，用来解释神经网络输出的方法还有遮挡敏感度（Occlusion Sensitivity）和类别活化映射（Class Activation Maps），它们同显著图一样，都可以用来帮助理解一个网络做决策的过程是什么样的。只不过显著图与类别活化映射的区别在于后者并没有向后传播，而是仅仅做了上采样。

2.4 梯度上升法

梯度上升法（Gradient Ascent）又称 Class Model Visualization。它是一种可以探索网络中间的某个激活函数正在“思考”什么的方法。其原理是保持训练得出的权重不变，然后尝试对一个输入 x 最小化式 2-7 中定义的损失函数 L：

$$L = Score_{some\ category}(x) - \lambda||x||_2^2 \qquad (式\ 2-7)$$

即网络给出的某个分类的得分减去一个正则化项。这个损失函数将用于最大化某个指定滤波器的激活值，即将当前输入的图像像素点向着梯度的方向去“增强”。以该函数为优化目标优化后，将可以看到使这个滤波器激活的究竟是什么。其中正则化的作用是使一个像素周围的其他像素点均与该像素值相差较小。对于图像输入而言，正则化项的效果是使 x 在可视化之后能够看上去更加平滑自然一些。当然，使用比 L2 正则化更好的正则化方法可以得到更好的可视化效果，如高斯模糊（Gaussian Blurring）。

该方法的使用步骤为，首先将输入 x 向前传播（forward propagation），接着计算式 2-7 所定义的 L，然后再向着输入 x 的方向一直计算 L 相对于 x 的导数直至 x 本身，以找到为分类得分贡献最多的那个输入。最终使用式 2-8 来更新这个输入，并反复迭代。

$$x = x + \alpha\frac{\partial L}{\partial x} \qquad (式\ 2-8)$$

2.5 本章小结

本章对实验中将使用到的关键技术进行了简明扼要的介绍，包括 3.1 节中将使用到的神经网络的基本结构，以及将在 4.4 节中用到的两种可以帮助理解神经网络预测结果的方法。

第 3 章 模型建立及参数调试

本章在对毕业设计任务进行了分析理解的基础上，在 3.1 节中对预测方法进行了选择，3.2 节中即使用所选的方法进行了模型结构建立，随后在 3.3 节中引入后文模型参数调整及实际预测时可以对调整及预测效果进行评价的指标，最后在 3.4 节中介绍对 3.1 节中建立的模型结构进行参数调整的过程。

3.1 预测方法的选择

鉴于高频股票市场数据体量极大，且股票价格的可变性和不确定性较强，一些传统方法如数据挖掘、统计方法和非深度神经网络模型并不适合用来处理本文实验中将要用到的 FI-2010 数据集中的股票价格。传统机器学习算法如逻辑回归、支持向量机（Support vector machines, SVM）和决策树等等在巨量数据面前均会遭遇性能瓶颈，只能大致适应数据的趋势，很难学习到数据内部的细节，精度不会太高。因此，作者选择在实验中使用深度神经网络来解决问题。

在深度神经网络中，循环神经网络和它的几种变体很适合处理时间数据。而卷积神经网络虽然在大多数时候被应用于解决图像分类问题，但 2016 年 Google 旗下的 Deepmind 团队提出的 WaveNet 网络在序列分析的问题上产生了非常好的效果，故作者认为值得一试。综上所述，该两种神经网络与研究目的相匹配，适合用来着手开展研究。

在后文实际实验时，为了避免在手动使用 Python 和 Numpy 在构建反向传播时会产生各种问题，本文在整个实验过程中采用了深度学习编程框架 Keras 来辅助搭建网络模型。Keras 是一个使用 Python 编写的高级神经网络 API（编程框架），能够在包括 TensorFlow 和 CNTK 在内的多个低级框架之上运行，使得使用者可以在几小时之内就可以构建出来一个深度学习算法，便于作者迅速地试验调试不同的模型，节省时间。虽然 Keras 比低级框架限制性更大，如一些非常复杂的模型不能够在 Keras 中实现，但对于本文的研究问题而言，Keras 已经足够用了。

因为本文计划使用循环神经网络和卷积神经网络，而使用它们需要使用者独立指定模型需要的激活函数以及优化算法。作者在考察了数个激活函数，如 Sigmoid 激活函数，tanh 激活函数和整流线性单位函数（Rectified Linear Unit, ReLU）激活函数后，最终决定将本文全部涉及到的模型中所有的激活函数取为 ReLU 激活函数。这是因为虽然

其它几个函数也可以起到使高度非线性的神经网络线性化的作用，但相比之下，ReLU函数尾部的导数值更大，有助于缩短训练时间。

而在优化算法的选择上，因为 Adam（Adaptive moment estimation）算法结合了Momentum 优化算法和 RMSprop 优化算法的优点，改为了使用梯度的指数加权平均来更新权重，即在式 2-1 中将最后的导数项替换为了权重导数的指数加权平均除以导数梯度的方均根（如式 3-1 及式 3-2 所示）：

$$w - \alpha \frac{v_{dw}}{\sqrt{s_{dw}} + \epsilon} \qquad (式\ 3-1)$$

$$b \longleftarrow b - \alpha \frac{v_{db}}{\sqrt{s_{db}} + \epsilon} \qquad (式\ 3-2)$$

能够给本文的模型训练带来优势，故作者选择它作为所有模型的优化算法。其优势具体有两点：一是它可使得模型在学习时在遇到损失函数鞍点时，比使用标准的梯度下降优化算法更快地离开鞍点，二是它赋予了模型自适应学习速率的功能，方便作者在实验初期使用更大的学习速率。

3.2 预测模型结构的建立

作者利用在 Stanford CS230 为期 9 个月的深度学习课程中学到的知识，建立了本节所述的四种模型的结构，分别为长短期记忆循环神经网络模型、卷积神经网络模型、长短期记忆循环神经网络及卷积神经网络融合模型和空洞卷积模型，以期在预测任务中使用。

模型的结构均直接遵从于 Andrew Ng, Kian Katanforoosh 讲师以及无数科学家的经验构建而成，其中构成模型的各个隐藏层可能在 3.3 节中的模型参数调整实验中依据实验效果有所删减。

本节使用的模型结构图均是使用 Keras 提供的神经网络可视化函数 plot_model 绘制得到的。plot_model 函数可以借用 grapviz 画出利用 Keras 构建的网络模型结构图，并导出为图片。使用该函数绘制出来的图片中的文字只能为英语，本文会加以解释。

首先是长短期记忆循环神经网络模型结构，具体模型结构见图 3-1。Andrew Ng 讲师在课程中提到长短期记忆循环神经网络模型结构过大会导致计算时间过长，难以训练，故本文所提出的短期记忆循环神经网络模型结构较小，仅有 1 层 LSTM 层，为避免后文实验中可预料到的过拟合现象，为该模型添加了 Dropout 层：

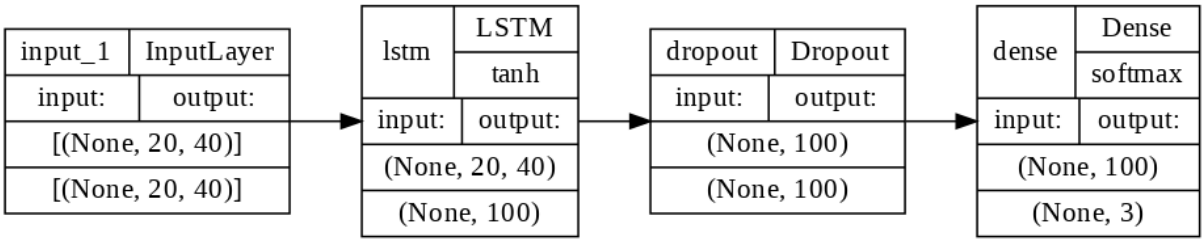
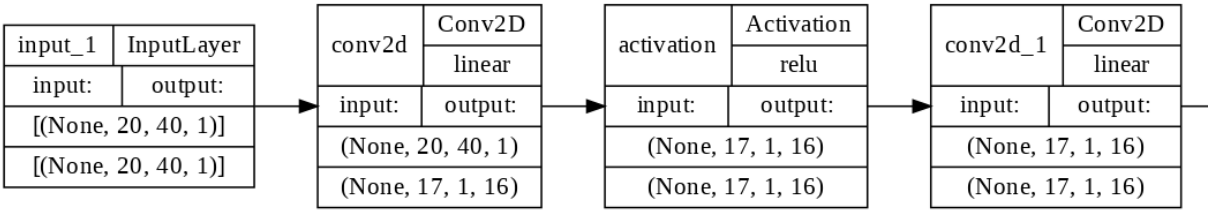


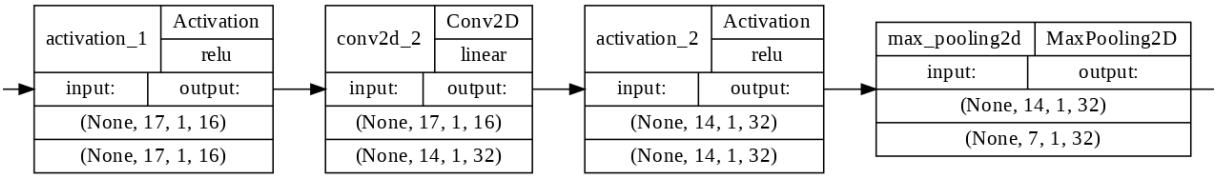
图 3-1 长短期记忆循环神经网络模型结构

图 3-1 中的 InputLayer 即为输入层，lstm 为 LSTM 层，dropout 为 Dropout 层，dense 为全连接层。每层下方还显示了该层的大小。可以注意到在全连接层后紧密连接了一个 SoftMax 层，SoftMax 层可以讲该层输入的值转化输出为一个 0～1 之间的概率，便于进行分类。

