



同濟大學
TONGJI UNIVERSITY

数据挖掘大作业
LSTM 算法实现上证指数预测

学号：1853862

姓名：王家振

指导老师：向阳

2021. 1. 9

目录

- 1、研究背景和意义..... 3
- 2、问题描述 3
- 3、数据获取与清理..... 3
 - 3.1 数据获取 3
 - 3.2 数据清理 4
- 4、特征工程 6
 - 4.1 特征使用方案 6
 - 4.2 特征处理 7
- 5、模型算法设计14
- 6、实验分析与模型效果评价.....16
- 7、预测结果分析17
- 8、大作业代码.....17
- 9、学习体会24

1、研究背景和意义

从数据挖掘课程本身来说，此次大作业作为本学期数据挖掘课程的期末考试，用以检验本学期所学知识，提高动手实验能力，提高用所学知识解决实际问题的能力。从股票预测研究的角度来说，随着我国经济的不断发展与经济实力的提高，人们对于金融领域的关注度不断增强。而股票市场作为风险和投资双高的投资市场，对股票价格进行预测从而更好地进行选股和择时以获得最大收益也成为了诸多研究者密切关注的领域。当前较为常用的股票预测方法包括技术分析法、基本分析法、时间序列法、神经网络法等多种方法。其中，神经网络法由于其处理非线性问题的独有优势，在股票预测领域得到广泛应用。在预测过程需要重视股票影响因素和模型的构建，通过数据预处理、优化参数、结合多种网络结构等方法提升模型的预测精度。本次大作业通过 LSTM 算法，结合数据清理和衍生参数、特征选择、离散化、归一化、主成分分析等特征工程方法，用以预测上证指数的涨跌，以期实现准确率较高的股票指数预测方案。

2、问题描述

本次大作业以金融领域数据挖掘问题为目标，通过获取行情数据，并利用 talib 库获取行情指标，对行情指标进行特征工程，从而构建数据挖掘模型，并进行模型评价，利用该模型预测明日上证指数（代码：000001.SH）涨跌。

3、数据获取与清理

3.1 数据获取

本次大作业数据从网易财经(<http://quotes.money.163.com/>)获取上证指数历史数据，从英为财经(<https://cn.investing.com/>)获取美元指数期货历史数据，以及美元/人民币汇率历史数据，数据以 CSV 格式存储，文件名分别为 stock.csv, dollar.csv, rate.csv。

3.2 数据清理

3.2.1 一次清理

首先，通过 pandas 的 read_csv 方法，将上述所说的数据读入，因为在这些数据中会出现 None 这样的情况（如图表 1 所示），所以通过去除这些 None 所在行的数据，实现数据的一次清理。另外，对于原本的 csv 文件，它们的日期的数据类型不同（如图表 2 所示），因此，需要将几个文件的日期数据类型统一，然后才能将这几个文件中的数据合并。同时，在美元指数数据和美元/人民币汇率数据中存在“%”（如图表 3 所示），需要将百分号去掉，才能读取为 float 类型。

2	##	'000001	上证指数	99.98	99.98	95.79	96.05	None	None	None	1260	494000
3	##	'000001	上证指数	104.39	104.39	99.98	104.3	99.98	4.41	4.4109	197	84000
4	##	'000001	上证指数	109.13	109.13	103.73	109.07	104.39	4.74	4.5407	28	16000
5	##	'000001	上证指数	114.55	114.55	109.13	113.57	109.13	5.42	4.9666	32	31000

图表 1：None 数据

1	日期	1	日期
2	2010/1/4	2	2010年1月4日
3	2010/1/5	3	2010年1月5日
4	2010/1/6	4	2010年1月6日
5	2010/1/7	5	2010年1月7日
6	2010/1/8	6	2010年1月8日
7	2010/1/11	7	2010年1月11日
8	2010/1/12	8	2010年1月12日

图表 2：日期数据类型不同

	A	B	C	D	E	F	G
1	日期	d收盘价	d开盘价	d最高价	d最低价	d成交量	d涨跌幅
2	#####	77.53	77.92	78.19	77.26	-	-0.42%
3	#####	77.62	77.37	77.71	77.09	-	0.12%
4	#####	77.49	77.65	78	77.36	-	-0.17%
5	#####	77.91	77.37	78.08	77.3	-	0.54%
6	#####	77.47	77.98	78.19	77.35	-	-0.56%
7	#####	77	77.21	77.23	76.79	-	-0.61%
8	#####	76.95	77.09	77.31	76.76	-	-0.06%

