東南大學

毕业设计(论文)报告

题 目：基于机器学习的股票价格与趋势预测

学　　号：07316131

姓　　名：王凯琳

学　　院：数学学院

专　　业：统计学

指导教师：钱成

起止日期：2020.1-2020.6

东南大学毕业(设计)论文独创性声明

本人声明所呈交的毕业(设计)论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

论文作者签名：王凯琳　日期：　2020 年　5　月　31　日

东南大学毕业(设计)论文使用授权声明

东南大学有权保留本人所送交毕业(设计)论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布(包括刊登)论文的全部或部分内容。论文的公布(包括刊登)授权东南大学教务处办理。

论文作者签名：王凯琳　导师签名：

日期：　2020 年　5　月　31　日　　日期：　2020 年　5　月　31　日

摘　　要

股票趋势预测因其具有重要的经济意义与研究价值，始终保持着较高的研究热度。近年来，深度学习理论在多个领域中崭露头角。在股票预测中，高频率的交易数据变化快、数量大、利润空间广的特点使其成为与深度学习理论相结合的最佳选项。

本文以基于堆叠式 LSTM 神经网络的高频股指价格趋势预测为研究目标，进行了理论介绍、基于 Python 的网络搭建与案例实证分析，最终得到了具有较强的趋势预测能力、泛化能力与获得超额收益能力的交易模型。

利用中国 A 股市场中的沪深 300、中证 500、上证 50 三只股票指数的分钟频数据，研究首先验证了以 ROCAUC 损失函数为优化目标的优势。其次，在平稳化的价格数据的基础上，通过将技术指标与 PCA 降维技术相结合，构建了在高频股票走势预测中具有效果提升作用的新特征集。基于以上网络结构与特征集，进行了多个对比实验，并利用不同的评价指标评估了 3 只股指在未来 1 分钟、5 分钟、15 分钟趋势预测中的表现。结果表明，模型在不同情形下，均具有良好的、稳定的预测性能，并能获得超过市场收益率的超额收益。

关键词：趋势预测，高频交易，堆叠 LSTM 神经网络，ROCAUC 损失函数，技术指标

ABSTRACT

This article focuses on forecasting stock index price trends with high-frequency data via the stacked LSTM neural network. The network is constructed on Python platform under Tensorflow framework. To evaluate the effectiveness of the proposed model, a series of case studies and comparison experiments are conducted, which suggest that the proposed LSTM neural network is capable of trend prediction with a decent accuracy and has adequate potentials for obtaining excess returns.

Stock index data used in this study are three influential stock indexes in the Chinese A-share market: CSI 300, CSI 500, and SSE 50. Firstly, this research verifies the advantages of setting ROCAUC loss function as the optimization objective function in the LSTM network when compared with Cross-Entropy loss function. Secondly, a novel feature set, which consists of stationary price features, principal components of technical indicators obtained via Principal Component Analysis(PCA), is constructed and proved to give better performances on this task. Afterwards, multiple comparison experiments are performed then evaluated under various evaluation criteria including ROCAUC, accuracy and returns. As the results suggest, the proposed model behaves well and stably when predicting the close price trend on the scale of 1, 5, and 15 minutes.

KEY WORDS: Trend Prediction, High Frequency Trading, Stacked LSTM Neural Network, ROCAUC Loss Function, Technical Indicators

目　　录

第一章 绪论

1.1 课题背景和意义

股票价格与趋势预测在许多领域都具有重要的研究价值与经济意义。股票价格的本质是时间序列，但由于其具有较大的波动性、非线性和高噪声的特点，传统的线性模型往往无法提供出令人满意的预测效果。因此，股票价格和趋势的预测始终被认为是十分困难的。近年来，随着计算能力的提高、算法的发展与研究者们对更高预测准确率的追求，神经网络逐渐成为了股票预测中的主流方法，越来越多的人们开始将其应用于金融数据的预测中，而长短期记忆神经网络则是其中的经典模型之一。与常规的神经网络不同，LSTM 网络所具有的选择性长短期记忆模块使得其在挖掘序列数据的时间依赖性上极具优势，因而更适合于股票价格的预测。本题中，将基于 LSTM 神经网络对国内 A 股股票指数进行高频的价格趋势预测，并尝试结合技术指标、降维等手段，以期获得具有更强预测能力、泛化能力与经济价值的模型。

1.2 研究现状

1.2.1 深度学习网络在股票预测中的崛起

股票价格是金融市场中最常见的时间序列之一，其预测问题在多年以来始终是人们的研究热点。股票的预测模型基本可以分为线性模型与非线性模型两大类。常见的线性模型大多依赖于线性假设或分布与独立性假设，面对非线性时间序列数据则难以给出准确的预测效果。因此，越来越多的研究者与金融行业从业者将目光转向了非线性模型。非线性模型包括支持向量机(SVM)，深层神经网络(DNN)等机器学习模型。由于大多数股票价格具有高噪声、非线性、非平稳的特征，经典的线性模型在股价预测中的适用性、准确性以及泛化能力都受到了极大的限制，而非线性预测模型因其能有效挖掘特征之间非线性关系，在股票价格、股票指数预测中的表现明显优于传统的线性模型，逐渐成为股票预测模型中的主流。例如[2]中结果就曾表神经网络模型表现均优于 ARIMA 线性模型。

随着计算能力的提高、算法的发展、数据量爆炸式的增长与研究者们对更高预测准确率的追求，深层神经网络(DNN)因其更强的拟合能力、优秀的预测性能在众多非线性预测模型中脱颖而出，成为在许多领域中炙手可热的算法模型。最初，DNN 更多地被应用于信号处理，语音识别和图像分类等领域，其卓越的表现引得越来越多的人们开始尝试将多种神经网络的变形应用于金融时间序列预测中。直观地讲，DNN 能够根据数据特征进行高度

的抽象和自动化的学习，在学习异质性信息过程中淡化无关因素、强化有效因素的作用，由此获得更好的金融数据预测效果。正如 Heaton 等在[4]中所概括的那样，深度学习在金融市场研究中具有：①对输入变量形式更强的包容性；②拟合变量间复杂非线性关系的有效性；③避免过拟合的有效性等明显的优势。

1.2.2 LSTM 网络在金融数据中的应用

相较于一般的 DNN 模型，深度递归神经网络(DRNN)更受到金融数据研究者的青睐。主要原因在于 RNN 成功地将“时间”这一概念融入模型中：它在学习当前时刻数据的同时，可以结合之前时间序列数据所形成的短期记忆以学习网络权重，这使得 RNN 成为了最适合应用于时间序列分析的神经网络结构。然而，RNN 最大的缺点之一，即在优化过程中易出现梯度消失或爆炸现象，使得 RNN 的学习与优化效率大打折扣，应用也受到了局限。

为了克服这一问题，长短时记忆(LSTM)网络应运而生[1],并逐渐被应用于信号传播、语音识别、文本识别等序列预测中，取得了很好的预测效果。直观来看，LSTM 网络作为 RNN 的变种之一，其独特之处在于在 RNN 网络结构的基础上添加了更多控制信息传递的参数与门单元，这些门单元使得网络能同时对历史信息与新信息进行过滤，形成对于学习有益的长期记忆和短期记忆。在参数学习方面，LSTM 有效克服了 RNN 在优化过程中易出现的梯度消失与梯度爆炸现象，使得其在挖掘序列数据长期依赖关系中极具优势。

近年来，研究者们开展了一系列针对 LSTM 网络在金融数据预测中应用效果的研究与探索。一些研究专注于各个神经网络与线性模型的预测效果对比：Hiransha Ma 和 Gopalakrishnan E.Ab 使用 NSE 和 NYSE 的每日收盘价作为数据集, 预测了两市场中五家不同公司的股价。实验显示，CNN 预测效果最优，而神经网络整体表现均优于 ARIMA 模型[2]。在[10]中，Di Persio 和 Honchar(2017)使用了多个 RNN 网络模型以预测谷歌股价走势, 结果表明 LSTM 神经网络的预测效果更优于一般的 RNN 与 GRU 神经网络。在[8]中，杨青等人基于全球范围内 30 个股票指数数据, 以 3 种不同期限的预测为目标, 基于 DLSTM、SVR、MLP 和 ARIMA 模型开展了对比实验，发现 DLSTM 神经网络拥有很强的泛化能力与最高的准确率。

模型对比之外，研究者们也尝试了数据平稳化、标准化、变量筛选等不同数据预处理方式对预测效果的影响(平稳化, TIs, PCA)：Gao, T[14]等人利用 Standard & Poor’s 500,

NASDAQ, 和 Apple (AAPL) 的日频数据对日收盘价进行预测。他们在原始特征(开盘价，收盘价，最高价，最低价，交易量)的基础上添加了多个股票技术指标作为输入特征，也尝试了利用 PCA 对技术指标进行降维后的新特征集的预测效果。实验对比显示，利用原始股票特征和 PCA 降维后的技术特征集作为输入特征的网络预测效果最好。Chih-Hung 等人基于比特币的每日价格，以 MSE，RMSE，MAE 和 MAPE 作为评价标准，对比了两种不同的 LSTM 模型(常规 LSTM 模型和 AR(2)LSTM 混合模型)的表现, 并发现 AR(2)LSTM 混合模型表现优于传统的 LSTM 模型，为时间序列作为输入数据的预处理提供了新的思路[1]。在 Li 等人的文章 [13]中,作者选取了指数数据、交易量和技术指标作为输入特征，将实时滤波技术和 LSTM 网络相结合，对 HSI, SSE, SZSE, TAIEX, NIKKEI, and KOSPI 多个指数数据进行了预测，实验发现这一混合模型相较于经典的 LSTM 模型有明显的更好的预测效果。 [12]中建立了将新闻特征融入 DLSTM 的预测模型，实验发现此模型具有更高的预测准确率及更强的鲁棒性。

LSTM 另有两种变形。双向 LSTM(BLSTM)同时利用了预测值前、后的时间序列作为输入序列，以更大程度开发数据在不同时间方向上的相关关系；堆叠式 LSTM(SLSTM)网络则由多个 LSTM 层逐层堆叠构建而成，以执行更深程度的学习，通常更适合于捕获在时间尺度上较为复杂的模式。这两种模型各有所长，也适用于不同的场景。如在[3]中，作者将双向 LSTM(BLSTM)和堆叠式 LSTM(SLSTM)与简单 LSTM 网络的预测效果进行了比较，结果表明双向 LSTM 表现最佳。

1.2.3 研究现状总结与展望

在近期发表将 LSTM 应用于金融数据预测的文献中，前期研究的目标主要集中在线性模型与深度学习模型或不同的深度学习模型之间的预测效果之间的对比[2]，但存在样本数据涉及的股票类型与数量不够完备(以国外市场个股为主)、训练数据集时间期限不统一、预测期限不统一等问题，缺乏对 LSTM 神经网络在金融数据预测中普适性的探索；近期的一些研究者们开始在数据预处理[1][5]、网络内部结构的优化与扩展[5]、cost 函数、正则化[7]、优化器及其学习率的选取等方向进行了探索，但存在研究对象数量不充足、特殊性强，缺乏对其普适性验证的问题；在研究对象上，多以国内外个股数据的小时频或日频及以下频率的价格预测或趋势预测，而鲜少以分钟频或更高频率的国内股票指数为研究对象。因此，本文将基于多只国内股票指数的分钟频数据，对其进行高频的价格预测或趋势预测。

1.3 本文研究内容

本文以高频的股票指数趋势预测为基本研究目标，通过将 LSTM 神经网络、技术分析和 PCA 技术相结合的手段进行文献阅读与理论方法学习，最终利用 Python 中的 Tensorflow 框架进行网络搭建与高频量化交易的实证。

在理论与方法部分，作者通过对相关文献与书籍的阅读，了解并叙述了 LSTM 神经网络的结构搭建、超参数优化、优化算法、股票技术分析方法与主成分分析法等方法的原理与在股票高频交易中研究现状，具体描述了在实证部分搭建网络所利用的技术方法。

实证部分，本文基于我国 A 股市场中的沪深 300、中证 500、上证 50 三只股票指数在 2018.1-2019.6 期间的分钟频数据，首先构建了基本数据集(特征集 1)、含技术指标的数据集(特征集 2)、对非价格特征 PCA 降维后的数据集(特征集 3)共三个数据集，通过对比实验，验证技术指标与 PCA 方法在提高高频数据预测准确率上的积极作用；而后利用在网络中具有较高预测准确率、较低计算代价的特征集 3 实现在不同股票指数、不同预测时间长度上的性能测试，验证了所搭建网络具有较强的趋势预测能力、泛化能力与通用性。

第二章 神经网络构建的相关理论

本章主要介绍研究中在数据处理、模型选择和网络构建过程中重要的概念和理论，以更好地辅助大家理解深度 LSTM 神经网络模型的作用机制、训练过程，并进行结果解读。

2.1 LSTM 神经网络

2.1.1 人工神经网络(ANN)

人工神经网络是一种非线性的函数逼近器。顾名思义，它是研究者受自然界中生物神经网络的启发，模仿其结构与信息传递机制而构建的人造的神经网络。其运行机制与生物神经元相似：它通过权重矩阵与非线性激活函数的组合模拟神经元之间的复杂的信息传递方式，从而实现像生物一样自主学习的目标，旨在从数据中概括、识别出潜在而本质性的趋势，完成对输出变量和输入变量之间复杂关系的建模。一般人工神经网络的首层与末层分别是输入层和输出层，都是单层结构；而中间层均被称为隐藏层，层数不限。信息从输入层流入网络，后传递至隐藏层进行处理，再流向输出层，得到最终的输出值。ANN 的主要优点是它能够从数据中学习普遍的模式，这是许多常规方法都无法做到的。

前馈神经网络(FNN)是人工神经网络中的一种，它采用了一般人工神经网络的基础框架。其各层神经元之间的连接方向是链状单向的，信息的传递是无反馈的，即神经元都只能从上一层接收输入，向下一层进行输出，同层的神经元间无法相互连接，因此被称为“前馈”。一般 FNN 的结构图由图 1 给出：

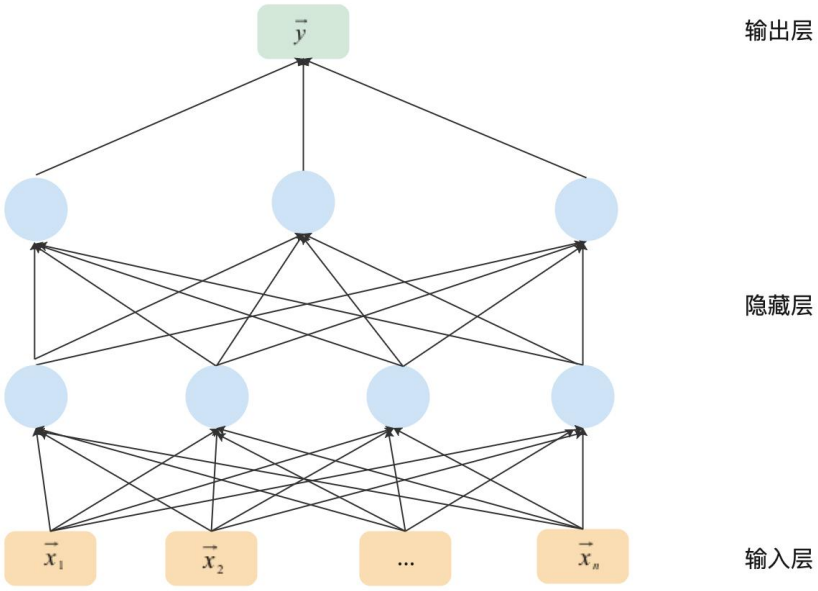

图 1 一般 FNN 的结构图

2.1.2 循环神经网络(RNN)