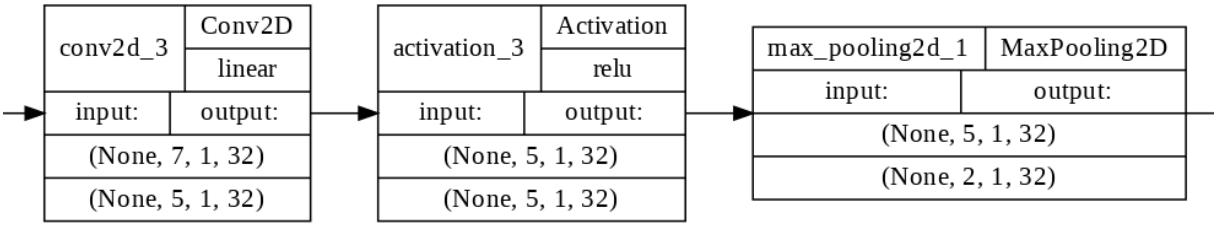
同样地，按照 Andrew Ng 讲师的建议，在建立卷积神经网络模型结构时，随着卷积神经网络的加深，应使输入的宽、高减少，通道数增加。本文建立的卷积神经网络模型结构如图 3-2 所示，因网络过大故分四段显示。根据 2.1 节中的知识可以得知该网络遵从了 Andrew Ng 讲师的建议。



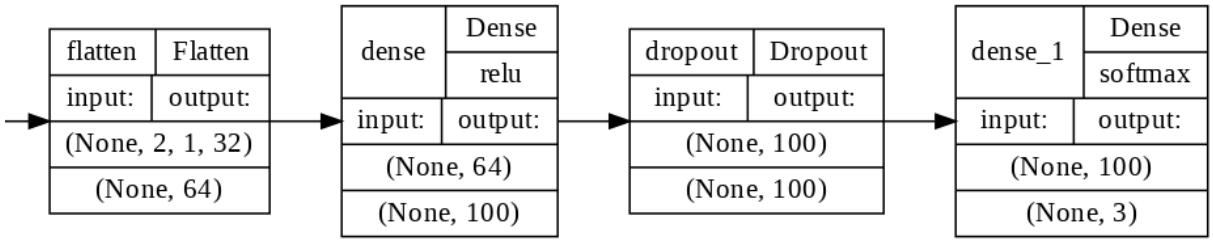
(a) 第一部分卷积神经网络模型结构



(b) 第二部分卷积神经网络模型结构



(c) 第三部分卷积神经网络模型结构



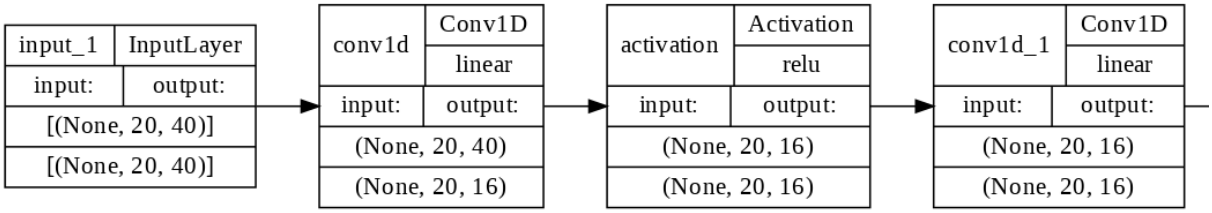
(d) 第四部分卷积神经网络模型结构

图 3-2 卷积神经网络结构图

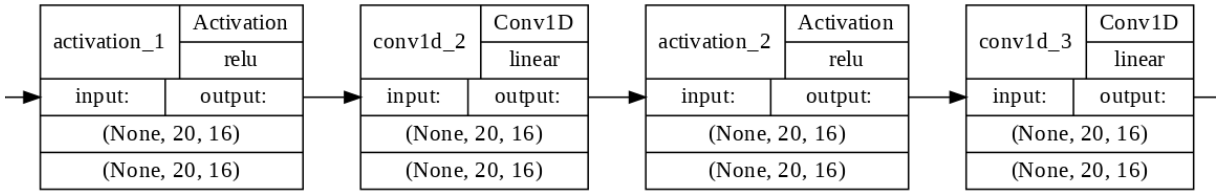
其中，Conv2D 为卷积层，Activation 为激活函数，MaxPooling2D 为最大池化层，Flatten 为“拉平”层，其余标识与对图 3-1 的解释一致。

长短期记忆循环神经网络及卷积神经网络融合模型结构即是将图 3-2（d）中所示的卷积神经网络模型结构中第四部分的全连接层（标识为 Flatten）替换为了一个含 100 个 LTSM 单元的层，其余不变，故恕不再附图。

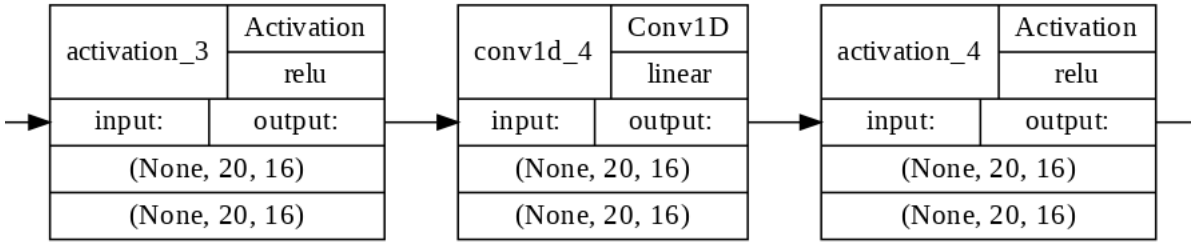
空洞卷积模型结构是借鉴 Wavenet 模型的结构建立的，如图 3-3 所示：



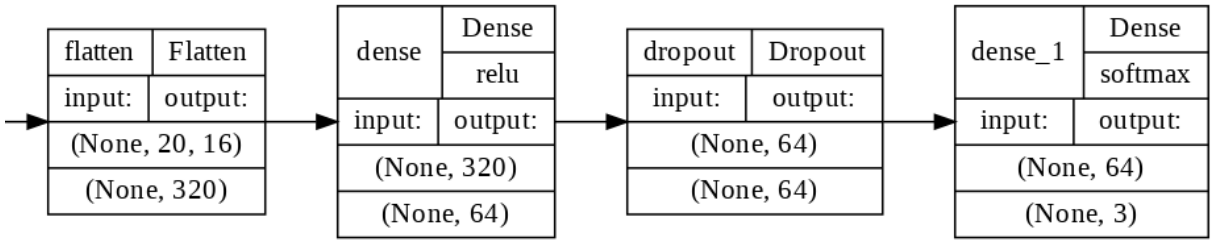
(a) 第一部分空洞卷积神经网络模型结构



(b) 第二部分空洞卷积神经网络模型结构



(c) 第三部分空洞卷积神经网络模型结构



(d) 第四部分空洞卷积神经网络模型结构

图 3-3 空洞卷积神经网络结构图

其中，Conv1D 为卷积层，其余标识与对图 3-1 及图 3-2 的解释一致。在这个结构中，作者在第二、第三、第四和第五个卷积层上使用了空洞卷积算法，空洞卷积模型也因此而得名。

以上四种模型具体实现时的程序代码详见附录 1。

3.3 预测评估指标

模型包含结构及参数，在 3.2 节中已经得到模型结构的情况下，下一步的工作应是对模型参数进行调优，但为了确定何种参数的取值才是最优的，需要先引入评估指标。

数据源论文中作者用来评估自己提出的两种基线模型时所使用的指标有四种，分别为准确率（Accuracy）、查准率（Precision）、查全率（Recall）和 F1 分数（F-score）。而本文实验时对预测结果设置的衡量指标为如下四种：损失值（Loss）、准确率、F1 分数和 Kappa 系数（Cohen's kappa coefficient），原因如下：

数据集作者采用的查准率指标的定义为“分类器认为是对的事情有多大概率真的是对的”，即若一个分类器有着 95%的查准率，这意味着分类器说某件事情是这样的，有 95%的几率这件事情就是这样的。其计算公式如式 3-3 所示。而查全率描述的是拿所有确实是对的的事情给分类器，它有多大几率能够正确地分类出来。如果一个分类器有着 90%的查全率，这意味着对于给定数据集中所有真的是对的的事情，这个分类器能够查出 90%。其计算公式如式 3-4 所示。

下表 3-1 使用二者的混合矩阵清晰地表述了它们的联系与区别：

表 3-1 查准率与查全率的联系与区别

	被分为类别相关	被分为类别无关
实际与类别相关	TP	FN
实际与类别无关	FP	TN

$$Precision = \frac{TP}{TP + FP} \qquad (式\ 3-3)$$

$$Recall = \frac{TP}{TP + FN} \quad (式\ 3-4)$$

而通常，人们往往既在乎查准率又在乎查全率，但时常出现其中一个很低而另外一个又很高的情况，很难在两者之间取舍。所以在实际应用中，通常引入另一个评估指标——F1 分数。它由查准率和查全率共同决定，但又不完全取决于其中的任何一个。实际上，F1 分数综合考虑了二者因素，对它们取了调和平均数，如此便定义了一个单一的量化指标，方便本文直接依据它进行比较。当且仅当查准率和查全率都很高时，F1 分数才会高。这样，本文就不再需要查准率和查全率作为两个单独的衡量指标予以体现了。

损失值则是指 2.1 节中提到的损失函数 L 输出的值。本文需要损失函数的原因是，在得到模型的输出后，需要比较它给出的输出值与真实值相差多少。另外也需要使用它来考察在反向传播时究竟单次以多大幅度调整模型参数才可以使模型能够比以往更精准地识别中价变化趋势。因此这里定义一个这样的函数，以此来驱使模型得到的预测值和真实值之间的差距缩小。