图表 3：数据存在%

代码如下：

```

# 从csv中读取数据
np.set_printoptions(suppress=True)
stock = pd.read_csv('stock.csv', encoding='gb2312')
dollar = pd.read_csv('dollar.csv')
rate = pd.read_csv('rate.csv')

# 去除不合理数据
stock=stock.mask(stock.eq('None')).dropna()

# 时间格式转换
stock['日期']=pd.to_datetime(stock['日期'], format='%Y/%m/%d')
dollar['日期']=pd.to_datetime(dollar['日期'], format='%Y年%m月%d日')
rate['日期']=pd.to_datetime(rate['日期'], format='%Y年%m月%d日')

# 去除%
dollar['d涨跌幅']=dollar['d涨跌幅'].str.strip("%").astype(float)
rate['r涨跌幅']=rate['r涨跌幅'].str.strip("%").astype(float)

# 合并
stock=pd.merge(stock, dollar, on='日期')
stock=pd.merge(stock, rate, on='日期')

```

一次清理后效果：

	日期	股票代码	名称	收盘价	最高价	最低价	开盘价	前收盘	涨跌额	涨跌幅	...	d开盘价	d最高价	d最低价	d成交量	d涨跌幅	r收盘	r开盘	r高	r低	r涨跌幅
0	2010-01-04	'000001	上证指数	3243.760	3295.279	3243.319	3289.750	3277.139	-33.379	-1.0185	...	77.92	78.19	77.26	-	-0.42	6.8285	6.8297	6.8297	6.8270	0.02
1	2010-01-05	'000001	上证指数	3282.179	3290.512	3221.462	3254.468	3243.760	38.419	1.1844	...	77.37	77.71	77.09	-	0.12	6.8268	6.8283	6.8288	6.8257	-0.02
2	2010-01-06	'000001	上证指数	3254.215	3295.868	3253.044	3277.517	3282.179	-27.964	-0.8520	...	77.65	78.00	77.36	-	-0.17	6.8278	6.8277	6.8291	6.8267	0.01
3	2010-01-07	'000001	上证指数	3192.776	3268.819	3176.707	3253.991	3254.215	-61.439	-1.8880	...	77.37	78.08	77.30	-	0.54	6.8281	6.8269	6.8282	6.8258	0.00
4	2010-01-08	'000001	上证指数	3195.997	3198.920	3149.017	3177.259	3192.776	3.221	0.1009	...	77.98	78.19	77.35	-	-0.56	6.8276	6.8278	6.8285	6.8260	-0.01

5 rows x 23 columns

3.2.2 二次清理

因为在获取股市技术指标之后，有些技术指标例如 MA 是与平均值相关的，导致技术指标数据个数与原有数据的数据个数不同，会出现许多 Nan 数据（如图表 4 所示），所以要对齐数据，在这里完成数据的二次清理。

```
[      nan      nan      nan      nan 3017.3802 3024.1422 3041.031
3059.1376 3074.1032 3066.6542 3062.4786 3055.5566 3054.7632 3045.1476
3050.7294 3047.1988 3031.9402 3016.6802 3016.9906 3015.952 3026.8196
3046.347 3058.4038 3059.6696 3054.287 3052.68 3062.5244 3077.593
3088.0524 3113.7004]
```

图表 4：MA 数据出现 Nan

代码如下：

```
#去除Nan
stock=stock.dropna()
print(stock)
```

二次清理后效果：

	日期	股票代码	名称	收盘价	最高价	最低价	开盘价	前收盘	涨跌幅	涨跌幅	...	k_data	d_data	j_data	wms_data	rsi	cci_data	macd_data	boll_upper	boll_middle	boll_lower
29	2010-02-12	'000001	上证指数	3018.133	3018.858	2993.437	2996.088	2985.499	32.634	1.0931	...	81.448809	64.185810	115.974807	41.010328	44.070893	89.020124	28.841	3227.855957	3034.59430	2841.332643
30	2010-02-22	'000001	上证指数	3003.398	3026.659	3002.811	3016.703	3018.133	-14.735	-0.4882	...	86.243917	76.846039	105.039674	18.213755	42.473471	106.651209	62.038	3192.606098	3022.90930	2853.212502
31	2010-02-23	'000001	上证指数	2982.575	2998.907	2938.753	2998.907	3003.398	-20.823	-0.6933	...	80.500078	82.730935	76.038364	32.263353	40.253007	2.323344	47.862	3145.310711	3009.69435	2874.077989
32	2010-02-24	'000001	上证指数	3022.177	3023.739	2955.069	2964.984	2982.575	39.602	1.3278	...	79.301110	82.015035	73.873261	3.280200	46.031625	80.006144	18.342	3122.429676	3003.21070	2883.991724
33	2010-02-25	'000001	上证指数	3060.618	3063.004	3022.373	3026.656	3022.177	38.441	1.2720	...	84.410916	81.404035	90.424679	1.379326	50.987051	178.449498	65.310	3097.949016	2998.29845	2898.647884