作为人工神经网络的变种，RNN 具有一般 ANN 的结构特征。但不同于 FNN，在输入形式上，RNN 允许以二维序列的形式作为输入，相比 FNN 增加了一个维度；在学习方向上，RNN 可以按照序列的演进方向进行循环式的学习，对于时间序列而言，该学习器则具有了形成“短期记忆”的能力，可看作是在时间维度上共享权值参数的神经网络。图 2 给出了按时间维度展开的 RNN 网络结构图[15]:

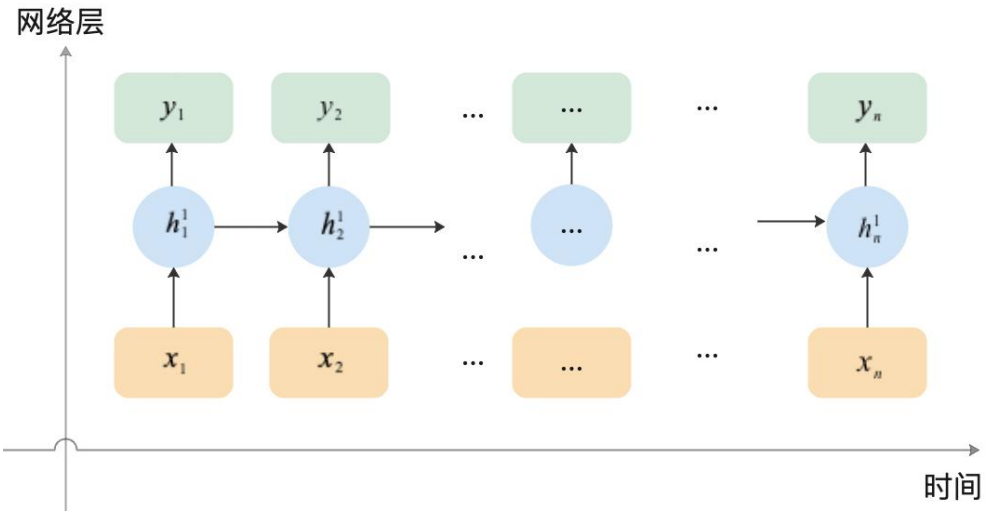

图 2 按时间维度展开的 RNN 网络结构图

图 3 给出了 RNN 的细节解剖图:

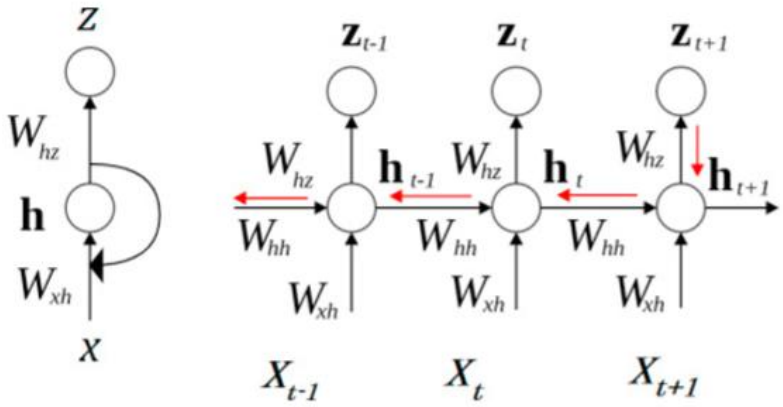

图 3 RNN 的细节结构图

在 RNN 中，其在 t 时刻的隐藏层状态 ht 不仅依赖于当前时刻x_t的信息，更取决于h_{t-1}，即过去的“记忆”。给定输入时间序列$X_{1:T} = (x_1, x_2, \ldots, x_t, \ldots, x_T)$，RNN 通过公式(1)(2)更新隐藏层状态 h:

$$z_t = Uh_{t-1} + Wx_t + b \tag{1}$$

$$h_t = f(z_t) \tag{2}$$

其中,z_t为隐藏层的净输入，U、W 为权重矩阵。h_t也常被称为状态(State)或者隐状态(Hidden State)。

从数学上来讲，RNN 隐藏层也可以被称为是一个动力系统[①]。理论[②]上，RNN 可以近似于任意非线性动力系统。

尽管 RNN 在理论上具有挖掘序列的时间依赖性的能力，但在实际应用中非线性激活函数的导数$f'(\cdot)$通常不为 1，且参数学习采用的随时间反向传播(BPTT)算法的本质是梯度下降优化算法，在挖掘数据长时间依赖性时，常常会出现梯度爆炸[③]或者梯度消失的现象，导致网络学习效率大打折扣，这是循环神经网络最大的缺点之一。通过对网络结构进行改造和优化，如添加门控机制等方法，研究者们对这一缺点进行了改善和弥补。

2.1.3 长短时记忆(LSTM)神经网络

---

① 动力系统(Dynamical system)是数学上的一个概念，指一个给定空间(如某个物理系统的状态空间)中所有点随时间按照一种固定的规则变化的系统。例如描述钟摆晃动、管道中水的流动等等的数学模型都是动力系统。在动力系统中有所谓状态的概念，状态是一组可以被确定下来的实数。动力系统的演化规则是一组函数的固定规则，它描述未来状态如何依赖于当前状态的。这种规则是确定性的，即对于给定的时间间隔内，从现在的状态只能演化出一个未来的状态。

② 定理 6.1 循环神经网络的通用近似定理 [Haykin, 2009]：如果一个完全连接的循环神经网络有足够数量的 sigmoid 型隐藏神经元，它可以以任意的准确率去近似任何一个非线性动力系统。

③ 梯度消失与梯度爆炸问题(Vanishing gradient and exploding gradient problem)常常出现在以梯度下降法和反向传播训练人工神经网络的时候。在每次训练的迭代中，神经网络权重的更新值与误差函数的偏导数成比例，在激活函数导数只偏小或偏大是，梯度值会几乎消失或爆炸，使得权重无法得到有效更新，甚至神经网络可能完全无法继续训练。

RNN 的变种之一，长短时记忆(LSTM)神经网络，是由 Hochreiter 和 Schmidhuber 于 1997 年引入的[20]。与一般 RNN 相比，LSTM 的内部结构是更加复杂的：传统的 RNN 通过简单的加权组合与非线性映射形成反馈回路，但 LSTM 的记忆模块拥有更多的参数和控制记忆的门控单元。它通过门的开关决定记忆的保留与更新，实现了具有“记忆长，更新快”特点的选择性记忆更新机制。LSTM 神经细胞如图 4 所示[15]：

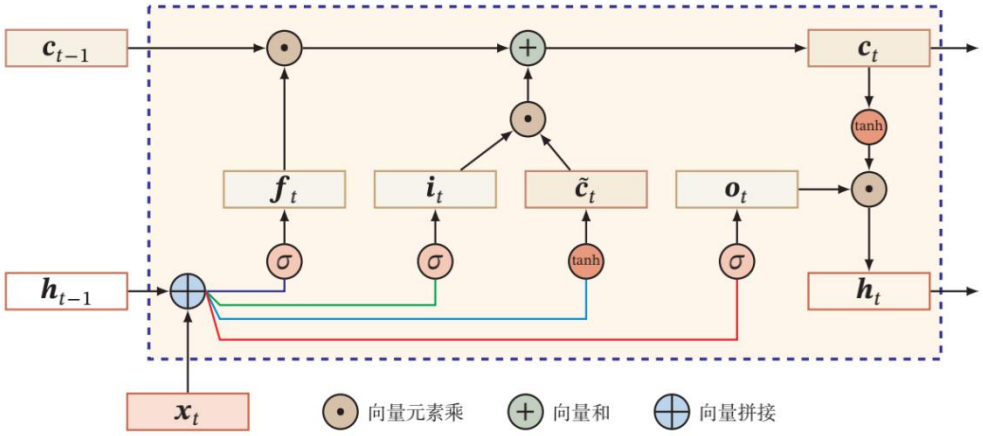

图 4　LSTM 神经细胞

门控机制(Gating Mechanism)用于判断来自历史信息与新信息的保留、传递与更新，它由输入门 i_t、遗忘门 f_t 和输出门 o_t 三个“门”组成，具体计算公式为：

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$

细胞状态 c_t 定义为：

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \#(6) \tag{6}$$

$$\tilde{c}_t = \tanh\left(W_g x_t + U_g h_{t-1}\right) \tag{7}$$

隐藏层状态，即该层的输出值为：

$$h_t = o_t \circ \tanh(c_t) \tag{8}$$

其中，$x_t \in \mathbb{R}^M$为网络在 t 时刻的输入，表示$\sigma(\cdot)$表示 sigmoid 激活函数。网络参数包

括：W、U 权重矩阵和偏差 b。

此外，由于门控机制的添加，LSTM 神经网络在一定程度上避免了梯度爆炸或者消失的问题，使得长短期记忆网络成为了能真正有效挖掘序列时间依赖性的循环神经网络。尽管比一般的 RNN 训练代价更高，LSTM 在挖掘时间序列数据特征中所特有的优势仍吸引着无数的研究者，并成为了时间序列预测中最受欢迎的神经网络模型。

2.1.4 堆叠长短时记忆(LSTM)神经网络

在一般 LSTM 层的基础上，可以通过增加网络的深度来实现对 LSTM 神经网络性能的进一步提升。在这里，网络的深度可以理解为输入x_t到输出y_t所经过变换的复杂程度。增加网络深度最直接的方法就是增加隐藏层的层数，形成层层堆叠的堆叠长短时神经网络。此时，在 t 时刻第 n-1 个隐藏层的输出 $h_t^{(n-1)}$则是在 t 时刻第 n 个隐藏层的输入，按时间和深度展开的堆叠 RNN 如图 5 所示：

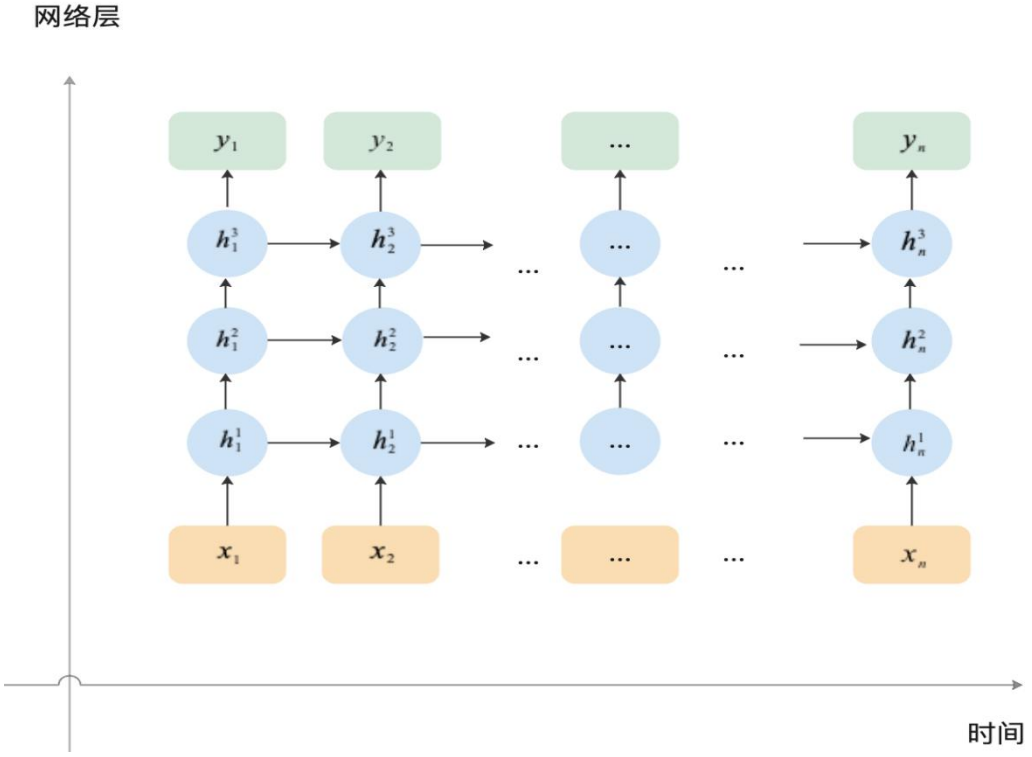

图 5　按时间和深度展开的堆叠 RNN 示意图

2.2 激活函数(Activation Function)

激活函数的采用实际上也是人工神经网络对生物神经的作用机制进行模拟的一种体现。采取激活函数的人工神经元则通过函数映射的方式模拟神经元被“激活”的过程，从而决定信息的传递与否和信息的内容。

在神经网络中，因为激活函数决定了隐藏层和输出层各个神经元的输出值，因此也在很大程度上决定了网络参数的学习与优化效率，如前文所提及的梯度爆炸与梯度消失问题就与 sigmoid 型激活函数的导数值直接相关。可以说，激活函数的选择对于神经元乃至整个神经网络都是十分关键的，这也是目前的研究热点之一。

早期的神经网络主要采取 Sigmoid 型的激活函数，而随着众多实验的开展与问题的涌现，分段线性函数逐渐成为了深度神经网络中最受欢迎的激活函数。本节将重点介绍几种常用的激活函数的定义、优劣势和适合的应用场景。

2.2.1 Sigmoid 型函数

Sigmoid 型函数因为图像均呈 S 型，因此也被称为 S 型函数。其中最常见的两个函数的具体定义如下：

1) Logistics 函数：

$$\mathrm{l}(\mathrm{x}) = \frac{1}{1+\mathrm{e}^{-\mathrm{x}}} \tag{9}$$

$$\mathrm{l}'(\mathrm{x}) = \mathrm{l}(\mathrm{x})\big(1 - \mathrm{l}(\mathrm{x})\big) \tag{10}$$

2) Tanh 函数：

$$\tanh(\mathrm{x}) = \frac{\mathrm{e}^{\mathrm{x}}-\mathrm{e}^{-\mathrm{x}}}{\mathrm{e}^{\mathrm{x}}+\mathrm{e}^{-\mathrm{x}}} \tag{11}$$

$$\tanh'(\mathrm{x}) = 1 - \big(\tanh(\mathrm{x})\big)^2 \tag{12}$$

可见，此类函数定义域为实数域，值域为(0,1)。当 x 取值大时，函数值接近于 1，神经元活跃；当 x 取值较小时，函数值接近于 0，神经元几乎不被激活，这和生物学上的理解是相符合的。以上两种 S 型激活函数形状见下图 6：