首先来定义逐元素的损失。对于本文研究的问题，股票未来的中间价格变化方向分类是对一只股票在未来即将到来的某一段时间内的中间价格的变动趋势进行变化方向类别判断，具体判断该只股票的中间价格在未来的这一段时间内是会涨、会跌还是保持不变。显然，这是一个传统的三分类问题，所以本文的损失函数设定为式 3-5 描述的一种最大似然估计（Maximum Likelihood Estimation）——分类交叉熵（Categorical Cross-Entropy Loss）：

$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_{i} y_i^{<t>}\, log\, \hat{y}_i^{<t>} \quad (式\ 3-5)$$

其中：

i: 表示预测类别的数量。

显然，作者定义的损失函数可以诱导模型去学习如何最小化预测值与真实值之间的差距。

接下来，把每个时间步上求得的损失加和，就得到了整个序列的损失，见式 3-6：

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>}) \quad (式\ 3-6)$$

其中：

T_y: 表示输出序列的长度。

当然在程序执行时，会把它复写为一个凸函数，以方便在优化过程中使用。

而增加 Kappa 系数作为本文模型质量的评估的标准之一是因为作者对原始数据集所含的三种类别的数据数量进行了统计，发现数据集中所含的“中价将会上升”、“中价不变”以及“中价将会下降”三种类别的数据数目的分布是不均匀的，如图 3-4 所示，其中横轴代表类别标签，可以看到其仅在 1、2、3 有值；纵轴代表被标为该类数据的数量。作者进行本文所有实验时使用的集成开发环境无法正常显示中文（显示为方框），故以英文做图例。可以通过柱形的高低直观地看到，各类数量相较而言是非均衡的。

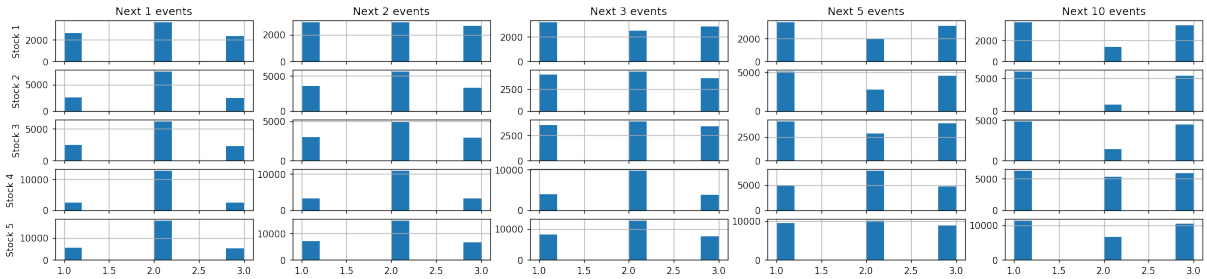


图 3-4 原始数据集所含三种类别的数据各类个数

Kappa 系数可以利用数据集的不平衡性归一化准确率。

现在已经得到了模型结构和调整参数及最终预测时所需要的评估指标，下一节（3.4 节）将介绍模型参数的调整过程。

3.4 模型参数调优过程及数据集介绍

参数的调整对于预测的精度至关重要，调整参数的目的即是使在 3.1 节中得到的模型可以在一切参数默认的基础上更好地进行分类。本文对模型的参数进行了大量的调优，主要调整的参数包括：学习速率、批次大小、隐藏层中隐藏单元个数、隐藏层数等。需要指出的是，调整过程并无固定章法，一般调参者均会通过请教在相关问题上有一定调参经验的同行来获得宝贵的调参经验。作者在调参过程中得到了作者指导老师的引导。

在介绍调优过程之前，需先引入调优过程及本文后续工作中所需要的数据集。在本节后续表格中所展现的所有模型的训练阶段，作者均使用了公开高频限价订单簿基准数据集 FI-2010[15]中的全五只股票前六个交易日的数据作为训练数据。而在模型验证阶段，所使用的数据为全五只股票第七和第八个交易日的数据。

该 FI-2010 数据集共包含了自 2010 年 6 月以来在赫尔辛基（Helsinki）交易所交易的某五只股票连续十个交易日（2010 年 6 月 1 日～2010 年 6 月 14 日）的限价订单簿数

据。它是通过对交易所的一个含有约 400 万条记录的原始信息薄（Message Book）每隔十条进行一次快照采样生成的，显然该订单簿中平均大约每隔 0.5 秒就有一张快照（有时该时间间隔会因市场活跃性的变化而有所不同），也即含有约 40 万条委托记录。

由 NASDAQ Nordic 运营的赫尔辛基证券交易所是一个纯电子限价单市场，此数据集作者选定的这五只股票仅在这一个交易所交易，这避免了模型在预测时可能会遇到的与分散市场相关的问题。此外，该数据集中不含开盘集合竞价的部分，而仅保留了连续竞价开始后半小时到收盘前二十五分钟之间的交易记录，使得此数据集不会因存在接近市场开盘和收盘时间的数据而存在偏差或其他异常。

该数据集中的每一条记录所包含的属性均是依照 Kercheval & Zhang 在文献[16]中提出的一种 144 维列向量的结构来存储的，如果直接使用全部的 144 个属性去训练模型，计算量会非常大，很难得出每个属性的权重。为了获取理想的实验效果，规避高维数据带来的麻烦，本文首先先使用前 40 维所包含的数据进行实验，而后在第四章将再利用随机森林作为特征选择算法，选出这 144 个属性中对预测最有帮助的属性，进行对比实验。此 144 维的列向量的前 40 个维度记录了市场上当前买卖双方前十档的委托价格和委托总量，向量全部 144 维的构成可见数据论文[15]中的表格 4，本文不再赘述。此外，这些记录都一一对应着 5 个面向不同预测时间范围的，存储着中间价格变化方向的类别向量。类别共有三种，其中类别向量中的 1 代表中价将会上升，2 代表中价相对不变，3 代表中价将会下降。

下图 3-5 是该数据集中第一只股票在第一天的第一级委托总量和委托价格的概况，其中横轴是第一支股票在第一天的交易次数，纵轴代表图例所示的四种变量各自在每一条交易记录中的取值。本章将基于这样的数据对模型进行调优并在后文使用调优得到的最优模型进行实测。

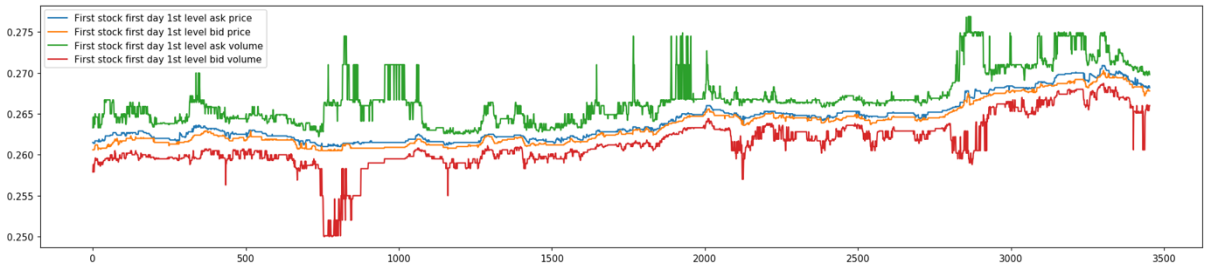


图 3-5 第一只股票在第一天的首档委托总量和价格

必须指出的是，数据集的作者提供了基于三种不同标准化方法处理后的数据集，分别为 Z-score 标准化、Min-Max 标准化和小数定标标准化后的数据集。Z-score 是一种

常见的基于原始数据均值和标准差进行标准化的方法，如 SPSS 中默认的标准化方法即为 Z-score；而小数定标标准化方法则是通过按一定规则移动数据的小数点的位置来实现对数据进行标准化的方法。其中小数点移动位数的多少取决于全部原始数据值中最大绝对值。本文所述的实验选用了经过 Z-score 标准化方法处理后的数据集。

另外，需要预先说明的是，在下述的所有实验中，每一个模型的输入均是多个滑动窗口。每一个窗口中都包含了数个原始数据集中的快照。每后一个窗口将在前一个窗口的基础上向后移动一个快照。作者在编写程序时利用程序逻辑保证了各个窗口中的数据不会存在跨越多支股票和多个交易日的情况。模型预测的是每个滑动窗口中最后一张快照之后到来的指定预测周期内的中价变化趋势。

本文仅对除空洞卷积模型之外的三种不同类型的模型进行了调试，在 3.1 节中得到的空洞卷积模型将不参与调试直接用于实测。调试记录见表 3-2、表 3-3 和表 3-4，每张表格对应了一种类型模型的调试记录。针对每一次调试的发现在各张表后予以了说明。

首先是针对长短期记忆循环神经网络模型进行的调试：

表 3-2 长短期记忆循环神经网络模型的调试记录

实验编号	预测周期	Loss	查准率	F1 分数	Kappa 系数
1	10	0.71/0.92	0.68/0.63	0.68/0.63	0.49/0.43
2	10	0.73/1.01	0.66/0.66	0.66/0.57	0.46/0.35

注 1：表格中后四列指标的记录格式均为“训练集上的得分/验证集上的得分”；

注 2：所有实验均是在单张 NVIDIA GeForce RTX 2080 Ti 显卡上训练完成的。

长短期记忆循环神经网络模型的第一次调参实验是作者在整个实验过程中尝试的第一种神经网络模型，同时它也将作为该种神经网络模型的基准模型。在这个模型的训练过程中，作者将数据输入了到模型结构图 3-1 中绘出的含有 100 个 LSTM 单元的层中。为避免预期一定会出现的过拟合问题，作者使用了 L2 正则化手段和 50%的 Dropout 予以解决，其中 L2 正则化所需要的正则化参数 λ，本文采用了 Keras 提供的默认值 0.01。本次实验模型的学习速率设定为 0.001，共遍历了数据集 100 遍（epoch）。单个数据批次含 500 个滑动窗口，其中每一个滑动窗口含有 20 个原始数据集中的快照。从表 3-2 中的结果来看其实不难注意到，这个模型已经有希望在实测时达到并超过任务书指定的目标了，但又因长短期记忆循环神经网络模型的实验结果过于好，仅用其达到任务

标准后就停止工作，难以达成规定的工作量，故本文进行了除使用此模型进行实测外的所有其他额外工作。

在第二次实验中，作者尝试降低了学习速率，以期在损失函数收敛时获得更好的结果。但出乎意料的是，结果显示第二次降低学习速率后得到的模型在验证集上的预测效果变差了。鉴于第一次实验所得到的模型已有极大希望达到任务书给定的研究要求，故停止对此种模型进行更多的调试。

接下来是对卷积神经网络模型的调试，见表 3-3：

表 3-3 卷积神经网络模型的调试记录

实验编号	预测周期	Loss	查准率	F1 分数	Kappa 系数
1	5	0.92/1.31	0.48/0.41	0.47/0.42	0.21/0.11
2	10	0.52/0.77	0.75/0.70	0.75/0.70	0.60/0.54
3	10	0.79/0.95	0.63/0.59	0.61/0.57	0.40/0.36
4	10	0.78/0.89	0.63/0.60	0.63/0.59	0.42/0.38
5	10	0.88/0.88	0.64/0.62	0.64/0.61	0.43/0.42

注 1：表格中后四列指标的记录格式均为“训练集上的得分/验证集上的得分”；

注 2：所有实验均是在单张 NVIDIA GeForce RTX 2080 Ti 显卡上训练完成的。

在卷积神经网络模型的第一次调参实验中，作者先随机采用了 0.001 的学习速率，预测每个滑动窗口中最后一张快照之后到来的五张快照的中价变化趋势。观察验证集上的损失函数值发现其在模型拟合后期开始上升。本次实验使用的模型中没有使用正则化。

在第二次实验中，改为了预测每个滑动窗口中最后一张快照之后到来的十张快照的中价变化趋势。

在第三次实验中，取消了第二个卷积层后的最大池化层，并且增加了第一个全连接层中神经元的数目，又在该层后采用了 60%的 Dropout。实验发现，得到的各个指标总体上来说更差了，但训练集和验证集的指标差距缩小了。

在第四次实验中，作者将遍历整个训练和测试集的次数增加至了 100 次。实验发现，在模型第 60～70 次遍历数据集时，验证集的损失函数值开始上升，这说明模型从这个训练阶段开始出现过拟合了。作者考虑在下一次实验中为模型使用 L2 正则化以尝试解决过拟合的现象。

在最后一次实验中，作者为第一个全连接层加上了 L2 正则化，并把 Dropout 的概率和每批次的大小分别降低到了 50%和 50。实验惊喜地发现，过拟合的问题被解决了。并且，这是卷积神经网络模型中迄今为止最好的结果；但值得注意的是，它在四个预测效果评价指标上依旧逊于长短期记忆循环神经网络。作者认为，这证明了卷积神经网络不太适合于处理序列数据。

最后是对卷积神经网络与循环神经网络融合网络模型的调试记录，见表 3-4：

表 3-4 卷积神经网络与长短期记忆循环神经网络融合网络模型的调试记录

实验编号	预测周期	Loss	查准率	F1 分数	Kappa 系数
1	10	0.74/0.81	0.67/0.65	0.67/0.65	0.48/0.46
2	10	0.75/0.84	0.67/0.64	0.67/0.63	0.48/0.45
3	10	0.74/0.79	0.68/0.66	0.68/0.66	0.50/0.49
4	10	0.71/0.80	0.69/0.66	0.69/0.65	0.52/0.48

注 1：表格中后四列指标的记录格式均为“训练集上的得分/验证集上的得分”；

注 2：所有实验均是在单张 NVIDIA GeForce RTX 2080 Ti 显卡上训练完成的。

在卷积神经网络与长短期记忆循环神经网络融合网络模型的调参实验中，作者先以单批次 50 个滑动窗口试训练，从训练过程中看，效果不差，其在第 26 遍遍历训练集和验证集时分别在损失函数值上取得了 0.82 和 0.89 的好结果，此效果显然已能与表 3-3 中卷积神经网络模型的最后一次的调试实验结果相提并论了。但本次参数配置的缺点在于使其运行速度过慢，具体表现为遍历一遍数据集约耗时近 4 分钟。综上，作者在本次尝试中，在模型训练至第 26 遍遍历数据集时即手动停止，随后将每批次大小增加至 500 个滑动窗口。实验发现，调整后遍历一遍数据集的时间降低到仅需 30 秒。如此调整后，在第 26 遍遍历训练集和验证集时分别在损失函数值上取得的值分别为 0.96 以及 1.00，可以接受。从表 3-4 第一行中也可以看到，最终得到的四种指标值并不差。但同时也能注意到，其在训练集和验证集上的指标相差较大，需在接下来的实验中提高正则化的强度。

在第二次实验中作者即提高了正则化的强度，将 Dropout 的比率提升到了 60%。结果显示，这样做后预测效果仅有少许提升，因此作者认为没有必要继续调整 Dropout 了。

在第三次实验中，作者先尝试把学习速率从 0.001 提升到 0.1，发现模型无法学习，接着又将学习速率降低到 0.0005，结果从评价指标上发现预测效果有提升。

在本种模型最后一次实验中，作者将每批次的大小降至250个滑动窗口，从预测效果上看，并无提升。故从本次实验可以得知，对于神经网络模型中含有 LSTM 层的情况下，降低每批次大小对预测效果的提升帮助并不大。因而在调整此参数的过程中，应当在学习效率和要在多大程度上避免损失函数陷入局部最小值二者中平衡每批次的大小，选取一个最优值。

在全部三种模型的调试过程中，作者发现，对于卷积神经网络模型而言：

1、它偏好较低的学习速率（≤0.001）和较小的批次（batch）大小（≤50）；

2、多于四个卷积层或两个最大池化层均会使预测效果变差；

3、尺寸小一些的滤波器比大一些的效果更好；

4、最优的窗口长度约为每个窗口容纳 100 个限价订单簿快照。

要特别指出的是，正是第三点发现，使得作者有了建立并使用空洞卷积模型进行预测的想法。

而对于长短期记忆循环神经网络模型来说：

1、虽然稍大些的批次大小在可接受的范围内降低了预测效果，但是它使训练效率提升了；

2、尽管对含 LSTM 单元的中间层使用了 L2 正则化和 Dropout，过拟合的现象依然存在；若欲将过拟合控制在一个可接受的范围内，则只能使用单个 LSTM 层；

3、最优的 LSTM 层大小也许小于实验中配置的 100 个 LSTM 单元。

经过以上的模型调优过程，目前在长短期记忆循环神经网络模型、卷积神经网络模型、长短期记忆循环神经网络和卷积神经网络融合模型这三个模型上都得到了各个模型种类目前最优的模型。下一章中，本文将采用这三类模型中得到的最优模型进行实训及实测。

3.5 本章小结

本章在充分理解了毕业设计任务书的基础上，首先在 3.1 节中对预测方法进行了分析及选取，并在 3.2 节中依照选取的方法建立了模型。其次，在 3.3 节中对后文模型参数调整及预测结果评判时均需要用到的评估指标进行了引入。最后在 3.4 节中，作者对在 3.2 节中得到的神经网络模型进行了大量的参数调整，得到了各类型网络中参数最优的模型，以进行第四章将要介绍的实测。

第 4 章 实测及拓展分析

本章在第 3 章对模型结构进行构建、参数调整的基础上，选用在众多调试实验中表现最优的模型，进行实训和样本外实测。除此之外，本章还进行了额外的迁移学习、特征选择后再预测以及对神经网络的预测原理进行解释三大工作。

4.1 模型测试及对比分析

本节使用 3.4 节中得到的各模型种类中最优的模型进行实训及实测，实验时依旧采用与模型调优时完全相同的实验环境。

实验共分四组，每组实验对应一种模型。其中前三组使用了 3.4 节中得到的各模型种类中最优的模型进行实验，而最后一组使用了 3.2 节中构建的类 WaveNet 模型直接进行实验，实验安排见表 4-1。四组的实验数据完全相同。下面首先进行实验简述，随后依据 3.3 节中介绍的评估指标对预测结果进行展示，最后依据结果进行对比及分析。

本节的实验目标是，给定输入训练集，希望可以得到一个能够自动且准确地告诉使用者中价变化出现在数据中何处的模型，并且最终用它进行实测。在整个实验中，作者使用如表 4-1 所示的四种不同结构的模型，辅以 FI-2010 数据集中全五只股票前八个交易日的前 40 维数据进行训练，并以全五只股票第九和第十个交易日前 40 维的数据进行了样本外预测。

进行训练的目的是使得模型中的每一层都去寻找一个自身得到的输入到目标输出 $\hat{y}$ 的映射，只有模型结构合适，参数调整正确，这个映射才会比较正确。在实测时，本节使得模型预测每个输入窗口中最后一个快照之后来临的 10 个周期内的价格变化方向。

表 4-1 实训及实测安排

实验序号	模型种类	对应前文中的模型
1	长短期记忆循环神经网络模型	表 3-2 实验编号 1 调试后所得的模型
2	卷积神经网络模型	表 3-3 实验编号 5 调试后所得的模型
3	长短期记忆循环神经网络与卷积神经网络融合模型	表 3-4 实验编号 4 调试后所得的模型
4	空洞卷积模型	3.1 节中直接定义的空洞卷积模型

注：所有表中的模型均将在单张 NVIDIA GeForce RTX 2080 Ti 显卡上训练完成。

四种模型在进行样本外实测后的预测效果如下表 4-2 所示。表 4-2 借 3.3 节中所定义的评价指标，展示了经过 3.2 节模型结构设计、3.4 节模型参数调优后使用四种模型进行中价变化方向预测的预测效果：