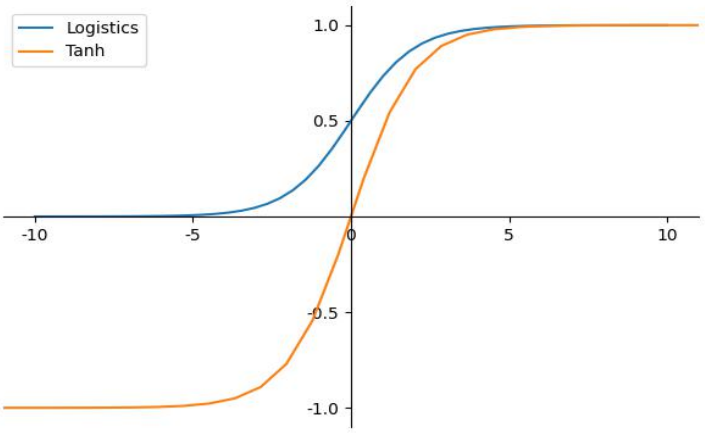

图 6 两种 S 型激活函数

S 型函数虽然具有全局连续可导的良好数学性质，但由于其为两端饱和④函数，因此在使用时容易出现梯度消失的现象，使得依赖梯度的网络学习变得困难，这也是它最大的缺点之一。

由于其具有将实数“压缩”至(0,1)区间的性质，S 型函数可以直接被理解为一个概率分布函数，因而常被应用于二分类网络的输出层。同时，使用 S 型激活函数的神经元也可被应用于门限机制，如 LSTM 神经网络中的输入门、遗忘门、输出门即采用了 S 型函数进行计算以控制信息的传递。

2.2.2 分段线性激活函数

分段线性激活函数中，以 ReLU 及其变形函数为代表的一类激活函数是目前最受欢迎的激活函数族。

1) ReLU(Rectified Linear Unit，修正线性单元)：

$$ReLU(x) = max(0, x) \tag{13}$$

④ 饱和函数：对于函数 f(x)，若 $x \to +\infty$时，$f'(x) \to 0$，则称 f 为左饱和；若 $x \to -\infty$时，$f'(x) \to 0$，则称 f 为右饱和；若同时满足左、右饱和，则称其为两端饱和。

ReLU 函数的特点在于：(1)当 x < 0 时，函数值始终为 0，神经元处于未激活状态，此时的神经网络因具有稀疏性特征而更具有生物解释性。事实上，在人类神经网络中，同一时刻处于激活状态的神经元个数仅占一小部分的。(2)当 x > 0 时，函数值为 x,导数为 1。从优化的角度来看，ReLU 更不容易出现 S 型激活函数所带来的梯度消失的问题，因而学习效率相对更高更稳定。

尽管 ReLU 在一定程度上解决了 S 型函数所存在的问题，但其对负值 x 统一取 0 的定义方法导致了一部分神经元在训练中容易始终处于不被激活的状态，因此造成网络无法正常学习的现象。针对此问题，人们提出了 ReLU 的几个变种，他们都很好的克服了 ReLU 神经元“死亡”的现象。

2) LReLU(Leaky ReLU,带泄露的 ReLU)

$$LReLU(x) = \begin{cases} x, \ if \ x > 0 \\ \alpha x, \ if \ x \leq 0 \end{cases} \tag{14}$$

此时的α是一个固定的、很小的常数，常取 0.01。也因此，LReLU 避免了 ReLU 神经元因未激活而出现的学习无效的现象。

3) PReLU(Parametric ReLU,带参数的 ReLU)

$$PReLU(x) = \begin{cases} x, \ if \ x > 0 \\ \beta x, \ if \ x \leq 0 \end{cases} \tag{15}$$

不同于 LReLU，此时的β是一个可被学习的参数，而非是一个固定的常数。对于 LReLU，事先确定的常数参量α，即 x ≤ 0 时函数的导数值，往往无法使网络达到足够好的学习效果，因此才提出了 PReLU。通过增加仅一个可学习参数的微小代价，PReLU 在学习过程中确定β的最优值，在避免神经元死亡的前提下又保证了网络的学习效率。

2.3 网络优化

对于一个结构确定的循环神经网络，获得一个拟合效果好，泛化能力强的网络并不是一件容易的事。尽管循环神经网络对序列的时间依赖性具有很强的拟合能力，但如果选取的学习准则不合理、优化算法效率低、网络过拟合现象严重，网络在实际应用中的效果都会大打折扣。因此，本节将重点介绍针对 LSTM 神经网络的 loss 函数、优化算法、避免过

拟合化方法的选取。

2.3.1 损失函数(Loss Function)

损失函数，又称代价函数(Cost Function)，是人为定义的、用来度量预测值和真实值之间差异的函数。

Loss 函数作为第一学习准则，将直接关系到网络的预测效果。合理的 Loss 函数，首先应当是一个合理的损失度量标准，此外，也应该结合优化算法的需求，采取更适合于计算的函数形式，使得参数学习与网络优化的过程更高效地进行。

下面将列出在定量预测和分类问题中常用的 Loss 函数：

1) 均方误差(Mean Square Error,MSE)

均方误差的本质是平方损失函数，一般用于连续变量的预测问题中。具体表达形式为：

$$L\left(y\middle|f(x)\right)=\sum_{N}\left(y-f(x)\right)^{2} \tag{16}$$

其中，y 为真实的标签值，f(x)为预测值，N 为样本个数。

2) 交叉熵损失函数(Cross Entropy Loss)

交叉熵在形式上属于负对数似然函数的一种。其具有多种变形，这是分类问题中最常使用的一类损失函数，具体定义如下：

令 y 是 K 维的 one-hot 标签，它代表着标签真实的概率分布，即：若 y 属于第 k 类，则 y 的第 k 维为 1，其他维均取 0；同样的，输出值 f(x)也是 K 维，第 k 维的取值代表该样本属于第 k 类的预测概率值，它代表着已知 x 的条件下，网络预测的概率分布，则此时的交叉熵表达式为：

$$\mathrm{L}\left(\mathrm{y}\middle|\mathrm{f}(\mathrm{x})\right)=\sum_{\mathrm{n}=1}^{\mathrm{N}}\mathrm{CE}_{\mathrm{n}} \tag{17}$$

where:

$$\mathrm{CE}_{\mathrm{n}}=-\sum_{\mathrm{k}=1}^{\mathrm{K}}\mathrm{y}_{\mathrm{k}}\log\left(\mathrm{f}_{\mathrm{k}}(\mathrm{x})\right)$$

$$\sum_{\mathrm{k}=1}^{\mathrm{K}}\mathrm{y}_{\mathrm{k}}=1\,,\mathrm{y}_{\mathrm{k}}\in\{0,1\}$$

$$\sum_{\mathrm{k}=1}^{\mathrm{K}}\mathrm{f}_{\mathrm{k}}(\mathrm{x})=1,\mathrm{f}_{\mathrm{k}}(\mathrm{x})\in\ [0,1]$$

3) ROC 曲线下面积(ROCAUC)

接收者操作特征曲线(Receiver Operating Characteristic curve，ROC 曲线)是一种坐标图型分析工具，ROC 曲线下面积(ROCAUC)也是二分类问题中常用的模型评价指标之一。具体定义如下：

对于二分类问题，即标签和输出结果只有两种分类的问题，可以根据真实标签 y 与预测值$\hat{y}$将结果定义为真阳性、伪阳性、真阴性、伪阴性[⑤]四种。对四种分类结果进行整合、计数得到混淆矩阵(Confusion Matrix)，形式如下：

	Positive	Negative
Positive_hat	TP	FP
Negative_hat	FN	TN

表 1 混淆矩阵

ROC 曲线以伪阳性率[⑥](FPR)X 轴，真阳性率[⑦](TPR)为 Y 轴，计算公式见(18)(19)。对二分类样本集进行分类阈值与(FPR, TPR)形成一一映射后进行绘制，即可形成的该二分类样本集对应的 ROC 曲线。

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{18}$$

$$FPR = \frac{FP}{FP+TN} \tag{19}$$

ROCAUC(Area Under Curve)指 ROC 曲线下的面积。一般地，AUC 越接近 1，分类器

⑤ 真阳性(True Positive,TP)：预测值为阳性，真实值为阳性；

伪阳性(False Positive,FP)：预测值为阳性，真实值为阴性；

真阴性(True Negative,TN)：预测值为阴性，真实值为阴性；

伪阴性(False Negative,FN)：预测值为阴性，真实值为阳性。

⑥真阳性率(TPR)：又被称为召回率(Recall)，表示在所有真实值为阳性的样本中，预测值也为阳性的比例。

⑦伪阳性率(FPR)：表示在所有真实值为阴性的样本中，得到错误的判断比例，即预测值为阳性的比例，

的分类效果越好；AUC 越接近 0.5，或 ROC 形状接近随机猜测线[⑧]，则分类器分类效果与随机猜测越相近，分类器失效；；若 AUC 小于 0.5，则分类效果劣于随机猜测。

与精确度、准确率等分类器的评价标准相比，ROCAUC 是一种对分类能力更为客观和全面的度量标准。原因在于 ROCAUC 同时追踪了分类器的灵敏度和误诊率，使得在此标准下的好的分类器能够在即保证高的召回率又保持较低的误诊率，这在实际问题中是非常具有意义的。如在股票趋势预测中，高灵敏度虽然能使得投资者尽可能多的把握住上涨的机会，但如果只关注灵敏度的提升而忽略“误诊”的代价，错把跌势当涨势，此时一个错误的决定将会带来直接的损失，因此，这种“错识”的代价往往比“错失”更高，因而对误诊率的控制是十分必要的。

然而，AUC 作为依赖于排序的评价方式，其本身是不连续且不可导的，这给针对 AUC 的直接优化带来了困难。因此，许多学者也针对这一问题提出了 AUC 的替代损失函数。受篇幅所限，在此不再展开介绍，详细内容参考[16]。

2.3.2 优化算法

神经网络的优化问题始终非常具有挑战性。从优化的角度看，神经网络具有参数多、网络结构多样和高度的非线性的特点，因此其优化问题属于比较棘手的高维空间中的非凸优化问题，鞍点众多，找到全局最优解是非常困难的；从计算成本的角度看，随着数据集规模的增大，网络无法承受基于梯度的传统优化算法利用整个数据集计算梯度的代价，因此基于部分样本的小批量梯度下降法(Mini-Batch Gradient Descent)[⑨]成为主流，而由此带来的批量大小、学习率等超参数也增加了网络学习的难度；同时，神经网络也常常面临着参数过拟合的现象，这同样也给网络优化问题增加了难度。

针对以上问题，本节将介绍目前最常用且有效的优化算法、参数初始化方法和防止过拟合的方法。

[⑨]随机分类线：ROC 空间的随机分类线值从 (0, 0) 到 (1,1) 的对角线，它将 ROC 空间划分为左上 / 右下两个区域，在这条线的以上的点代表了一个好的分类结果(胜过随机分类)，而在这条线以下的点代表了差的分类结果(劣于随机分类)。

1) Adam 优化算法

对于给定的数据集，学习率 ε 和梯度的估计 G_θ 直接决定了参数 θ 优化的方向与速度，一般的优化过程可以表达为：

$$\theta \leftarrow \theta - \frac{\epsilon}{n}\sum_{i=1}^{n} G_\theta(x^{(i)}, y^{(i)}) \tag{20}$$

在实际训练过程的后期，固定的学习率容易因为步长过大造成最优解的错失，一阶梯度方向也会出现震动，固定的梯度方向计算方法会造成更大的不稳定性。因此，一个合理的、能够自适应地调节学习率和梯度方向方法将会大大提高优化的效率。

自适应动量估计(Adaptive Moment Estimation，Adam)算法即是一种综合了学习率调整和梯度估计修正的综合算法。也是目前在深度网络学习中最稳定、优化效率最高、效果最好的算法之一。具体计算方法见下图 7：

Adam 算法

说明：下列流程中，g_t^2表示对按元素进行平方计算，即$g_t \odot g_t$。β_1^t和β_2^t指β_1与β_2的 t 次方。一般的，网络初始参数设定值为：α=0.001，β_1 = 0.9，β_2 = 0.999，$\varepsilon = 10^{-8}$。

参数说明：

α：步长，$\beta_1, \beta_2 \in [0,1)$: 指数衰减率，f(θ): 含参数θ的目标方程

参数初始化：

θ_0：初始参数向量，$m_0 \leftarrow 0$: 初始化梯度一阶矩，$v_0 \leftarrow 0$: 初始化梯度一阶矩，$t \leftarrow 0$: 初始化时间 t

过程：

当θ_t未收敛时：

t ←t+1

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (计算 t 时刻目标函数的梯度)

$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ (更新有偏的一阶估计量)

$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ (更新有偏的二阶估计量)

$\widehat{m_t} \leftarrow m_t/(1-\beta_1^t)$ (计算修正后的的一阶估计量)

$\widehat{v_t} \leftarrow v_t/(1-\beta_2^t)$ (计算修正后的的二阶估计量)

$\theta_t \leftarrow \theta_{t-1} - \alpha\widehat{m_t}/(\sqrt{\widehat{v_t}} + \varepsilon)$ (更新参数θ)

结束循环

返回最终参数θ_t

图 7 Adam 算法

2) 参数初始化方法

优化问题中的参数初始化方法十分关键，下面主要介绍对网络进行首次初始化时常用的随机初始化(Random Initialization)方法和预训练初始化(Pretrained Initialization)方法。随机初始化方法一般指对各个参数按照一定分布进行随机初始化的方法。这里介绍一种最常

用的、基于方差缩放[10]的参数初始化方法：Xavier 初始化。详细内容可见[15]中的 7.3 节。

Xavier 初始化：

对于神经网络第 L 层的第 m 个神经元$a_m^{(L)}$, 设第 L 层有$M^{(L)}$个神经元, 第 L-1 层有$M^{(L-1)}$个神经元，则此时权重 w 的初始化公式如表[2]所示：

表 2 Xavier 初始化表

激活函数	均匀分布	正态分布
Logistics	$w \sim U\left[-4\sqrt{\frac{6}{M^{(L-1)}+M^{(L)}}}, 4\sqrt{\frac{6}{M^{(L-1)}+M^{(L)}}}\right]$	$w \sim N\left(0, 16 \times \frac{2}{M^{(L-1)}+M^{(L)}}\right)$
Tanh	$w \sim U\left[-\sqrt{\frac{6}{M^{(L-1)}+M^{(L)}}}, \sqrt{\frac{6}{M^{(L-1)}+M^{(L)}}}\right]$	$w \sim N\left(0, \frac{2}{M^{(L-1)}+M^{(L)}}\right)$
恒等函数	$w \sim U\left[-\sqrt{\frac{6}{M^{(L-1)}+M^{(L)}}}, \sqrt{\frac{6}{M^{(L-1)}+M^{(L)}}}\right]$	$w \sim N\left(0, \frac{2}{M^{(L-1)}+M^{(L)}}\right)$

3）Dropout 法

作为具有强大拟合能力的机器学习模型，神经网络在没有监测机制的情况下，往往会出现过拟合现象。对于神经网络一类复杂的机器学习模型，泛化能力是评价其好坏的最重要的标准之一，因此采用适当的防止过拟合的机制与方法是重要且必要的。经典机器学习模型中常用的l_1, l_2正则化方法在深度神经网络中效果并不好，而早停(Early Stopping)法、丢弃(Dropout)法则在实践中被证明更为有效。下面将详细叙述丢弃法的原理与机制：

丢弃法，又称 Dropout 法，是指在层与层之间随机丢弃部分神经元及其连接的方法。训练过程中，该层的各个神经元以概率 p 被保留，即每个神经元的保留独立、同分布地服从于以 p 为参数的二项分布。而在测试过程中，不再使用 Dropout 法，即对所有神经元进行保留,示意图如下图 8：

[10]方差缩放：初始化一个深度网络时，为了缓解梯度消失或爆炸问题，而尽可能保持每个神经元的输入和输出的方差一致，根据神经元的连接数量进行自适应的调整初始化分布的方差，这类方法称为方差缩放(Variance Scaling).