表 4-2 实测结果

模型种类	Loss	准确率	F1 分数	Kappa 系数
长短期记忆循环神经网络模型	0.70/0.91	0.71/0.62	0.67/0.61	0.48/0.41
卷积神经网络模型	0.92/0.95	0.52/0.52	0.50/0.51	0.26/0.27
长短期记忆循环神经网络与卷积神经网络融合模型	0.74/0.83	0.68/0.63	0.68/0.64	0.49/0.45
空洞卷积模型	0.41/0.37	0.87/0.89	0.89/0.89	0.84/0.84

注 1：表格中后四列指标的记录格式均为“训练集上的得分/验证集上的得分”；

注 2：所有实验均是在单张 NVIDIA GeForce RTX 2080 Ti 显卡上训练完成的。

表 4-2 中所有模型均经过了对整个训练集 100 轮（epoch）的训练。显然，以上四种模型中任意一种模型的结果已经超过了数据源论文中给出的基线准确率（基线 F1 分数等于 41%，基线准确率等于 48%），超额达成了毕业设计的目标。

接下来对表中四种模型取得的结果进行简单的比较及分析。不难发现：

1、空洞卷积模型的效果远优于其他模型；

2、单独的卷积神经网络泛化效果不错；

3、向 CNN 模型中增加一层 LSTM 层提升了它的性能；

4、考虑到 CNN+LSTM 模型比单一的 LSTM 模型过拟合程度更小，因此作者估计在更长时间的训练之后，它可以给出比单一的 LSTM 模型更好的结果。

4.2 迁移学习

为了再增加工作量，本文进行了本节（4.2 节）的迁移学习、4.3 节中的特征选取后再预测以及 4.4 节中的解释神经网络模型原理的工作。

一般的迁移学习指仅训练分类层权重或仅训练最后一层及分类层权重，但本节所进行的迁移学习重新训练了模型全部层的权重。本节先用全五只股票前八天的数据训练了实测中实验序号 3 使用的长短期记忆循环神经网络模型，并用训练后的模型在第九、第十天的数据上分别对各只股票进行了样本外预测。而后，又以各只股票前八天

的数据分别训练了五个上述长短期记忆循环神经网络模型，并使用它们同样在第九、第十天的数据上分别对各只股票进行了样本外预测。作者对这两种情况得到的预测效果进行了对比，发现在五只股票上训练而得的模型在所有指标上与单只股票训练而得的模型相同，结果见表 4-3。

表 4-3 迁移学习结果

5→1		1→1	
F1 分数	Kappa 系数	F1 分数	Kappa 系数
0.66/0.63	0.47/0.41	0.78/0.63	0.66/0.41

注 1：表格中四列指标的记录格式均为“训练集上的得分/验证集上的得分”；

注 2：本次实验是在单张 NVIDIA GeForce RTX 2080 Ti 显卡上训练完成的。

表 4-3 中所示的实验结果其实并不在作者的预判之中。作者预计经过五只不同股票数据训练后的模型应当比只经过仅使用将要预测的股票的训练数据训练得到的模型预测精准度更高。

4.3 经过特征选取后的卷积神经网络训练及实测

本文 3.3 节中提到，作者将首先仅使用前 40 维所包含的数据进行实验，而后再将利用随机森林作为特征选择算法，选出原数据集中提供的 144 个属性中对预测最有帮助的属性进行对比实验。下面介绍特征选择采取的算法及实验结果。

特征选择也称特征子集选择，它是指从已有的特征中选择一个特征子集出来。特征选择可以去掉对于预测结果影响不是非常大的特征，从而可能使预测结果更加精准。特征选择的任务就是要对每个已有特征的重要性进行量化，最终方便根据实际需要，选择最重要的数个特征。下面简单介绍一种非常成熟的特征选择算法——随机森林：

随机森林是最流行的机器学习算法之一。它是以决策树为基础的一个集成学习算法。它因可以提供平稳良好的预测性能、过拟合程度低和可解释性强而广为人用。特别地，可解释性强是指人们很容易就能计算出每个变量对最终决策的贡献有多大。

使用随机森林进行特征选择属于嵌入法特征选择的范畴。所谓嵌入法是指将特征选择过程嵌入到模型训练的过程当中，二者同时进行。这意味着在模型训练结束后，不仅可以得到接受训练后的模型，还能同时得到训练过程中被自动选择出来的特征。嵌入法是相对于过滤法和包装法而言的，嵌入法的好处有：准确性高、泛化性更好，以及结果可解释强。当然本文并不需要随机森林进行训练后得出的模型，作者仅需借

助它完成特征选择，而后使用前文已经得到的，实测效果中相对最差的卷积神经网络模型进行再训练和预测。

随机森林由许多棵决策树组成，其中每棵树都是因有放回地对数据集观测值和特征进行随机抽样而建立的。没有被抽到的数据将用来对使用被抽到的数据建立出的决策树的性能进行评估。这样，并非每棵树都能“看”到所有的特征以及所有的观测值，所以这些决策树是非相关的，也正因此，这种算法不容易过拟合。随机森林中的每棵树描述了一系列基于单一或组合特征的分类问题。在树的每个节点上，也即每个分类问题上，决策树用随机选择到的特征对观测值进行分类。特征重要性的高低就取决于每个分类结果是否准确。

作者使用了 scikit-learn 包中的随机森林算法对原数据集中作者提供的 144 个属性进行了重要性排序。随机森林中决策树的棵树作者随机设置为了 100 棵，因本文所采用的数据集数据量大，故同时随机限制树的最大深度为 3。经过随机森林算法的自动选择，其给出的前十个对预测任务来说最重要的特征为：第 69 个、第 88 个、第 87 个、第 86 个、第 91 个、第 82 个、第 98 个、第 89 个、第 92 个和第 93 个。实现选择过程的具体代码请见附录 1 代码①-5。

在得到随机森林给予的前十个最重要的特征后，作者注意到，它们恰好都集中在原数据集中的“价格和交易量对时间的导数”这一特征子集中。这一特征子集亦共有 40 个向量，其中每一个元素是对整个数据集前 40 维特征分别对时间求导数而得到的。作者使用这一特征子集中的前 20 维替换了前 40 维中后 20 维的第五至第十档买卖价及交易量，得到了一个含有一至五档买卖价及交易量，和一至五档买卖价及交易量分别对时间的导数的 40 维融合特征集合。下面，作者使用这个融合特征集合中全五只股票前八天的数据对实测中使用过的卷积神经网络模型进行重新训练，而后使用全五只股票第九天、第十天的数据进行样本外实测，实验结果如表 4-4 所示：

表 4-4 特征选择后的卷积神经网络模型实训及实测结果

模型种类	Loss	准确率	F1 分数	Kappa 系数
卷积神经网络模型	0.88/0.88	0.60/0.61	0.64/0.61	0.43/0.42

将表 4-4 所示的实测数据与表 4-2 中对应模型的实测数据进行对比，可以发现，进行特征选择后再训练模型并样本外预测，模型在验证集上的损失函数值下降了 13%，准确率提高了 17%，F1 分数和 Kappa 系数分别上升了 26%和 25%。

同时，作者也发现使用本节提出的融合特征集合进行训练时，实验训练过程中模型损失函数下降得要比 3.4 节中对卷积神经网络模型进行最后一次调试时下降得更平滑，如图 4-1 及图 4-2 所示：