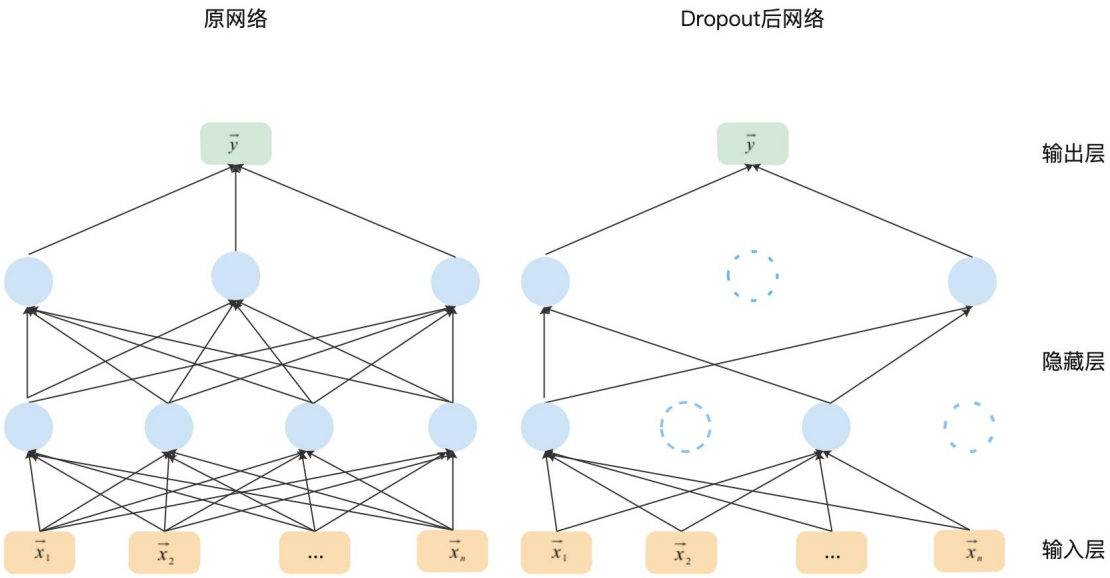

图 8 Dropout 法

Dropout 法的应用能大大增加网络的泛化能力。原因在于，通过随机丢弃神经元，原本定义下的网络产生了许多结构不尽相同，而参数共享的子网络。得到训练后，每个子网络都成为了一个独一无二的、具有预测能力的学习器，尽管他们之间并不独立。子网络将其学习好的参数共享给母网络，并最终利用母网络进行预测。这一想法，本质其实是通过对不同模型的集成以获得一个具有更强泛化能力的模型，和 Bagging 法(Bootstrap aggregating，引导聚集算法)有异曲同工之妙。实际中，对神经网络模型使用 Bagging 法训练的代价是很高的，而 Dropout 法利用同参不同构的方法，以很小的计算代价就实现了类似 Bagging 法的作用，因此成为了深度学习中最受欢迎的正则化方法。

4) 早停法(Early Stopping)

早停法是神经网络训练中常用的防过拟合策略。其原理主要是通过在训练过程中训练集与验证集上的损失函数值进行监督，当出现验证集的 Loss 值进入平台期或开始明显上升时，而训练集 Loss 仍在下降时，即标志着过拟合现象的出现。此时可以提早停止网络训练并保存之前训练过程中验证集 Loss 值最低的模型作为最优模型。

本文的实验中，以 3 个训练周期为单位进行对验证数据集回测。基于对训练过程的观察，选定：

$$|\mathrm{Loss_n} - \frac{\sum_{t=0}^{4} \mathrm{Loss_{n-t}}}{5}| < 0.003 \tag{21}$$

即，第 n 次 Loss 值与第 n-4 至第 n 次 Loss 平均值的绝对差值小于 0.003，作为判断标准对网络应用了早停法。

2.4 股票指数

股票价格指数(Stock Price Index)，即股票指数，是由证券交易所或金融服务机构所制定的、用以反映该股票市场或其某一子市场中整体股票价格水平的指标。相较于单只股票价格，股票指数具有波动性小、市场代表性强的特点，股票指数的预测在宏观上具有更大的经济价值和意义。

股票指数主要通过股票编成方式和计算方式进行区分。常见的股票指数多由某国家里最大规模的证券交易所中最具市场影响力的上市公司编成，采取价格加权或市值加权的方法计算得到。如美股市场中的道琼斯工业平均指数[11](Dow Jones Industrial Average, DJIA)、标准普尔 500[12](Standard & Poor's 500, S&P 500)等。

在我国 A 股市场中，中证 500 指数、沪深 300 指数和上证 50 指数通常被认为是最具有代表性的三只股票指数，也被广大投资者、研究者和政府工作者作为判断我国股票市场运行状况的参考指数。同时，这三只股票这也是本课题的数据集来源。下面依次进行介绍:

① 沪深 300 指数: 从中国沪深两市股票市值大，流动性好的 300 支 A 股，并通过市值加权计算得出，反映了最有影响力的主要股票的整体市场价值。

② 中证 500 指数:由全部 A 股中剔除沪深 300 指数成份股及总市值排名前 300 名的股票后，总市值排名最靠前的 500 只股票通过市值加权的方式计算得到，因此主要反映了中国 A 股市场最好的一批中小市值公司整体股票价格水平。

③ 上证 50 指数:由中国上海证券市场中 50 支市值大且流通性好的 A 股，通过市值

[11]道琼斯工业平均指数(Dow Jones Industrial Average，DJIA)：由美国股票市场中 30 支最有代表性的工商业公司股票构成，通过计算价格的算术平均值得到，其大致可以反映美国整个工商业股票的价格水平，但并不代表其组成公司的市值。

[12]标准普尔 500(Standard & Poor's 500, S&P 500)：由美国纽约证券交易所和纳斯达克(NASDAQ)股票交易所中，几乎是全美最高金额买卖的 500 只股票价格，采用市值加权法计算得到。它是美国仅次于道琼斯工业平均指数第二大指数。但标准普尔 500 总包含的公司数量更多，因此更能够反映整体的市场变化，且市值加权的计算方式将公司规模考虑在内，使得选取的公司在股票市场更具有影响力。

加权的方式计算得到，可以说是沪深 300 成分股中在市值意义上的“精英”，但其在金融行业的集中度极高，多为金融蓝筹股，因此在行业意义上的广度远不及沪深 300。

这三只股票指数的关系可以通过下图 9 进行表示：

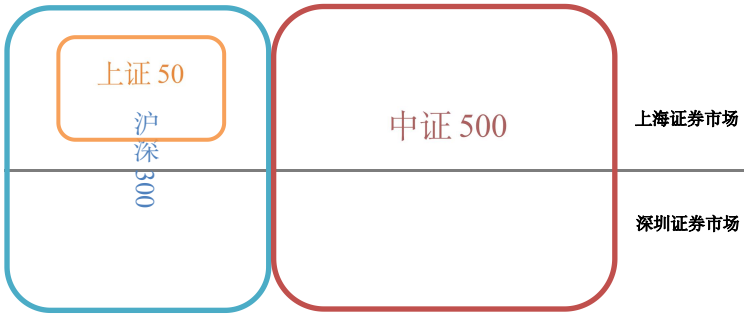

图 9 中证 500 指数、沪深 300 指数和上证 50 指数的关系

2.5 技术分析

技术分析和基本面分析通常是股票投资者常用的择股分析手段。基本面分析，是通过对公司财务状况、行业前景和国家政策等经济学方向上的研究来判断该公司的发展状况和潜在价值，常常被应用于选择股票。技术分析，则是利用股票本身的价格数据，基于一定的数学模型并通过数学公式计算得到的量化指标，很适合在神经网络中作为补充的输入变量，以帮助提升网络的表现；技术指标能即时地反映股票价格和交易状况的变化，因此常常被投资者应用于的股票的择时买卖。

本课题中，同样选取了 9 个技术指标作为基本输入变量(开盘价 OP(t)，收盘价 CP(t)，最高价 HP(t)，最低价 LP(t)，交易量 VO(t))的补充，表 3 列出了各个技术指标及其计算公式：

表 3 技术指标

名称	缩写	计算公式
指数平滑移动平均线	MACD	$MACD(p=CP,t_s=12,t_l=26,t) = EMA(p,t_s,t) - EMA(p,t_l,t)$
相对强弱指数	RSI	$RSI(N,t) = 100 \times \frac{RS(N,t)}{1+RS(N,t)}$
动量	MOM	$MOM(n,t) = CP_t - CP_{t-n}$

续 表 3 技术指标

收盘价变动率百分比	PROCP	$PROCP(p=CP,N=12,t)=\frac{p_t-p_{t-N}}{p_{t-N}}\times 100\%$
威廉指数	WILLR	$WILLR(n,t)=\frac{max\{HP_{t-n},\dots,HP_t\}-CP_t}{max\{HP_{t-n},\dots,HP_t\}-min\{LP_{t-n},\dots,LP_t\}}\times 100\%$
典型价格	TP	$TP(t)=\frac{HP_t+LP_t+CP_t}{3}\times 100\%$
绝对价格震荡	APO	$APO(p,t_s=26,t_f=12,t)=MA(p,t_s,t)-MA(p,t_f,t)$
交易量变动率百分比	VROCP	$PROCP(p=VO,N=12,t)=\frac{p_t-p_{t-N}}{p_{t-N}}\times 100\%$
能量潮	OBV	$OBV(t)=OBV(t-1)+VO,\ if\ CL_t \geq CL_{t-1}$ $OBV(t-1)-VO,\ if\ CL_t < CL_{t-1}$

$$EMA(p,tl,t)=\frac{2}{tl+1}\times p_t+(1-\frac{2}{tl+1})\times EMA(p,tl,t-1)$$

$$RS(N,t)=\frac{\text{t 前 N 日内收盘涨幅之和}}{\text{t 前 N 日内收盘跌幅之和的绝对值}}$$

$$MA(p,t)=\frac{\sum_{i=1}^{t}p_i}{t}$$

2.6 主成分分析(PCA)

主成分分析是可以通过简单的线性变换，将一组可能线性相关的特征向量投影成为一组线性不相关、依照方差贡献值排列的向量，由此同时实现对特征向量集去相关性和降维功能的分析方法。具体计算流程如下：

X 为含有 n 个数据点，p 个特征的数据矩阵，即 $X\in\mathbb{R}^{n\times p}$; X(i, j)记矩阵 X 第 i 行，第 j 列元素。PCA 流程如下：

① 数据标准化：为避免各特征值间的数量规模不同所带来的影响，首先对原始数据进行标准化变换得到标准化数据矩阵 Z：

$$Z_{i,j}=\frac{x_{ij}-\bar{x}_j}{s_j} \tag{22}$$

$$where:$$

$$\bar{x}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}$$

$$s_j = \sqrt{\frac{\sum_{i=1}^{n} \left(x_{ij} - \bar{x}_j\right)^2}{n-1}}$$

按照特征值进行降序排列，得到对应的特征值和特征向量集$(\lambda_1, w_1), (\lambda_2, w_2), \ldots, (\lambda_p, w_p)$。

② 计算主成分值及其对应的方差解释值

此时，称$Y_k = w_k^T Z$为第 k 个主成分，k = 1,2,...,p。

同时可以证明:$Var(Y_k) = \lambda_k$且$Corr(Y_k, Y_g) = 0$ if $k \neq g$,即所得到的 p 个随方差贡献值λ_k递减的、相互独立的主成分集。

③ 确定选取主成分个数 K

主成分个数 K 的选取准则有很多，选取相对也比较主观。本研究中，采取保留 70%方差解释率的准则对 K 进行选取，即：

④ 计算 Z 的相关系数矩阵$S_{p\times p}$:

$$S = \frac{Z^T Z}{n-1} \tag{23}$$

此时，S_{ij}即代表 Z 第 i 列与第 j 列的相关系数。

⑤ 计算矩阵 S 的特征值与特征向量：

当 Z 满秩时，可通过求解相关系数矩阵 S 的特征方程$|S - \lambda I_p| = 0$得到 p 个特征值和其对应的特征向量；或者亦可以通过对 Z 进行奇异值分解的方法计算得到主成分。

许多研究表明，具有强相关性的特征集在神经网络的训练过程中会带来许多不良影响，如：易落入局部最优、降低模型的泛化能力等；同时，冗赘的特征集也会大大增加网络训练的时间和计算力代价。因此，在神经网络的训练中，对输入集进行相应的的变量筛选与变换是具有一定价值的。

在本次研究中，所选取的 9 个技术指标之间存在一些明显和潜在的相关性，如同参数的 MOM 与 PROCP 之间的缩放关系，故选择 PCA 进行降维，以期对网络的预测能力、泛

化能力和计算效率进行进一步提高。

第三章 实验结果与分析

3.1 数据集构造与预处理

3.1.1 数据集构造与描述

1) 基本数据集

本实验使用了中国 A 股市场中最具有代表性的 3 支股票指数：沪深 300(399300)，上证 50(000016)，中证 500(000905)作为数据来源。

每支股票指数对应数据集均由该股指在 2018 年 1 月至 2018 年 6 月期间的分钟频数据构成，包含约 29000 个数据点。

基本数据特征选取了：开盘价(OpenPrice, OP),收盘价(ClosePrice, CP)，最高价(HighestPrice, HP)，最低价(LowestPrice, LP) 和交易量(TradeVolume, VO)共五个变量作为基本数据特征集。以沪深 300(399300)为例，基本数据集表头如下：

表 4 沪深 300 的基本数据表

OpenPrice	ClosePrice	HighestPrice	LowestPrice	TradeVolume
4045.21	4045.21	4045.21	4045.21	62962500
4047.88	4054.69	4054.69	4047.88	167443900
4054.58	4053.13	4054.64	4053.13	112311800
4052.21	4052.83	4052.83	4051.39	98417600
4052.53	4054.28	4054.39	4052.53	108536500

2) 含技术指标的数据集

含技术指标的数据集是在基本数据集的特征上，添加了指数平滑移动平均线(MACD)、动量(MOM)、威廉指数(WILLR)、典型价格(TP)等九个技术指标后所构成的数据集，2.5 节中的表 3 具体列出了各个指标及计算方法。此时的数据集共含有 14 个输入特征。

3) 对非基本价格指标进行 PCA 降维后的数据集

鉴于技术指标之间存在明显相关性，本文利用 PCA 技术对数据集 2 中的非基本价格特

征，即 9 个技术指标与交易量，共计 10 个特征进行了降维。依照保留 70%方差解释率、Elbow 等主成分保留个数的规则，最终选择保留 3 个主成分。此时，数据集共含有 7 个输入特征。

3.1.2 价格数据平稳化

1) 数据平稳化

对于时间序列预测，许多模型都是基于数据平稳性的假设。这里的平稳性一般是指弱平稳，即弱平稳过程的一阶矩和二阶矩不随时间变化。定义如下：

弱平稳：时间随机过程 X(t)被称为平稳当：

① $E\{x(t)\} = m_x(t) = m_x(t+\tau), \tau \in \mathbb{R}$

② $E\{x(t_1)x(t_2)\} = R_x(t_1, t_2) = R_x(t_1+\tau, t_2+\tau) = R_x(t_1 - t_2, 0), \tau \in \mathbb{R}$

常见的时间序列平稳性探测方法主要包括观察法和单位根检验法。观察法通过绘制并观察自相关系数(ACF)图和偏相关系数(PACF)图来发现趋势或周期性的存在与否，相比于单位根检验法而言是一种更具有主观性的方法。单位根检验法则是一种统计方法。当序列存在滞后相关，即不平稳时，其滞后算子多项式特征方程则存在单位根；反之，则不存在单位根。ADF 检验(Augmented Dickey-Fuller test)利用构造 t 统计量进行假设检验的方法对单位根进行检测；若拒绝“存在单位根”的原假设，则认为该序列是平稳的。由于平稳性并非本文的研究重点，因此不再进一步讨论，而仅将 ADF 检验作为平稳性检验工具使用。

从图 10 中绘制的价格曲线可见，原始序列存在明显的时间相关性。在实验中，当直接利用原始数据进行价格预测时极易出现网络预测滞后的现象，而这样的永远“迟到”的预测在实际应用中上是几乎没有意义的。因此，进一步对价格特征分别进行了一阶差分，平稳后的价格序列见图 11。

对此时价格特征对应的时间序列进行 ADF 检验后发现，统计量均在 95%置信区间意义上拒绝原假设，即一阶差分后的数据可以被认为是平稳时间序列。从实际意义的角度来看，对原分钟频价格的预测被转化为了平稳的、一阶差分价格序列的预测，这时的序列是对价格变化大小的直接反映；对于网络的学习而言，此时的输入数据集去除了冗余的价格数量本身而更直接地反映了变化，使得网络不得不从变化中寻找变化的动力与关系，因此更有利于网络对价格变化隐藏推力的挖掘与提取。

图 10 中证 500 指数、沪深 300 指数和上证 50 指数的收盘价价格曲线

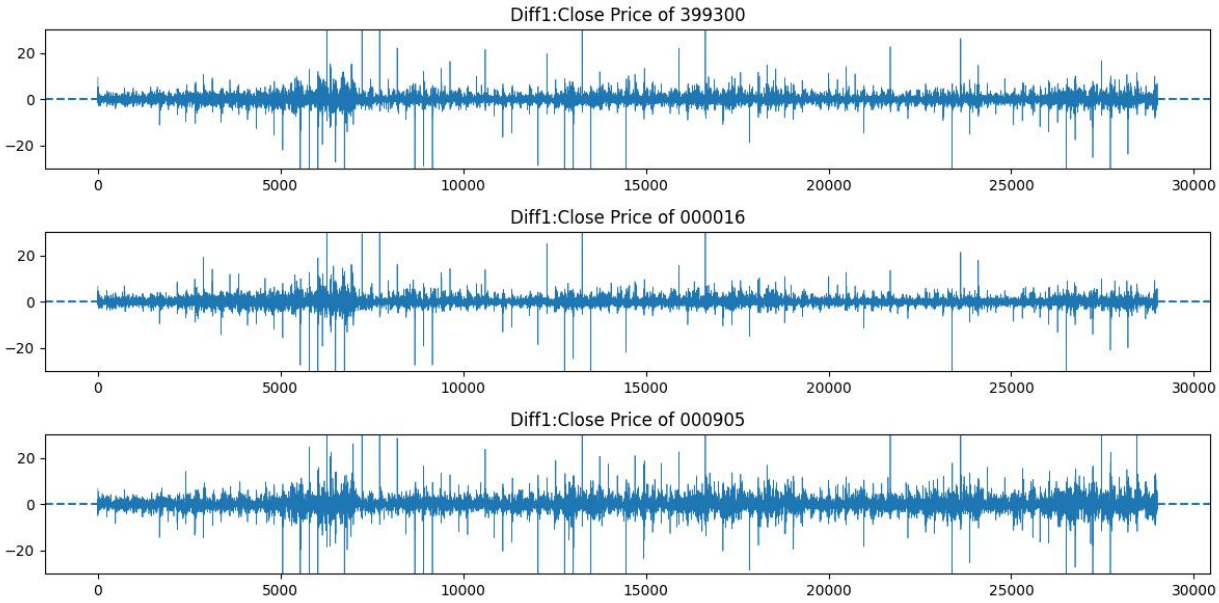

图 11 一阶差分后的中证 500 指数、沪深 300 指数和上证 50 指数的收盘价价格曲线

2) 数据平稳化后的数据集

以沪深 300(399300)为例，对价格数据进行一阶差分操作后，得到平稳化后的基本数据集(特征集 1)、含技术指标的数据集(特征集 2)表头、PCA 降维后的数据集表头(特征集 3)如下：

表 5 沪深 300 平稳化后的基本数据集

diff_open	diff1_close	diff1_high	diff1_low	TradeVolume
-1.77	-0.38	-1.77	-0.88	117791000
-0.22	-2.25	-0.22	-1.75	94868900
-2.55	-1.21	-2.55	-1.21	96502200
-1.15	-1.17	-1.15	-1.45	104216700
-1.29	-0.49	-1.29	-0.4	79667200

表 6 沪深 300 含技术指标的数据集

diff_open	diff1_close	diff1_high	diff1_low	TradeVolume	MACD	RSI	...	OBV
-1.77	-0.38	-1.77	-0.88	117791000	3.047652174	66.62683522	...	16360400
-0.22	-2.25	-0.22	-1.75	94868900	2.472870071	63.98614565	...	-78508500
-2.55	-1.21	-2.55	-1.21	96502200	1.89783714	62.60573472	...	-175010700
-1.15	-1.17	-1.15	-1.45	104216700	1.332351657	61.283258	...	-279227400
-1.29	-0.49	-1.29	-0.4	79667200	0.835035662	60.72749103	...	-358894600

表 7 沪深 300 PCA 降维后的数据集

diff1_open	diff1_close	diff1_high	diff1_low	PC1	PC2	PC3
-1.77	-0.38	-1.77	-0.88	-1.820428037	1.96171484	0.57931839
-0.22	-2.25	-0.22	-1.75	-1.096595524	1.694003185	-0.01990392
-2.55	-1.21	-2.55	-1.21	-0.439728619	1.735168217	-0.007778411
-1.15	-1.17	-1.15	-1.45	0.061792021	1.846795098	0.175148857
-1.29	-0.49	-1.29	-0.4	0.211719684	1.550550757	-0.411712627

3.2 网络结构与参数设置

在网络本身可学习的参数之外，网络结构和优化参数这一类超参数(Hyperparameter)

的选取对网络的预测效果具有决定性的作用。超参数的优化本质上是一种组合优化问题，很难找到非常有效的优化算法。一般地，会结合过往经验，通过网格搜索法[13]、贝叶斯搜索法[14]、随机搜索法[15]等方法进行优化。

本文中，对以上超参数(隐藏层单元数、参数初始化方法、激活函数、Dropout 保留率等)的优化是通过阅读文献划定一个基本的取值范围，利用网格搜索法进行确定的。由于这并非本文讨论的重点，因此不再列出优化的计算过程，将直接给出各个超参数的设定。

3.2.1 网络结构与优化参数

网络是含 3 层 LSTM 隐藏层、1 层全连接层的深层 LSTM 神经网络，其结构如图 10 所示。

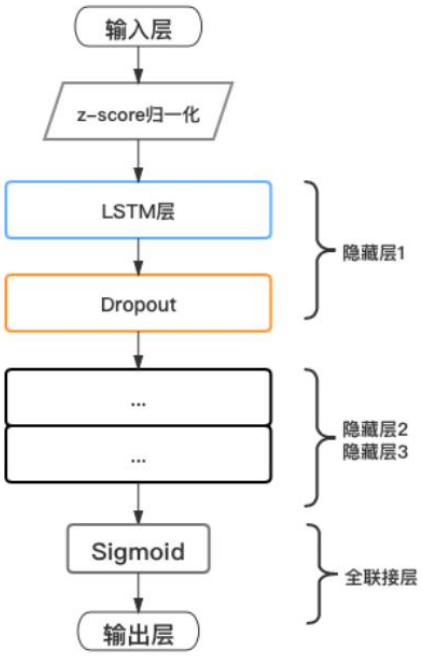

图 12 中证 500 指数、沪深 300 指数和上证 50 指数的关系

[13]网格搜索(Grid Search)：一种通过尝试所有超参数的组合来寻址合适一组超参数配置的方法。

[14]贝叶斯优化(Bayesian optimization)：是一种自适应的超参数优化方法，根据当前已经试验的超参数组合，来预测下一个可能带来最大收益的组合。

[15]随机搜索(Random Search) ：是一种在实践中比较有效的改进方法是对超参数进行随机组合，然后选取一个性能最好的配置，这就是随机搜索在实践中更容易实现，一般会比网格搜索更加有效。

其中，设定 LSTM 隐藏层单元数均为 32，权重参数初始化方法采取高斯分布的 Xavier 初始化，激活函数选取 LReLU 函数。全连接层的输出维度为 1，在分类问题中，激活函数选取 Sigmoid 函数。各隐藏层间含 Dropout 层，其中 Dropout 的保留率视训练过程中过拟合程度进行调整；值得注意的是，在进行验证集和测试集上的计算时应不再使用 Dropout，即将保留率设置为 1。优化算法采用 Adam 算法，初始学习率设定为 0.001，其他参数取默认初始值。

本文中，选取 Tensorflow 框架中的 loss_layers.roc_auc_loss 函数，即 ROCAUC 的替代损失函数[16]作为网络的损失函数。在进行的前期实验与部分参考文献中，以交叉熵为损失函数的 LSTM 神经网络常常会出现网络对数据标签不平衡极度敏感，预测结果永远为标签中出现最多那一类，无法跳出局部最优点的情形。此时网络的学习几乎是无效。而选取 ROCAUC，从预测意义的角度来看，它在股票预测问题中是对预测效果更加全面与客观的反映；从优化的角度出发，ROCAUC 替代损失函数的性质很好的解决了标签不平衡所带来的问题，网络不易落入鞍点，从而实现了更好、更有效的预测效果。因此，本文选取了并不常用的 ROCAUC 作为优化目标。

LSTM 网络中，输入序列的长度为 30。在本文中，即利用连续的前 30 分钟的观测值预测接下来 t 分钟内的涨跌趋势。网络优化采用批量优化法，批大小(Batch size)经优化后选取的值为 128, 即每次优化都是基于 128 个长度为 30 的输入序列进行梯度计算与迭代的，这 128 个样本是随机选取的。

网络中添加了早停机制以对网络在训练集和验证集上的表现进行监督, 具体方法见 2.3 节。

超参数的设置与解释如表 8 所示：

表 8 网络超参数于参数设置

参数名	参数意义	默认参数值
Hidden_size	LSTM 中隐藏节点的个数	32
Batch_size	Batch size	128
Num_layers	LSTM 网络层数	3
Timesteps_X	输入序列的时间长度	30

续 表 8 网络超参数于参数设置

Timesteps_Y	预测序列的时间长度	1/5/15
Adam:learning_rate	Adam 优化器的初始学习率	0.001
Dropout:state_keep_prob	Dropout 方法中保留神经元的比例	0.5

3.2.2 数据处理

1) z-scores 归一化

数据集中各个特征的量纲不尽相同，特征的取值范围与分布也差异极大。在依赖于距离计算的许多机器学习模型中，归一化是保证模型有效学习的必不可少的数据预处理步骤之一。对于神经网络而言，尽管可以通过学习和调整权重参数将输入特征映射至适合的取值范围内，但这会给网络优化造成不必要的负担，大大降低网络的学习效率。因此，在神经网络中对数据进行归一化是一个对网络学习效率和质量的提升都有非常有益的步骤。

z-scores 归一化，又称标准归一化，是最常用的归一化方法之一。其基本思想是将原数据集调整为均值为 0，方差为 1 的归一化数据集。对于含有 N 个样本的特征 $x_n, n = 1,2,,\ldots,N$，z-scores 归一化计算方法如下：

$$\hat{x}_n = \frac{x_n - \mu}{\sigma} \tag{24}$$

$$where: \mu = \frac{\sum_{n=1}^{N} x_n}{N}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n} \left(x_n - \mu\right)^2}{N}$$

本文中，z-scores 归一化并非是基于整个数据集进行，而是采取对每一个输入特征序列中的特征进行归一化的方式。

2) 分类标签

本实验的预测目标是对在 Timesteps_Y 时间长度内的股票价格趋势(涨或跌)进行预测，是一个二分类问题。因此，需要对原始价格值进行二值化处理。处理方法如下：

$$Y\left(t, Timesteps_Y\right) = \begin{cases} 1, & if\ CP_t + CP_{Timesteps_Y} > 0 \\ 0, & if\ CP_t + CP_{Timesteps_Y} \leq 0 \end{cases} \tag{25}$$

即，若 t + Timesteps_Y 时的收盘价高于 t 时刻的收盘价，则记标签Y_t为 1，否则记为 0，此时的输入则为 t -Timesteps_X 至 t 区间内的特征序列。