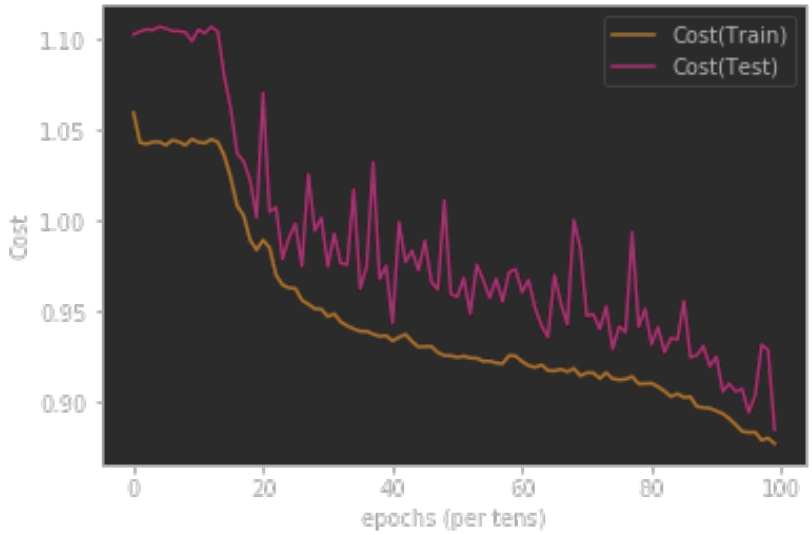


图 4-1 对卷积神经网络模型进行最后一次调试时损失函数关于遍历数据集次数的图像

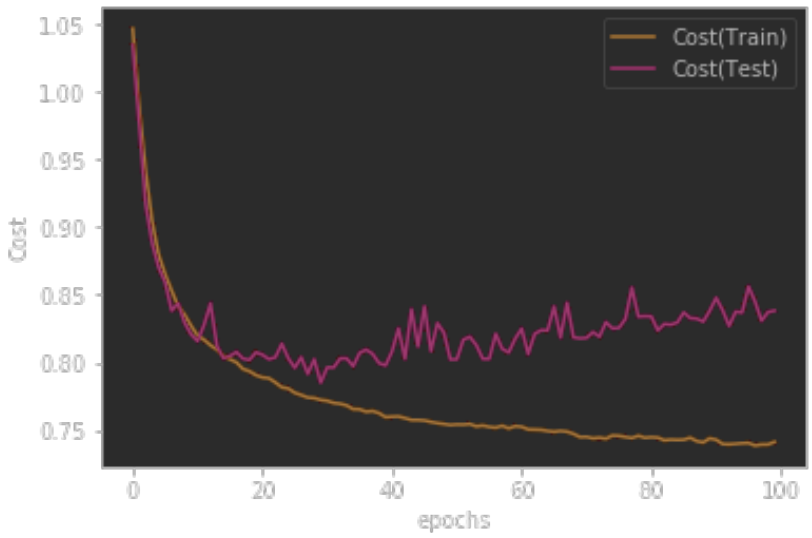


图 4-2 本节实验损失函数关于遍历数据集次数的图像

可以清晰地看到，损失函数在训练伊始的前 20 次遍历数据集时，使用融合特征集合训练，损失函数下降得更加平缓。

4.4 卷积神经网络预测原理探索

在许多人看来，使用深度神经网络模型进行预测这件事情的可解释性太差。虽然在 4.1 节中已经使用这种方法达到了实验目标，但其他领域的人难免会质疑该方法的靠

谱性。本节同样利用在 Stanford CS230 深度学习课程中学到的知识，以实测中使用过的卷积神经网络模型为例，对本文采用的预测方法加以解释。

作者首先遍历了整个训练集，试图用 2.3 节中提到的方法找出最后的分类层分别在“中间价将会上升”和“中间价将会下降”这两个类别中给出评分最高的那些数据对应于每一输入批次中的哪些数据，以了解本文得到的卷积神经网络模型自身是如何“看待”中价上升和下降的。一开始，作者先是依次察看了各个输入窗口，但因数据噪音太大，并未得出任何有用的结论。后来，当作者开始对整个批次中模型给出的预测值最高的输入序列取平均值后发现，本文训练出来的卷积神经网络模型分别在两个分类中给了如图 4-3 及图 4-4 所示的这些原始数据最高的分类评分，佐证了作者得到的卷积神经网络模型确实正在正确地进行分类。其中横轴代表输入窗口的长度，容易注意到实测中使用的卷积神经网络模型输入窗口恰为 100，而纵轴为各个变量的变量值。

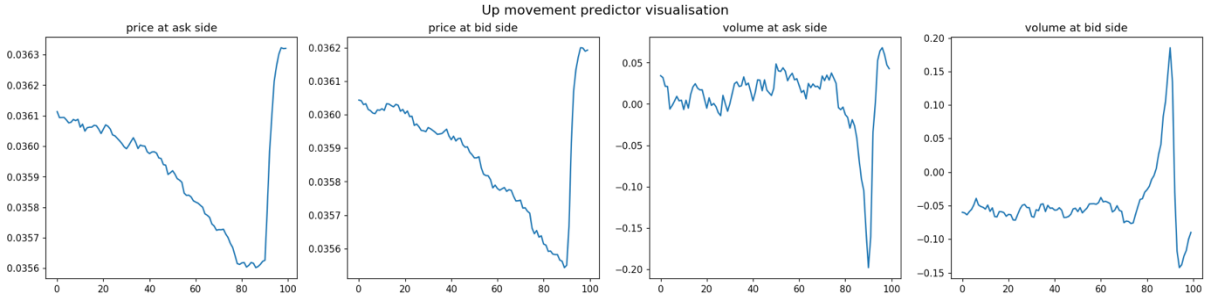


图 4-3 模型给“中价将会上升”分类得分最高的数据

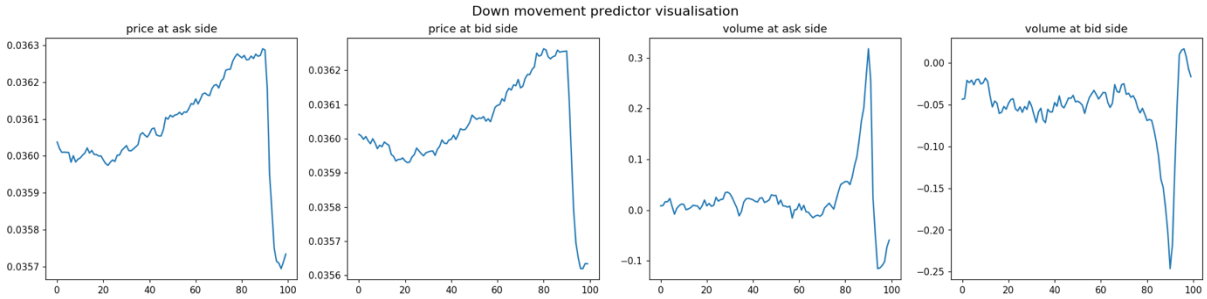


图 4-4 模型给“中价将会下降”分类得分最高的数据

接着，作者分别选择了卷积神经网络模型的某一靠后的激活层及最后的全连接层，希望了解这两层的滤波器学到了什么。这里的实验用到了 2.4 节所述的梯度上升法，作者将损失函数定义为了各层给出的评分。在考察模型中的最后一个激活层时，使用梯度上升法，得出了如图 4-5 所示的最能激活这一层的输入的前三档买卖价和如图 4-6 所示的前三档成交量。其中横轴依然代表模型输入窗口长度，且图 4-7 横轴意义同理。纵轴表示对应变量的变量值。图像表示，本文得到的卷积神经网络模型看起来恰恰是在

寻找巨量价格数据中有波动的那些数据，这证明了该模型已经学会了在输入含有价格的明显变化时积极地作出反应。

此外，作者还考察了最能激活最后的全连接层的输入的第一档买卖价数据，图形绘制如图 4-7 所示。

在图 4-7 中作者发现了一个有趣的现象，即梯度上升法得到的这个输入数据的尾部出现了方向截然相反的较高的峰。作者推测这个现象与一种称为买卖价格反弹的微观结构现象相对应，即当买入价和卖出价都有交易，但价格没有真正变动时的一种现象。

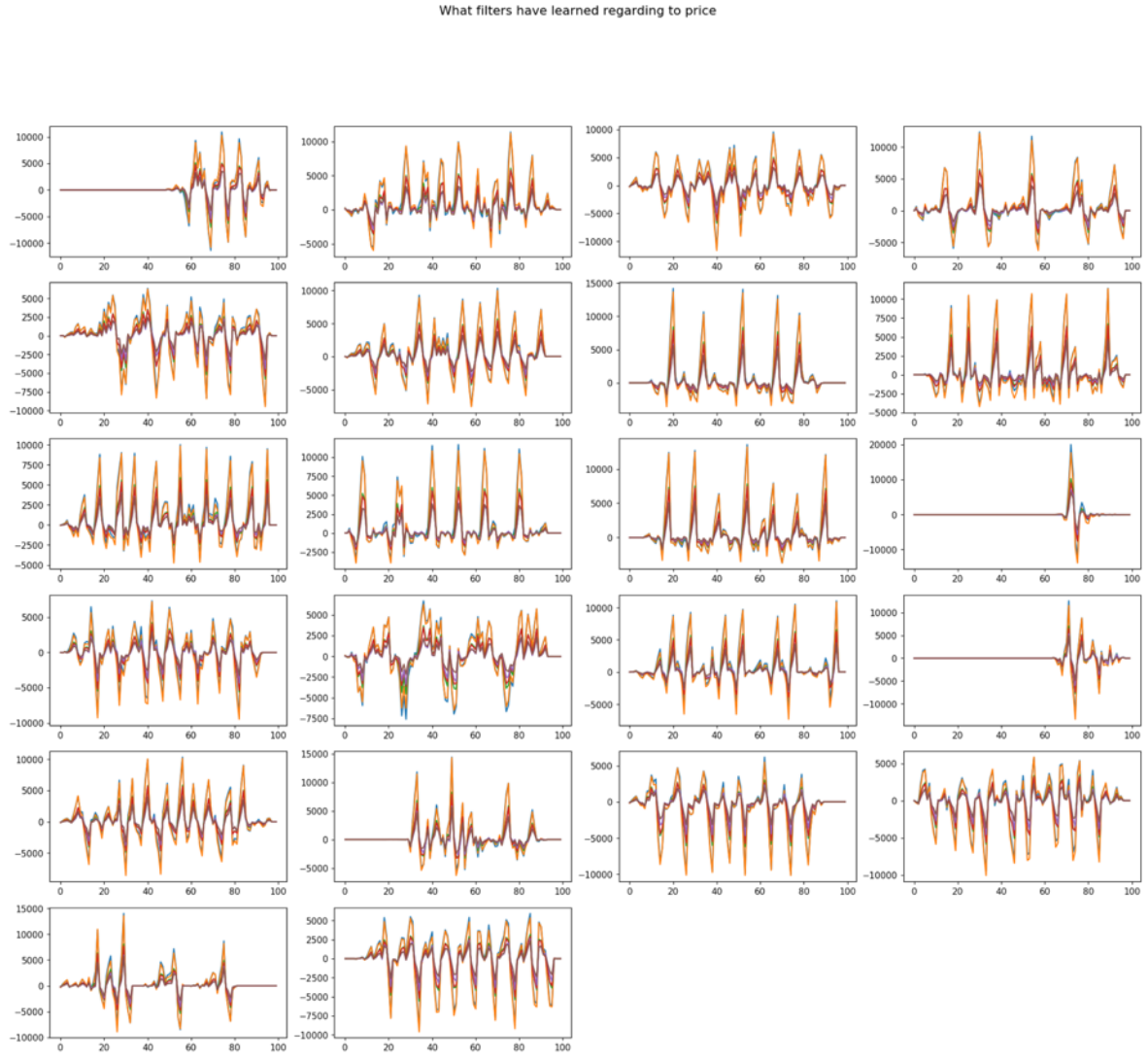


图 4-5 最后一个激活层学会了识别输入价格中这样的模式

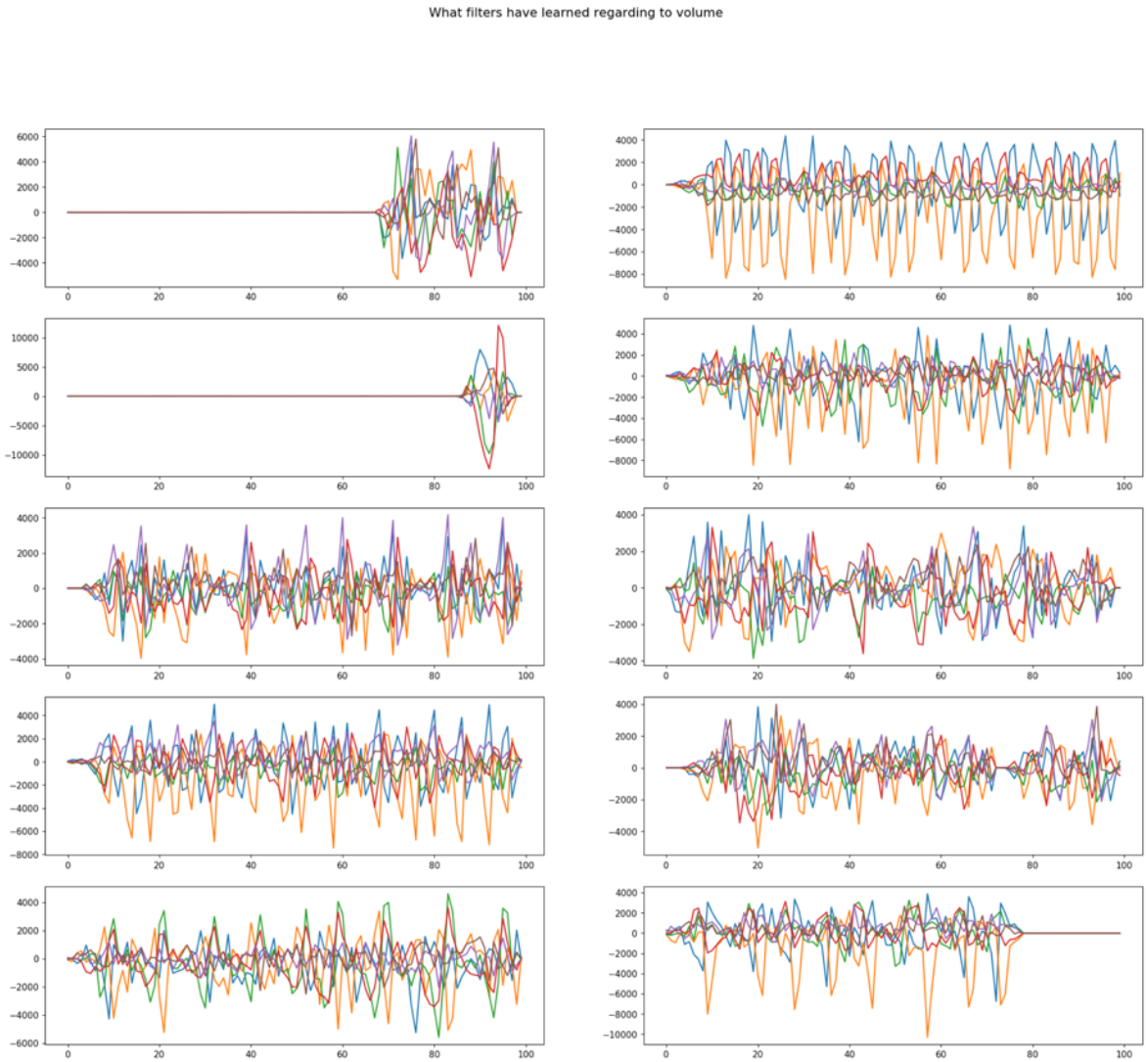


图 4-6 最后一个激活层学会了识别输入买卖量中这样的模式

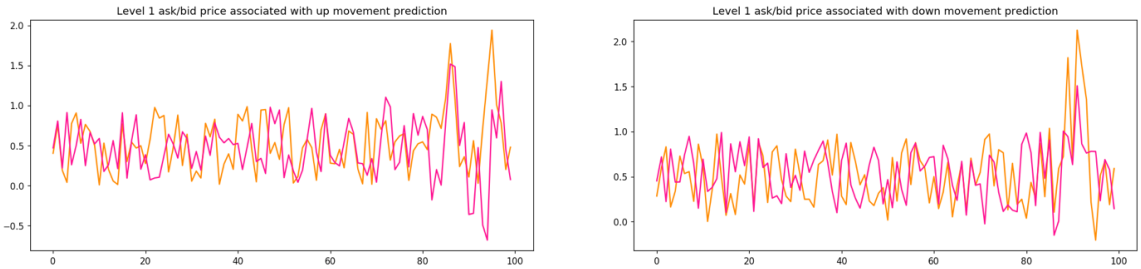


图 4-7 最能激活最后的全连接层的数据的模式

4.5 本章小结

本章 4.1 节中对在第 3 章中调试而得的各类最优模型进行了实测。此外，还在 4.2 节中进行了迁移学习、4.3 节中进行了特征选取和再预测，最后在 4.4 节中对以卷积神经网络模型为例，对本文 3.1 节中选取的预测方法的原理加以了解释。

结论

本文首先对适合于本课题的预测方法进行了选取，选择了使用深度神经网络作为预测的方法，随之针对本文的股价预测问题建立了长短期记忆循环神经网络模型、卷积神经网络模型、长短期记忆循环神经网络与卷积神经网络的融合模型以及空洞卷积模型，并对它们进行了参数择优。在经过对参数调优得到最优的模型结构及参数后，本文使用调优实验中得到的上述四种模型各自最优的模型进行了实测。实测结果远超任务书要求，文章对使用四种模型得到的预测效果进行了对比分析。另外，文章还进行了迁移学习、使用随机森林进行特征提取后再训练及测试以及以卷积神经网络为例，解释深度神经网络预测原理的三项额外工作。其中，特征提取工作为在实测中表现水准本不高的卷积神经网络模型带来了可观的性能提升。综合各个实验结果及对神经网络预测原理的分析，可以得出，本文基于 FI-2010 数据集构建的四种预测模型在高频交易的股价预测任务中得到了较好的实验结果且拥有潜在的应用价值。

未来的研究工作重点主要包含以下几个方面：

1、深究空洞卷积模型近乎 90%的预测准确率。因为其实如此高的准确率在实验时无法同样在未来 5 个周期的预测范围内得到，由于研究时间的限制，文章暂未证明此结果不具有偶然性；

2、使用更优的特征选择手段。本文仅用简单的随机森林就在评价指标上得到了如此大的提升，相信更精确的特征选择方式可以带来更大的精度提升；

3、将本文提出的模型运用于数据量更大、发布时间更新的限价订单簿数据集上；

4、使用更优的时间序列数据可视化方法，如 T-SNE；

致谢

望着高高的天，走了长长的路，幸而努力终有归处。

我将对家人最诚挚的谢意留在这里，对老师最温柔的告别留在这里，对学长和学长最美好的祝愿留在这里。

我有幸生于这样一个幸福的家庭里，没有我的父母、（外）祖父母为我付出的一切，我无法拥有今天的成就。家庭永远是我最坚实的后盾和最可靠的支柱。

我有幸遇到嵌入式技术实验室中一群相处如朋友般自然的老师，没有他们为我提供的教学资源，我无法顺利地自由开展研究。

我也有幸遇到这间实验室中爱养猫的学长和学长，愿我们一起跃入人海，各有灿烂千阳。

愿吾师吾友，顺遂平安。

年复一年，春绿冬藏，花开花落，人来人往。

他日若买桂花同载酒，愿还似，少年游。

参考文献

[1] Xingyu Zhou,Zhisong Pan,Guyu Hu,Siqi Tang,Cheng Zhao,Qian Zhang. Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets[J]. Mathematical Problems in Engineering,2018,2018.

[2] Henrique B M , Sobreiro V A , Kimura H . Stock price prediction using support vector regression on daily and up to the minute prices[J]. The Journal of Finance and Data Science, 2018, 4(3):183-201.

[3] Qureshi F I . Investigating Limit Order Book Characteristics for Short Term Price Prediction: a Machine Learning Approach[J]. arXiv e-prints, 2018.

[4] Zhang Z , Zohren S , Roberts S . DeepLOB: Deep Convolutional Neural Networks for Limit Order Books[J]. IEEE Transactions on Signal Processing, 2018, 67(11).

[5] Ayaz Hussain Bukhari,Muhammad Asif Zahoor Raja,Muhammad Sulaiman,Saeed Islam,Muhammad Shoaib,Poom Kumam. Fractional Neuro-Sequential ARFIMA-LSTM for Financial Market Forecasting[J]. IEEE Access,2020,8.

[6] Ntakaris Adamantios,Kanniainen Juho,Gabbouj Moncef,Iosifidis Alexandros. Mid-price prediction based on machine learning methods with technical and quantitative indicators[J]. PloS one,2020,15(6).

[7] Lanbouri Z , Achchab S . Stock Market prediction on High frequency data using Long-Short Term Memory[J]. Procedia Computer Science, 2020, 175:603-608.

[8] Tu, Y . Predicting High-Frequency Stock Market by Neural Networks[D]. Imperial College London Department of Mathematics, 2020.

[9] Gu Y , Yan D , Yan S . Price Forecast with High-Frequency Finance Data: An Autoregressive Recurrent Neural Network Model with Technical Indicators[C]// International Joint Conference on Artificial Intelligence. 2020.

[10] Yadav K , Yadav M , Saini S . Stock values predictions using deep learning based hybrid models[J]. CAAI Transactions on Intelligence Technology, 2021, 7(1).

[11] Wang, M . Essays on the Applications of Machine Learning in Financial Markets[D]. Columbia University, 2021.

[12] Yang Y , Wu Y , Wang P , et al. Stock Price Prediction Based on XGBoost and LightGBM[J]. E3S Web of Conferences, 2021, 275:01040.

[13] Zeng L , Wang L , Niu H , et al. Trade When Opportunity Comes: Price Movement Forecasting via Locality-Aware Attention and Adaptive Refined Labeling[J]. Papers, 2021, 2107:11972.

[14] Lv X , Zhang L . Residual Gated Recurrent Unit-Based Stacked Network for Stock Trend Prediction from Limit Order Book[M]. Springer, 2021.

[15] Ntakaris A , Magris M , Kanniainen J , et al. Benchmark dataset for midprice forecasting of limit order book data with machine learning methods[J]. Journal of Forecasting, 2018(4).