网络的输出是取值范围在(0,1)之间的连续值，因此也需要进行离散化。处理方法如下：

$$\widehat{Y}\left(t, Timesteps_Y\right) = \begin{cases} 1, & if\ output > 0.5 \\ 0, & if\ output \leq 0.5 \end{cases} \tag{26}$$

3) 滑动窗口(Sliding Window)

确定了网络的输入是长度为 30 的序列后，需要对原数据集进行分割处理形成对应的输入集和标签值。在这里，如果简单地对数据集按照序列长度进行不重叠的切分，实际上会大大降低对数据的利用率。因此，本文采用滑动窗口的方式对数据集进行划分。示意图见图 13：

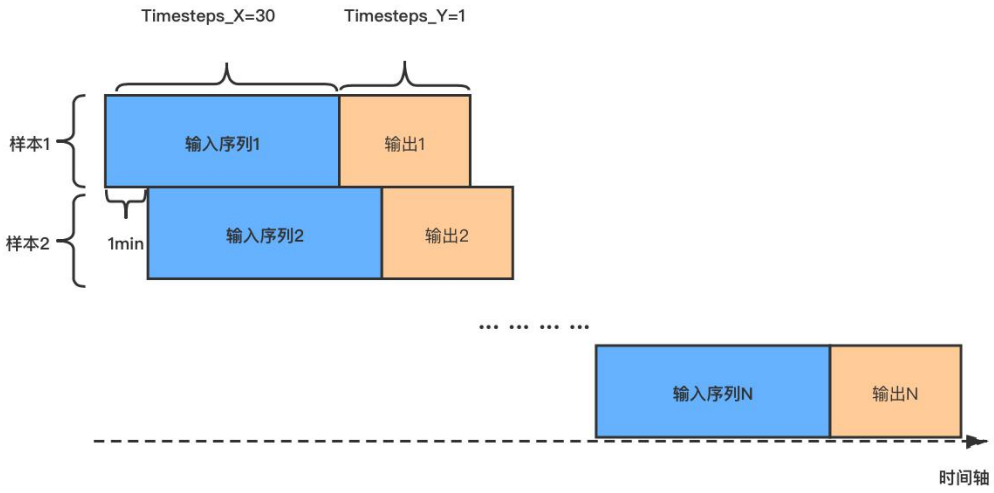

图 13 滑动窗口示意图

4) 数据集划分

每次实验中，均按照 0.64:0.16:0.2 的比例将整个数据集随机划分得到训练集(Training Set)，验证集(Validation Set)，测试集(Test Set)。训练集用于对网络进行训练，验证集用于在训练过程中进行监督并进行超参数的调整和优化，测试集则用于网络回测。

3.3 实验过程与结果

3.3.1 基于不同特征集的时的网络表现

3.1.2 中，原始数据集经过不同的特征选择和处理方式得到了三个具有不同结构的数据集，分别为：对价格数据进行平稳化后的基本数据集(特征集 1)、含技术指标的数据集(特征集 2)表头、PCA 降维后的数据集表头(特征集 3)。这一部分的目的在于选取预测能力强、计算代价小的数据集作为后续实验的数据来源。

为了选取最适合网络预测的数据集，本文利用沪深 300(399300)在 2018 年 1 月至 2018 年 6 月期间的数据，即完整数据集的前 1/3 部分，生成了对应的三个数据集。以预测未来 1 分钟内股指价格的涨跌趋势为目标，利用同样的训练集、验证集和测试集，对上述网络进行了训练。在 50 个周期的训练长度内，并通过添加早停机制对网络进行停止训练的控制。实验结果如表 9 所示：

表 9 预测未来 1 分钟内股指价格的涨跌

特征集名	特征集 1		特征集 2		特征集 3	
评价指标	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
Training Set	0.7471	0.6903	0.7630	0.7141	0.7621	0.7073
Test Set	0.7373	0.6809	0.7493	0.6970	0.7506	0.6923

对结果进行分析发现：

1) 特征集 1 与特征集 2 对比

在训练集和测试集上，特征集 1 的 AUC 与 Accuracy 均低于特征集 2 约 2%，可见本文选取的技术指标的添加能在一定程度上帮助提高网络的表现。

2) 特征集 2 与特征集 3 对比

无论是在训练集还是测试集上，特征集 3 的 AUC 与 Accuracy 均与特征集 2 的表现几乎持平，差值在 0.5%以内。可以说，针对特征集 2 中的非数据特征进行 PCA 降维的处理，对网络的表现几乎不会造成负面影响；此外，从计算代价的角度看，降维后的特征集 3 的特征数仅为特征集 2 的一半，因此带来的网络参数量的减少也使得计算代价有明显的降低。

3) 过拟合与欠拟合

网络在测试集上的 AUC、Accuracy 均低于训练集，且差值在 2%以内；通过对图 12 的观察可以发现，网络训练停止于其在测试集上的表现进入稳定平台后的阶段。因此，训练未出现明显的过拟合或欠拟合现象。

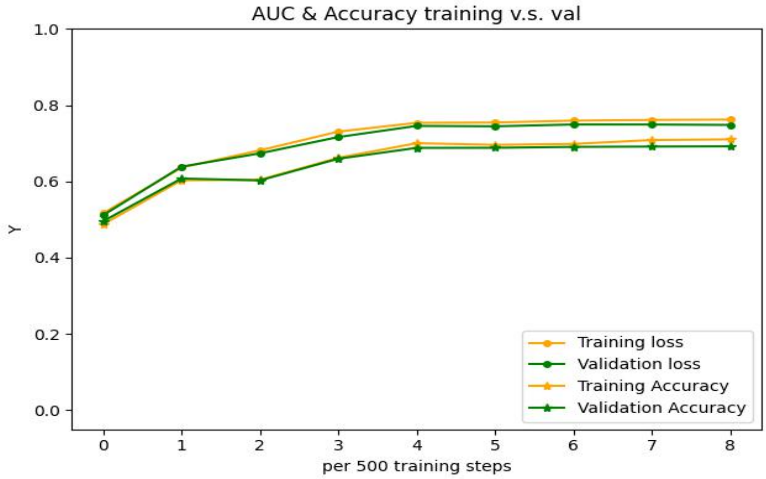

图 14 训练集与验证集在训练过程中的 loss 与 accuracy 线

与“盲猜”相较，网络仅通过对基本价格特征与成交量数据的学习就实现了对 AUC 与 Accuracy 的 24%和 18%的提升量，而特征指标的添加仅带来了 2%左右的提升量。因此，从预测效果的角度来看，特征集中技术指标的添加是对预测效果具有稳定的提升作用的，然而基本价格特征与成交量在神经网络预测中仍然是起到主要作用的。

通过对技术指标进行 PCA 降维后，这样的降维方法在明显降低网络复杂度、缩短训练时间的同时也不会对网络的预测准确率造成损失。从这一角度讲，这是一种能够为本神经网络性能带来进一步提升的方法。

综上，特征集 3，即对特征集 2 中非基本价格特征进行 PCA 降维后的数据集，是在综合预测能力与计算代价后最适合网络训练的数据集。因此，后续实验将主要利用特征集 3 对网络预测性能、泛化能力进行进一步的探究与验证。

3.3.2 网络在其他股票指数上的表现

上一节通过搭建的堆叠 LSTM 神经网络，已基于三个不同的特征集对沪深 300(399300)进行了 1 分钟价格趋势预测的实证与分析，筛选出了 PCA 降维后的数据集是预测性能最好的网络。本节，将对股票池中的其他股指，上证 50(000016)与中证 500(000905)，取 2018 年 1 月至 2018 年 6 月期间的分钟频数据进行网络训练，结果如表 10 所示：

表 10 三只股指对未来 1 分钟内涨跌预测表现对比

股票指数名	沪深 300(399300)		上证 50(000016)		中证 500(000905)	
评价指标	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
Training Set	0.7621	0.7073	0.6708	0.6468	0.7363	0.6866
Test Set	0.7506	0.6923	0.6297	0.5988	0.7277	0.6759

由上表数据可见，神经网络在选取的三只股票指数中的平均预测准确率为 65.57%，平均 AUC 为 0.7027，预测效果不错，但网络在各只股票上的表现差异较大。

对于沪深 300 与中证 500 这两支构成股数量较大、数量级相近的股指，网络的表现相近，且明显较优，平均预测准确率达到 68.41%，平均 AUC 达到 0.7391；而在上证 50 中的表现则远不及前者，预测准确率与 AUC 分别为 59.88%，0.6297。

综合以上分析可见，在 1 分钟价格趋势的预测中，网络在构成股数量大、广度大、种类丰富的股票指数走势预测中表现更优。而上证 50 差强人意的表现，其原因或许与涉股数量小、构成股在金融行业具有高度的密集性有关，具体原因有待在未来的研究中进一步分析。

3.3.3 网络在更长期预测中的表现

为了对网络的预测能力进行进一步的探索，本节在沿用前文中神经网络结构、超参数设定与数据集的基础上，对网络在未来 5 分钟、15 分钟收盘价趋势预测进行了实验，具体结果见表 11：

表 11 三只股指对未来 5 分钟内涨跌预测表现对比

股票指数名	沪深 300(399300)		上证 50(000016)		中证 500(000905)	
评价指标	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
Training Set	0.7537	0.7080	0.7335	0.6928	0.7308	0.6836
Test Set	0.6972	0.6536	0.6674	0.6332	0.6749	0.6339

上表反映了利用同样的数据集与网络对股票指数收盘价在未来 5 分钟内涨跌趋势的预

测。在测试集上的平均预测准确率为 64.02%，平均 AUC 为 0.6798，表现良好。

横向比较发现，各只股指的 AUC 与精确度方差很小，AUC 取值在 0.66-0.70 之间，进度取值在 63.3%-65.4%之间，说明网络在此时间长度的趋势预测中稳定性更强。其中，在沪深 300 数据集上的表现是最优的。

纵向来看，与 1 分钟预测结果相比，5 分钟整体预测效果的评价指标值明显低于前者。值得注意的是，网络在上证 50 数据集中的预测效果却与整体趋势恰恰相反，指标值不减反增。从这个角度来看，这一现象进一步反映了在本研究方法下，沪深 300 与中证 500 具有更加相似的性质，而上证 50 的表现则明显区别于另外两者。

表 12 三只股指对未来 15 分钟内涨跌预测表现对比

股票指数名	沪深 300(399300)		上证 50(000016)		中证 500(000905)	
评价指标	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
Training Set	0.8493	0.8115	0.8441	0.8038	0.8569	0.8180
Test Set	0.7948	0.7497	0.7867	0.7480	0.8106	0.7530

表 12 显示, 在 15 分钟趋势预测中, 网络分类结果的 AUC 与准确率达到了平均 0.7974, 75.02%的水平。此评价标准之下，15 分钟时间长度内的预测是在三种不同时间长度预测中效果最好的。

另一方面，在数据集发生变化时，这一时间长度上的预测是目前稳定性最强的预测。三只股指在测试集上预测的 AUC 取值范围为 0.79-0.81，准确率为 75.8%-75.3%，波动区间明显减小，方差更小。

综上，从 AUC 与准确率的角度进行对比可见，15 分钟的趋势预测具有预测效果最好、稳定性最强的性质。

3.3.4 高频交易收益率分析

基于 3.3.3 中确定的最终网络参数，对测试集中近 6000 个样本点进行了交易模拟。交易通过 LSTM 神经网络对沪深 300、上证 50、中证 500 三只股指在未来 1、5、15 分钟内的收盘价走势进行预测。若预测为上涨，则买入一股；若预测为下跌，则不买入或卖出以保

持不再持有该股的状态。绘制得到累积收益率曲线如图 13 所示。其中，t 时刻交易操作的策略收益率按照：$收益率_t = I(在\ t\ 时刻买入该股) \times \frac{收盘价_{t+Timestep\_y}}{收盘价_t}$ 的简单收益率计算方式，真实收益率计算方式则为在测试时间段内始终持股的简单收益率。

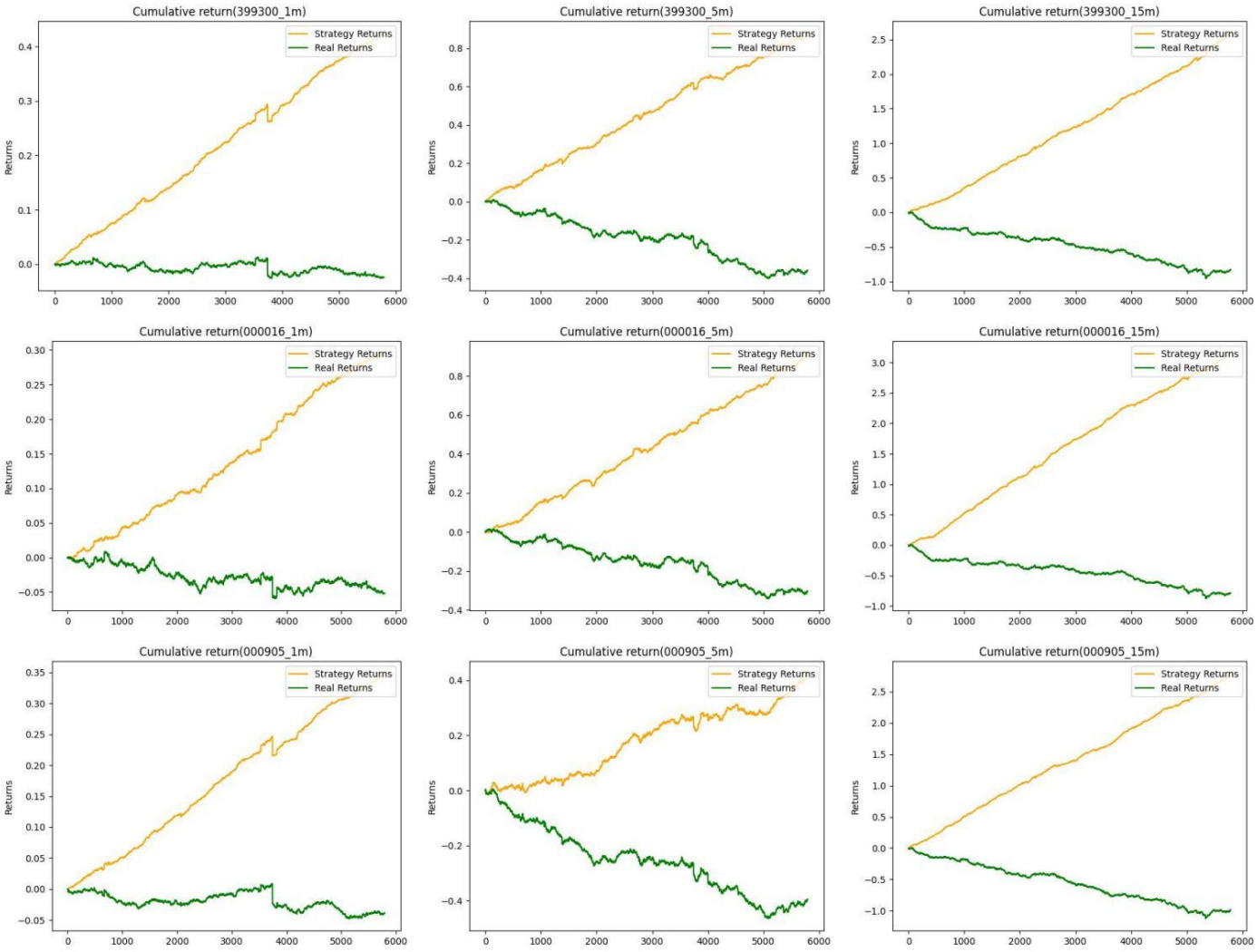

图 15 三只股指在 1 分钟、5 分钟、15 分钟走势预测中的交易策略回报率与真实回报率

从图中明显可见，测试时间段内，交易策略回报率线与真实回报率线均呈明显的左箭头型。无策略的股票收益率在大多时间内保持负值、持续下跌的状况，而结合神经网络预测后的策略交易回报率始终维持正值，且持续上涨。这一现象在不同股指、不同预测周期中都是普遍存在的。因此可以说明，这一策略模型是明显有效的。

为进一步量化交易回报率以获得更为精准的效果评估，基于上述结果，本文进一步对

不同股指在不同预测时间长度内的策略与非策略交易进行了日收益率的计算，结果见下表13:

表 13 收益率表

股票指数名	沪深 300(399300)		上证 50(000016)		中证 500(000905)	
	收益率(天)	基线(天)	收益率(天)	基线(天)	收益率(天)	基线(天)
1 分钟	1.757%	-0.102%	1.222%	-0.214%	1.441%	-0.160%
5 分钟	0.726%	-0.297%	0.758%	-0.250%	0.624%	-0.328%
15 分钟	0.821%	-0.228%	0.869%	-0.218%	0.770%	-0.271%

计算结果显示，进行最高频的 1 分钟频交易操作时，策略日收益率最高，均在 1.2%以上；而 5 分钟、15 分钟频的交易收益率相对较低，取值区间在 0.62%至 0.82%之间。在无策略交易收益率普遍在-0.3%至-0.1%水平的情况下, 本模型策略仍能获得 1%左右的理想超额收益，这说明模型具有规避风险与损失、获取创造超额收益的潜力。

值得注意的是，上表的计算方式是在设定投资股数在各个时间点保持相同，即在已进行预测的时间区间内不再做决策的基础上进行计算的，这时的交易频率与预测时间长度是呈反比的。此外，本文中收益率的计算是并未将实际交易过程中的交易手续费纳入考虑范围。实操中，这将是一笔可观的费用，因此如何使得本文研究成果在实际交易中获取超额利润仍需要后续更加深入的研究。

第四章 总结与展望

4.1 工作总结

本文以基于 LSTM 神经网络与高频股票价格趋势预测为研究目标，进行理论学习、基于 Python 的网络搭建与数据实证，最终得到了具有较强的趋势预测能力、泛化能力与获得超额收益的交易模型。研究主要从我国 A 股市场中的沪深 300、中证 500、上证 50 三只重要股票指数的高频数据入手，通过对堆叠式 LSTM 神经网络进行搭建与超参数的优化，验证了网络在不同的预测时间长度、不同评价标准下的表现。

一方面，在超参数优化中，本研究选用了不常用的 ROC 曲线下面积(ROCAUC)的损失替代函数作为 Loss 函数进行目标优化，解决了使用交叉熵损失函数时 LSTM 网络学习无效的问题；从特征选取的角度，采用了高频技术指标与 PCA 技术相结合的方式对基本特征集进行了扩充，构建出在网络中表现更佳的新特征集，验证了在高频的趋势预测中技术分析与 PCA 降维技术同样是有助于提高预测准确率。

另一方面，在预测期限上，通过对未来 1 分钟、5 分钟、15 分钟 3 个不同时间跨度内收盘价的涨跌进行预测和结合交易策略进行的收益率计算，发现模型在不同时间长度中均具有较好的预测性能，并能获得超过市场收益率的超额收益,验证了所搭建网络在高频股指价格走势预测中较强的趋势预测能力、泛化能力与实际应用能力。

4.2 工作展望

针对在本研究的各项工作与实际验证中发现的问题，提出以下几点展望：

首先，本文在 3.3.4 节中的高频交易收益率分析并未考虑实际操作中的交易手续费用等，而费用通常在交易额的 0.1%以上，这使得高频交易的成本大大提升。为了抵消这一费用，可以通过对延长走势预测的时间长度或对定量地对股票价格进行预测以控制成本的方式实现。但由于并非本文的主要研究目标且内容较多，可作为后续研究方向，结合优化的交易策略，以获得具有更高收益率的模型和更真实客观的收益率估计。

此外，本文的技术指标仅选取了属于 3 个常用大类的、共九个技术指标。实际上股票技术指标数量众多、类型丰富，计算方式不尽相同，因此其中涉及的相关超参数与参数也有很多。本文参数的设置多选取日线分析中常用的参数值，且在 3.3.1 节中被证明对网络表现具有提升作用。因此在后续的研究中，可以进一步对技术指标的参数进行优化与调整。

最后，本文采用的模型是有监督学习中的深度学习模型，并未探索或考虑环境与时间因素对模型的影响。鉴于股票市场环境瞬息万变的性质，构建与环境变化相适应的动态模型，或许能够获得更高的收益期望。在后续的研究中，可以考虑将强化学习(Reinforcement learning)与量化交易相结合，以期构建可与环境进行交互学习的更优的模型。

参考文献

[1] C. H. Wu, C. C. Lu, Y. F. Ma, R. S. Lu.A New Forecasting Framework for Bitcoin Price with LSTM[C].2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 168-175, 2018

[2] M, Hiransha, Gopalakrishnan E.A., Vijay Krishna Menon, and Soman K.P. NSE Stock Market Prediction Using Deep-Learning Models[C].Procedia Computer Science 132 (January): 1351–62. doi:10.1016/j.procs.2018.05.050.

[3] Khaled A. Althelaya, El-Sayed M. El-Alfy, Salahadin Mohammed.Evaluation of Bidirectional LSTM for Short- and Long-Term Stock Market Prediction[C]. 9th International Conference on Information and Communication Systems (ICICS), 2018 IEEE, pp. 151-15

[4] Heaton J B，Poison N G，Witte J H．Deep Learning in Finance[J]．arXiv preprint arXiv：1602．0656i，2016．

[5] Ferdiansyah, F.et al. A LSTM-Method for Bitcoin Price Prediction: A Case Study Yahoo Finance Stock Market[C]. 2019 International Conference on Electrical Engineering and Computer Science (ICECOS), Electrical Engineering and Computer Science (ICECOS), 2019 International Conference on, pp. 206–210.

[6] Gustavsson A, Magnuson A, Blomberg B, et al. On the difficulty of training Recurrent Neural Networks[J]. Computer Science, 2013.

[7] 任君,王建华,王传美,王建祥.基于正则化 lstm 模型的股票指数预测[J].计算机应用与硬件.2018.04.008.

[8] 杨青,王晨蔚.2019.基于深度学习 LSTM 神经网络的全球股票指数预测研究[J].统计研究,2019.03.006.

[9] Goodfellow I., Bengio Y., Courville A. (2016). Deep learning[M]. The MIT Press.

[10] Di Persio L, Honchar. Recurrent Neural Networks Approach to the Financial Forecast of Google Assets[J]. International Journal of Mathematics and Computers in Simulation, 2017(1)：7-13．

[11] He K, Zhang X, Ren S, et al. Delving Deep into Rectifiers: Surpassing HumanLevel Performance on ImagenetClassification[C].Proceedings of the IEEE International Conference on Computer Vision. 2015: 1026-1034.

[12] Li, Xinyi, Yinchuan Li, Hongyang Yang, Liuqing Yang, and Xiao-Yang Liu. DP-LSTM: Differential Privacy-Inspired LSTM for Stock Prediction Using Financial News[R].2019.

[13] Zhixi Li, Vincent Tam.Combining the real-time wavelet denoising and long-short-term-memory neural network for predicting stock indexes[C].2017 IEEE Symposium Series on Computational Intelligence, SSCI, IEEE, 2017.

[14] Gao, T. and Chai, Y. Improving Stock Closing Price Prediction Using Recurrent Neural Network and Technical Indicators[J].NEURAL COMPUTATION, 30(10), pp. 2833–2854. doi: 10.1162/neco_a_01124.

[15] 邱锡鹏.神经网络与深度学习[M].https://nndl.github.io/nndl-book.pdf.2020

[16] Eban E. ,Schain, M., Mackey A, et al. Scalable Learning of Non-Decomposable Objectives[R].2016.

[17] 俞福福.基于神经网络的股票预测[D].哈尔滨：哈尔滨工业大学，2016.

[18] 刘忠仁.高频交易下股票技术分析的策略优化--基于 Adaboost 神经网络方法[D]. 上海师范大学，2019.

[19] Sezer OB, Gudelek MU, Ozbayoglu AM. Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review: 2005-2019[J].2019.

[20] Hochreiter s，Sehmidhuber J．Long Short-term Memory[J]．Neural Computation，1997(8)：1735-1780．

[21] Lin Chen, Zhilin Qiao, Minggang Wang, Chao Wang, Ruijin Du, Harry Eugene Stanley, Which artificial intelligence algorithm better predicts the Chinese stock market[C] .IEEE Access 6 (2018) 48625–48633.

[22] Lai CY, Chen R-C, Caraka RE. Prediction Stock Price Based on Different Index Factors Using LSTM[C]. 2019 International Conference on Machine Learning and Cybernetics (ICMLC), Machine Learning and Cybernetics (ICMLC), 2019 International Conference On, July 1, 2019, 1–6. doi:10.1109/ICMLC48188.2019.8949162.

附录 A 网络输出

399300 预测 1 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.7529	0.6913		
Test	0.7535	0.688	0.017569196	-0.00101614
	Up_hat	Down_hat		
Up	0.338225138	0.137948895		
Down	0.174033149	0.349792818		

399300 预测 5 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.74	0.6949		
Test	0.6828	0.6412	0.00726086	-0.002965196
	Up_hat	Down_hat		
Up	0.300120877	0.177516836		
Down	0.181315835	0.341046451		

399300 预测 15 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.8493	0.8115		
Test	0.7948	0.7497	0.008212543	-0.002282173
	Up_hat	Down_hat		
Up	0.343409915	0.136465711		
Down	0.113836587	0.406287787		

000016 预测 1 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.6708	0.6468		
Test	0.6297	0.5988	0.012223069	-0.002144176
	Up_hat	Down_hat		
Up	0.247928177	0.219095304		
Down	0.18214779	0.350828729		

000016 预测 5 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.7335	0.6928		
Test	0.6674	0.6332	0.007578205	-0.00250105
	Up_hat	Down_hat		
Up	0.304092557	0.163356933		
Down	0.203419099	0.329131411		

000016 预测 15 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.8441	0.8038		
Test	0.7867	0.748	0.008694342	-0.002175022
	Up_hat	Down_hat		
Up	0.344964588	0.127828641		
Down	0.124201071	0.4030057		

000905 预测 1 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.7363	0.6866		
Test	0.7277	0.6759	0.01441122	-0.001600482
	Up_hat	Down_hat		
Up	0.29281768	0.191298343		
Down	0.132769337	0.383114641		

000905 预测 5 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.7308	0.6836		
Test	0.6749	0.6339	0.006235943	-0.003274988
	Up_hat	Down_hat		
Up	0.282334657	0.20652737		
Down	0.159557935	0.351580038		

000905 预测 15 分钟趋势的网络输出

	AUC	Accuracy	Average Daily Return	Base Daily Return
Training	0.8569	0.818		
Test	0.8106	0.753	0.007702383	-0.002714097
	Up_hat	Down_hat		
Up	0.372775954	0.121091726		
Down	0.125928485	0.380203835		

一阶差分后各个价格序列 ADF 检验结果

开盘价：

(-55.703596853684274, 0.0, 9, 28989, {'1%': -3.430575598645753, '5%': -2.861639708375696,

'10%': -2.566823071747864}, 132161.4394109803)

收盘价：

(-53.414218206253224, 0.0, 10, 28988, {'1%': -3.4305756064289263, '5%': -2.861639711815513, '10%': -2.566823073578797}, 131800.9004872062)

最高价：

(-55.29300787424993, 0.0, 9, 28989, {'1%': -3.430575598645753, '5%': -2.861639708375696, '10%': -2.566823071747864}, 130298.51890234533)

最低价：

(-55.618126307372854, 0.0, 9, 28989, {'1%': -3.430575598645753, '5%': -2.861639708375696, '10%': -2.566823071747864}, 129826.79295311362)