[16] Kercheval A N , Zhang Y . Modelling high-frequency limit order book dynamics with support vector machines[J]. Quantitative Finance, 2016, 15(8):1-15.

附录 1

本附录提供了正文中 3.1 节构建的四种模型的 Python 程序代码，以及 4.3 节特征选择时用到的程序代码。

代码①-1 长短期记忆循环神经网络模型代码

```
def RNN_first_try(input_shape):
    """
    第一次搭建 RNN 的尝试:
    LSTM + L2 -> Dropout -> FullyConnected + SoftMax

    传入参数:
    input_shape -- 数据集形状

    :return:
    model_RNN -- Keras 中 Model() 的一个 instance
    """

    # 模型输入为 input_shape 大小的 tensor
    X_input = Input(shape=input_shape)

    # LSTM layer
    X = LSTM(units=100,kernel_regularizer=l2())(X_input)

    # Dropout
    X = Dropout(rate=0.5)(X)

    # FullyConnected + SoftMax
    X = Dense(units=3, activation='softmax')(X)

    # Create a Keras model instance
    model_RNN = Model(inputs=X_input, outputs=X, name='RNN_first_try')

    return model_RNN
```

代码①-2 卷积神经网络模型代码

```
def Conv2D_3rd_try(input_shape):
    """
    第三次搭建 CNN 的尝试:
    Conv2D -> RELU -> Conv2D -> RELU -> Conv2D -> RELU -> MaxPool -> Conv2D -> RELU ->
MaxPool -> Flatten + FullyConnected(l2) + RELU -> 50% Dropout↓-> FullyConnected + SoftMax

    传入参数:
    input_shape -- 数据集形状

    :return:
    model_CNN -- Keras 中 Model() 的一个 instance
    """

    # 模型输入为 input_shape 大小的 tensor
    X_input = Input(shape=input_shape)
```

```
    # Conv2D -> RELU
    X = Conv2D(filters=16, kernel_size=(4, 40), strides=(1, 1), data_format='channels_last')(X_input)
    X = Activation('relu')(X)

    # Conv2D -> RELU
    X = Conv2D(filters=16, kernel_size=(1, 1), strides=(1, 1), data_format='channels_last')(X)
    X = Activation('relu')(X)

    # Conv2D -> RELU
    X = Conv2D(filters=32, kernel_size=(4, 1), strides=(1, 1), data_format='channels_last')(X)
    X = Activation('relu')(X)

    # MaxPool
    X = MaxPooling2D(pool_size=(2, 1), strides=2)(X)

    # Conv2D -> RELU
    X = Conv2D(filters=32, kernel_size=(3, 1), strides=(1, 1), data_format='channels_last')(X)
    X = Activation('relu')(X)

    # MaxPool
    X = MaxPooling2D(pool_size=(2, 1), strides=2)(X)

    # Output Layer: convert X to a vector + FullyConnected + RELU
    X = Flatten()(X)
    X = Dense(units=100, kernel_regularizer=l2(),activation='relu')(X)

    # 50% Dropout
    X = Dropout(rate=0.5)(X)

    # FullyConnected + SoftMax
    X = Dense(units=3, activation='softmax')(X)

    # Create a Keras model instance
    model_CNN = Model(inputs=X_input, outputs=X, name='Conv2D_3rd_try')

    return model_CNN
```

代码①-3 空洞卷积模型代码

```
def Conv1D_1st_try(input_shape):
    """
    第一次搭建 CNN1D 的尝试:
    Conv1D -> RELU -> Conv1D(with dilation) -> RELU -> Conv1D(with dilation) -> RELU ->
Conv1D(with dilation) -> RELU -> Conv1D(with dilation) -> RELU -> Flatten + FullyConnected(l2) +
RELU -> 40% Dropout-> FullyConnected + SoftMax

    传入参数:
    input_shape -- 数据集形状

    :return:
    model_CNN -- Keras 中 Model() 的一个 instance
    """

    # 模型输入为 input_shape 大小的 tensor
```

```
    X_input = Input(shape=input_shape)

    # Conv1D -> RELU
    X = Conv1D(filters=16, kernel_size=2, padding='causal', strides=1)(X_input)
    X = Activation('relu')(X)

    # Conv1D(with dilation) -> RELU
    X = Conv1D(filters=16, kernel_size=2, padding='causal', dilation_rate=2)(X)
    X = Activation('relu')(X)

    # Conv1D(with dilation) -> RELU
    X = Conv1D(filters=16, kernel_size=2, padding='causal', dilation_rate=4)(X)
    X = Activation('relu')(X)

    # Conv1D(with dilation) -> RELU
    X = Conv1D(filters=16, kernel_size=2, padding='causal', dilation_rate=8)(X)
    X = Activation('relu')(X)

    # Conv1D(with dilation) -> RELU
    X = Conv1D(filters=16, kernel_size=2, padding='causal', dilation_rate=16)(X)
    X = Activation('relu')(X)

    # Output Layer: convert X to a vector + FullyConnected + RELU
    X = Flatten()(X)
    X = Dense(units=64, kernel_regularizer=l2(),activation='relu')(X)

    # 40% Dropout
    X = Dropout(rate=0.4)(X)

    # FullyConnected + SoftMax
    X = Dense(units=3, activation='softmax')(X)

    # Create a Keras model instance
    model_CNN_1D = Model(inputs=X_input, outputs=X, name='Conv1D_1st_try')

    return model_CNN_1D
```

代码①-4 卷积神经网络与长短期记忆循环神经网络融合模型代码

```
def squeeze_axis(x):
    """
    去掉三维通道的最后一维(CNN 通道上面定义为了 channel-last)
    :param x:
    :return:
    """

    return squeeze(x=x,axis=2)

def CNN_RNN_2nd_try():
    """
    第二次搭建 CNN+RNN 的尝试:
    Conv2D -> RELU -> Conv2D -> RELU -> Conv2D -> RELU -> MaxPool -> Conv2D -> RELU ->
MaxPool -> LSTM + RELU -> 60% Dropout↑-> FullyConnected + SoftMax

    传入参数:
```

```
    input_shape -- 数据集形状

    :return:
    model_CNN -- Keras 中 Model() 的一个 instance
    """

    # 模型输入为 input_shape 大小的 tensor
    # X_input = Input(shape=input_shape)

    model_CNN_RNN = Sequential()

    # Conv2D -> RELU

    # X = Conv2D(filters=16, kernel_size=(4, 40), strides=(1, 1), data_format='channels_last')(X_input)
    # X = Activation('relu')(X)
    model_CNN_RNN.add(layer=Conv2D(input_shape=(100, 40, 1), filters=16, kernel_size=(4, 40),
strides=(1, 1),
                                   data_format='channels_last', activation='relu'))

    # Conv2D -> RELU
    # X = Conv2D(filters=16, kernel_size=(1, 1), strides=(1, 1), data_format='channels_last')(X)
    # X = Activation('relu')(X)
    model_CNN_RNN.add(
        layer=Conv2D(filters=16, kernel_size=(1, 1), strides=(1, 1), data_format='channels_last',
activation='relu'))

    # Conv2D -> RELU
    # X = Conv2D(filters=32, kernel_size=(4, 1), strides=(1, 1), data_format='channels_last')(X)
    # X = Activation('relu')(X)
    model_CNN_RNN.add(
        layer=Conv2D(filters=32, kernel_size=(4, 1), strides=(1, 1), data_format='channels_last',
activation='relu'))

    # MaxPool
    # X = MaxPooling2D(pool_size=(2, 1), strides=2)(X)
    model_CNN_RNN.add(layer=MaxPooling2D(pool_size=(2, 1), strides=2))

    # Conv2D -> RELU
    # X = Conv2D(filters=32, kernel_size=(3, 1), strides=(1, 1), data_format='channels_last')(X)
    # X = Activation('relu')(X)
    model_CNN_RNN.add(
        layer=Conv2D(filters=32, kernel_size=(3, 1), strides=(1, 1), data_format='channels_last',
activation='relu'))

    # MaxPool
    # X = MaxPooling2D(pool_size=(2, 1), strides=2)(X)
    model_CNN_RNN.add(layer=MaxPooling2D(pool_size=(2, 1), strides=2))

    # Squeeze channel dimension
    model_CNN_RNN.add(layer=Lambda(function=squeeze_axis))

    # LSTM + RELU
    # X = Flatten()(X)
    # X = Dense(units=100, kernel_regularizer=l2(),activation='relu')(X)
    # X = LSTM(units=100,kernel_regularizer=l2())(X)
```

```
    # X = Activation('relu')(X)
    model_CNN_RNN.add(layer=LSTM(units=100, kernel_regularizer=l2(), activation='relu'))

    # 60% Dropout
    # X = Dropout(rate=0.5)
    model_CNN_RNN.add(layer=Dropout(rate=0.6))

    # FullyConnected + SoftMax
    # X = Dense(units=3, activation='softmax')
    model_CNN_RNN.add(layer=Dense(units=3, activation='softmax'))

    # Create a Keras model instance
    # model_CNN = Model(inputs=X_input, outputs=X, name='CNN_RNN_2nd_try')

    return model_CNN_RNN
```

代码①-5 特征选择过程代码

```
from sklearn.ensemble.forest import RandomForestClassifier

from numpy.lib.function_base import flip
from numpy.core.fromnumeric import argsort


label_rows=[144,145,146,147,148]

classifier=RandomForestClassifier(n_estimators=100,max_depth=3,random_state=0,class_weight='balan
ced')

for i,j in enumerate(label_rows):
    model=classifier.fit(train_dataframe.iloc[:,range(144)].values,train_dataframe.iloc[:,j].values)
    y_hat=model.predict(test_dataframe.iloc[:,range(144)].values)


top_10_features=flip(argsort(classifier.feature_importances_),axis=0)[:10]
print(top_10_features) # output: [69 88 87 86 91 82 98 89 92 93]
```