附录 B 相关代码

```
import pandas as pd
import numpy as np
import seaborn as sns
from pylab import *
import tensorflow as tf
import time
import warnings
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,roc_auc_score
import os
from tensorflow.models.research.global_objectives import loss_layers
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'  # 设置警告级别
warnings.filterwarnings('ignore')


# 定义参数，7 个
TIMESTEPS = 30  # 循环神经网络的训练序列长度。
HIDDEN_SIZE = 32  # LSTM 中隐藏节点的个数。
NUM_LAYERS = 3  # LSTM 的层数。
DIM_OUTPUT = 1  #最终输出值的 dim
BATCH_SIZE = 128  # batch 大小；最佳:128
state_keep_prob = .5
learning_rate = 0.001
N_EPOCH = 0  # epoch 数
random_state  = 2020
TIMESTEPS_Y =
scaler = StandardScaler()
```

```
#股票代号
stockname = '000016'
#数据集路径
file_name =
#outputs 保存路径
csv_path =
#model 保存路径
ckpt_path =
#恢复 model 时的训练步数
n_start = 0


##################数据处理#################################################
## 初始化方法：对每日数据进行 maxmin 初始化

def load_data(file_name):
    df = pd.read_csv(file_name, sep=',')
    df = np.array(df).astype(float64)
    return df  # 返回无 header 的 np
# x_seq_len: 用以预测的时间长度 ;y_seq_len: 预测的时间长度;以上参数需调

'''
input:data, x_seq_len=10,y_seq_len= 1
output: x, y(with batch)
'''
def get_batch(data, x_seq_len=TIMESTEPS, y_seq_len=TIMESTEPS_Y):
    x = []
    y = []
        #Get onehot
    for i in range(len(data) - x_seq_len - y_seq_len + 1):
```

```
        data_x = data[i: i + x_seq_len,1:]#前闭后开
        data_return = np.sum(data[i + x_seq_len:i + x_seq_len+y_seq_len,2])/ data[i
+ x_seq_len-1,0]
        data_y = 1 if data_return>0 else 0
        data_y = [data_y,data_return]
        data_x = scaler.fit_transform(data_x)# 即 TradeVolume diff_open diff_close
diff_high diff_low
        x.append(data_x)
        y.append(data_y)
    # 得到 x 和 y(最后 y_seq_len 位)
    x = np.array(x).astype('float64')  # 取 seq 的最后的 end_price 价格
    y = np.array(y).astype('float64')  # 取 seq 的最后 y_len 行的 end_price（1 列）
    return x, y
# 划分出连续的 test 数据集
'''不打乱顺序，按比例分出 training & test
output: train_x, train_y, test_x, test_y
'''
def split(x, y, training_pro=0.8):
    # 划分 training，test 集
    train_boundary = int(x.shape[0] * training_pro)
    train_x, train_y = x[: train_boundary, :, :], y[: train_boundary]
    test_x, test_y = x[train_boundary:, :, :], y[train_boundary:]
    return train_x, train_y, test_x, test_y
'''
读取 dir_name 文件夹中的全部 csv,得到 training & test datasets
'''

def get_data(file_name):
    # 打乱顺序，使得每次 test 集都不同
    start_time = time.time()
```

```
    #原csv数据
    data = load_data(file_name)
    #取前1/3数据
    data = data[:(data.shape[0]//3),:]
    X, Y = get_batch(data, TIMESTEPS,TIMESTEPS_Y)
    # Minmaxscaler to all data points
    end_time = time.time()
    print('Reading time is:', end_time - start_time)  # 2.34s,55647 samples
     train_X, test_X, train_y, test_y = train_test_split(X, Y, test_size=.2,
random_state = random_state)
    test_y,test_rr = test_y[:,0],test_y[:,1]
    train_y = train_y[:,0]
    train_X, val_X, train_y, val_y = train_test_split(train_X, train_y,
test_size=.2, random_state = random_state)
    # 划分train & val,打乱顺序
    return train_X, test_X, val_X, train_y.astype(int), test_y.astype(int),
val_y.astype(int),test_rr


##Operations
train_X, test_X, val_X, train_y, test_y, val_y,test_rr = get_data(file_name)

############################定义网络############################################
'''
outputs:predictions, loss(ce), train_op
'''
def lstm_structure(X, y, is_training, state_keep_prob=state_keep_prob,
HIDDEN_SIZE=HIDDEN_SIZE, NUM_LAYERS=NUM_LAYERS,
                   learning_rate=learning_rate):
```

```
    # 使用多层的LSTM结构：NUM_LAYERS层的LSTM网，每层HIDDEN_SIZE个，Dropout法
state_keep_prob=.5
    # activation=tf.nn.relu
    def lstm_cell():
        cell = tf.nn.rnn_cell.BasicLSTMCell(HIDDEN_SIZE,
activation=tf.nn.leaky_relu, name='lstm_cell')
        cell = tf.nn.rnn_cell.DropoutWrapper(cell,
state_keep_prob=state_keep_prob)
        return cell

    multi_lstm = tf.nn.rnn_cell.MultiRNNCell([lstm_cell() for _ in
range(NUM_LAYERS)])
    # 使用TensorFlow接口将多层的LSTM结构连接成RNN网络并计算其前向传播结果。
    outputs, _ = tf.nn.dynamic_rnn(multi_lstm, X, dtype=tf.float64)
    outputs = outputs[:, -1, :]  # 去掉batch维度
    # 对LSTM网络的输出再做加一层全链接层并计算损失。注意这里默认的损失为MSE。
    predictions = tf.contrib.layers.fully_connected(
        outputs, DIM_OUTPUT, activation_fn = tf.nn.sigmoid)
    predictions = tf.reshape(predictions,[-1])
    # 只在训练时计算损失函数和优化步骤。测试时直接返回预测结果。
    if not is_training:
        return predictions, None, None
    # 计算损失函数
    loss = loss_layers.roc_auc_loss(y,predictions)[0]# auc值,返回value&update_op
    loss = tf.reduce_mean(loss)
    #loss = tf.losses.sigmoid_cross_entropy(y,predictions)
    # 创建模型优化器并得到优化步骤。
    # optimizer应为Adam
    train_op = tf.contrib.layers.optimize_loss(
        loss, tf.train.get_global_step(),
```

```
        optimizer="Adam", learning_rate=learning_rate)

    return predictions, loss, train_op

'''
outputs:loss,accuracy, predictions, labels(0:down 1:up)
'''
def run_test(sess, test_X, test_y):
    # 将测试数据以数据集的方式提供给计算图。
    ds = tf.data.Dataset.from_tensor_slices((test_X, test_y))
    ds = ds.batch(1)  # 测试集，部分batch
    iterator = ds.make_one_shot_iterator()
    X, y = iterator.get_next()  # 调用模型得到计算结果。这里不需要输入真实的y值。
    with tf.variable_scope("model", reuse=tf.AUTO_REUSE):
        # 无dropout
        prediction,loss_ce, _ = lstm_structure(X, y, False, state_keep_prob=1)
    # 将预测结果存入一个数组。
    predictions = []
    labels = []
    for i in range(test_X.shape[0]):  # test集长度
        p, l = sess.run([prediction,y])
        predictions.append(p.squeeze())
        labels.append(l.squeeze())  # 计算rmse作为评价指标。
    #0:up 1:Down
    loss = roc_auc_score(labels,predictions)
    predictions =  np.rint(predictions)#四舍五入，<=0.5,down; >=0.5,up
    accuracy = np.mean(predictions==test_y)#np.mean(predictions==labels)
    return loss,accuracy, predictions, labels

'''
```

```
outputs:Acc_test, Loss_test,Acc_train,Loss_train, pre_y_test (最终网络在 test
training 集上的)
'''
def run_training(sess, BATCH_SIZE=BATCH_SIZE,N_EPOCH=N_EPOCH,
state_keep_prob=state_keep_prob, HIDDEN_SIZE=HIDDEN_SIZE,
NUM_LAYERS=NUM_LAYERS,
              learning_rate=learning_rate,restore=True):
    ds = tf.data.Dataset.from_tensor_slices((train_X, train_y))
    ds = ds.shuffle(100000).repeat(N_EPOCH).batch(BATCH_SIZE)  # 不再打乱顺序
    iterator = ds.make_one_shot_iterator()  # 应该Initializable iterator
    X, y = iterator.get_next()

    #定义网络返回值
    with tf.variable_scope("model", initializer=tf.glorot_normal_initializer(),
reuse=tf.AUTO_REUSE):
        # tf.glorot_normal_initializer() as default initializer for variables
within this scope
        _, loss, train_op = lstm_structure(X, y, True,
state_keep_prob=state_keep_prob, HIDDEN_SIZE=HIDDEN_SIZE,
                                   NUM_LAYERS=NUM_LAYERS,
learning_rate=learning_rate)
    #初始化：在网络结构搭建之后！！
    sess.run(tf.global_variables_initializer())
    # Add ops to save and restore all the variables.
    saver = tf.train.Saver()

    if restore == True:
        # restore
        saver.restore(sess, ckpt_path + '-' + str(n_start))
        print("Model restored.")
```

```
        # Later, launch the model, initialize the variables, do some work, save the
variables to disk.


    loss_training = []  # 调steps
    acc_training = []  # 调steps
    loss_val = []
    acc_val = []  # 调steps
    #TRAINING_STEPS = N_EPOCH * len(train_X)//BATCH_SIZE
    best_auc = 0.5
    # 训练TRAINING_STEPS 个batch，即优化TRAINING_STEPS 轮
    for i in range(N_EPOCH):
        # 每个Epoch 进行测试
        #training 集上
        if i%3==0:
            Loss_train, Acc_train, _, _ = run_test(sess, train_X, train_y)
            acc_training.append(Acc_train)
            loss_training.append(Loss_train)


            # 在整个val 集上的测试
            Loss_val, Acc_val, _, _ = run_test(sess, val_X, val_y)
            loss_val.append(Loss_val)  # 每个epoch 计算loss_test
            acc_val.append(Acc_val)


            print(i, 'Epoch')
            print('loss_train:', round(Loss_train, 4), 'accuracy_train:',
round(Acc_train, 4), 'loss_val',round(Loss_val, 4), 'accuracy_val:',
round(Acc_val, 4))


            if i > 0 and loss_val[-1] > best_auc:
                best_epoch = n_epoch + i
```

```
                best_sess = sess


            if i >= 15:
                if abs(loss_val[-1] - mean(loss_val[-5:])) < 0.003:
                    break
    #save_path = saver.save(best_sess, ckpt_path, global_step=best_epoch)
    # 最终模型 在test 集上的 rmse,mae,predictions
    Loss_test,Acc_test,  pre_y_test, labels_test = run_test(sess, test_X, test_y)
    # 最终模型 在training 集上的 rmse,mae
    Loss_train,Acc_train, _ , _= run_test(sess, train_X, train_y)
    # # 画RMSE & MAE training v.s val 训练过程中的曲线 和 直线：train 及test 集上最终
的mae & rmse
    return acc_training,loss_training,acc_val,loss_val,Acc_test,
Loss_test,Acc_train,Loss_train, pre_y_test


def output(csv = True,restore = True):
    start_time = time.time()
    acc_training,loss_training,acc_val,loss_val,Acc_test,
Loss_test,Acc_train,Loss_train,pre_y_test = run_training(sess,restore=restore)
    end_time = time.time()
    # #画训练过程 loss 图
    figure()
    title('AUC & Accuracy training v.s. test')
    plot(loss_training, label='Training loss', marker='o', ms=4,c = 'orange')
    plot(loss_val, label='Validation loss', marker='o', ms=4, c = 'g')
    plot(acc_training, label='Training Accuracy', marker='*', ms=6,c = 'orange')
    plot(acc_val, label='Validation Accuracy', marker='*', ms=6, c = 'g')
    ylim((0.4, 1))
    my_x_ticks = np.arange(0, len(loss_training), 1)
    xticks(my_x_ticks)
```

```
    xlabel('Per 3 Epoches')
    ylabel('Y')
    legend(loc = 'lower right')
    show()
    #Confusion matrix(dataframe)
    C2 = confusion_matrix(pre_y_test > 0, test_y > 0, labels=[True, False])
    C2 = transpose(C2)
    C2 = pd.DataFrame(C2 / len(pre_y_test), columns=['Up_hat', 'Down_hat'],
index=['Up', 'Down'])
    print(C2)  # 打印出来看看
    print(Acc_test)
    Base_acc = mean(test_y[:-TIMESTEPS_Y]==test_y[TIMESTEPS_Y:])
    #计算回收率
    day_returns = sum(test_rr[pre_y_test > 0])*240/((len(test_rr))*TIMESTEPS_Y)
#计算每天收益率
    base_returns = mean(test_rr)*240/TIMESTEPS_Y
    str_returns = np.cumsum(test_rr * pre_y_test)
    real_returns = np.cumsum(test_rr)
    #画收益曲线图
    figure()
    title("Cumulative return("+stockname+'_'+str(TIMESTEPS_Y)+'m)')
    plot(str_returns, label='Strategy Returns', ms=4,c = 'orange')
    plot(real_returns, label='Real Returns', ms=4, c = 'g')
    #ylim((0.4, 1))
    # my_x_ticks = np.arange(0, len(str_returns), 1)
    # xticks(my_x_ticks)
    ylabel('Returns')
    legend(loc = 'upper right')
    show()
```

```
    # 输出 train & test 上的AUC&Accuracy
    df_ef = pd.DataFrame([[round(Loss_train,4),round(Acc_train,4),None,None],
[round(Loss_test,4),round(Acc_test,4),day_returns,base_returns]],
                         columns=['AUC', 'Accuracy','Average Daily Return
Rate','Base Daily Return Rate'], index=['Training', 'Test'])
    print(df_ef)
    # 输出7个参数
    df_par = pd.DataFrame(
        {'Hidden_size': [HIDDEN_SIZE], 'Batch_size': [BATCH_SIZE], 'NUM_LAYERS':
[NUM_LAYERS], 'TIMESTEPS': [TIMESTEPS],
         'TIMESTEPS_Y': [TIMESTEPS_Y], 'Dropout:state_keep_prob':
[state_keep_prob],
         'Adam:learning_rate': [learning_rate],'Training set
size':[train_X.shape], 'Num of Epochs': [N_EPOCH],'Random stat':
[random_state],'Training_time': [end_time -
start_time],'Filename':[file_name.split('/')[-1]]})
    print(transpose(df_par))

    df_training = pd.DataFrame(transpose(np.vstack([loss_training,
loss_val,acc_training,acc_val])), columns=['loss_train',
'loss_val','acc_train','acc_val'])

    # 保存tables
    if csv == False:
        pass
    else:
        df_par.to_csv(csv_path, mode='a')
        df_ef.to_csv(csv_path, mode='a')
        C2.to_csv(csv_path, mode='a')
        df_training.to_csv(csv_path, mode='a')
```

```
    return



##################run default session#####################################
with tf.Session() as sess:
    output(csv = False,restore= True)
```

致　　谢

行文至此，意味着论文已接近结尾，我于东南大学的本科生涯也步入最后的尾声了。

首先，我想要向我的毕业设计导师，钱成老师，致意并感谢。从 19 年末选定导师至今的半年时间里，由于我在伯克利交换与新冠疫情的原因，我和钱成老师从没有正式的见过一面。在这样特殊的时期，钱老师在承受繁重的工作、教学与科研压力之外，仍旧始终保持着对我毕设工作与升学事宜的关心与支持，这使我非常感动，也备受鼓舞。钱老师是东大数院最优秀的毕业生之一。从密歇根大学安娜堡分校获得统计学博士学位归国后，钱老师不忘初心，再次回到东大数院，以统计系教授的身份进行教学与科研，也是因此，我有幸成为了钱老师的学生。而在不久的将来，我也即将赴密大攻读统计学硕士，转换身份，成为钱老师的“学妹”，这是多么奇妙的缘分啊。当然了，自称是“学妹”其实并不合适，仅论专业能力一项，钱老师已经是我难以望其项背之人了。尽管如此，我相信在我未来相当长的一段人生里，钱老师都会是如灯塔一般的方向，指引我不断前行，不问耕耘，不弃匠心，不忘初心。再次感谢钱老师于业、于德、于心对我的言传身教，愿万事胜意。

其次，我要感谢在东大数院相识的可爱的老师们与同学们。作为大二转入数院的转系生和大四一整年在外的交换毕业生，真正与大家分享的时间其实只有两年。这两年中，有过挑灯夜战的艰辛，有过恍然大悟的欣喜，有收到女生节礼物时的小小甜蜜，也有努力不被认可时的怀疑失落。但是在数院学习时，我时常是满足的，满足于知识中的纯粹，满足于思考本身散发的孤美；虽然那时觉得身体疲惫，但头脑总是活跃与激昂的，不似曾经和现在的迷茫。感谢你们在这个美丽世界、在这段美好年华中给予我的引领与陪伴，我也很有幸能为你们带来过一些向着更好的改变。这四年，我愿称不虚此行。朋友们，大步向前走吧，要记得 Stay hungry and stay foolish.

六月快要来了，一起回东大吧。