5 rows x 35 columns

4、特征工程

4.1 特征使用方案

因为本次大作业目标是进行上证指数预测，所以选取了上证指数历史数据，又因为股市涨跌与宏观经济指标、金融指标密不可分，所以选取了美元指数历史数据、美元/人民币汇率历史数据。在实验过程中，原本希望加入更多宏观数据和利率数据。但许多数据获取难度大，例如人民币三大指数数据在中国外汇交易中心网站上只提供近一年的数据，不能满足训练需要；有些数据离散程度大，按月或按季度变化，难以和其他的按天变化的数据对齐，例如新增人民币贷款额、社会融资额以及国家统计局网站提供的各类数据，而且这些数据适用于股市长期预测，不适用于短期预测。基于以上考虑，我选择了上证指数、美元指数、美元

/人民币汇率的数据，在这些数据的基础上计算技术指标，并做进一步的特征工程。

4.2 特征处理

4.2.1 衍生参数

利用原始的上证指数历史数据，即开盘价、收盘价、最高价、最低价、涨跌幅、成交量等，获取股市技术指标。技术指标分析的基本观点认为，所有股票的实际供需量及其背后起引导作用的种种因素，包括股票市场上每个人对未来的希望、担心、恐惧等等，都集中反映在股票的价格和交易量上。由此在“价与量”的基础上，依照一定算法计算出“技术指标”，从而在一定程度上反映股票的走势状况。技术指标有“反趋向指标”、“趋向指标”、“能量指标”、“大盘指标”、“压力支撑指标”等十大类，在本次实验中，选取了其中一些较为重要的技术指标。

1) MA 指标：MA 指标即移动平均线（MA）具有趋势的特性，它比较平稳，不像日 K 线会起起落落地震荡。越长期的移动平均线，越能表现稳定的特性。不轻易向上向下，必须等股价趋势的真正明朗。移动平均线说到底是一种趋势追踪工具，便于识别趋势已经终结或者反转，新的趋势是否正在形成。

2) KDJ 指标：KDJ 指标是一种相当新颖、实用的技术分析指标，它起先用于期货市场的分析，后被广泛用于股市的中短期趋势分析，是期货和股票市场上最常用的技术分析工具。随机指标 KDJ 一般是用于股票分析的统计体系，根据统计学原理，通过一个特定的周期（常为 9 日、9 周等）内出现过的最高价、最低价及最后一个计算周期的收盘价及这三者之间的比例关系，来计算最后一个计算周期的未成熟随机值 RSV，然后根据平滑移动平均线的方法来计算 K 值、D 值与 J 值，并绘成曲线图来研判股票走势。

3) 威廉指数：威廉指数又称威廉超买超卖指数，主要用于研究股价的波动，通过分析股价波动变化中的峰与谷决定买卖时机。它利用振荡点来反映市场的超买超卖现象，可以预测高点与低点，从而显示出有效的买卖信号，是用来分析市场短期行情走势的技术指标。

4) RSI 指标：RSI 指标最早被用于期货交易中，后来人们发现用该指标来指导股票市场投资

效果也十分不错，并对该指标的特点不断进行归纳和总结。现在，RSI 已经成为被投资者应用最广泛的技术指标之一。投资的一般原理认为，投资者的买卖行为是各种因素综合结果的反映，行情的变化最终取决于供求关系，而 RSI 指标正是根据供求平衡的原理，通过测量某一个期间内股价上涨总幅度占股价变化总幅度平均值的百分比，来评估多空力量的强弱程度，进而提示具体操作的。RSI 的应用法则表面上比较复杂，包括了交叉、数值、形态和背离等多方面的判断原则。

5) CCI 指标：CCI 指标是专门测量股价、外汇或者贵金属交易是否已超出常态分布范围。属于超买超卖类指标中较特殊的一种。波动于正无穷大和负无穷大之间。但是，又不需要以 0 为中轴线，这一点也和波动于正无穷大和负无穷大的指标不同。它最早是用于期货市场的判断，后运用于股票市场的研判，并被广泛使用。与大多数单一利用股票的收盘价、开盘价、最高价或最低价而发明出的各种技术分析指标不同，CCI 指标是根据统计学原理，引进价格与固定期间的股价平均区间的偏离程度的概念，强调股价平均绝对偏差在股市技术分析中的重要性，是一种比较独特的技术指标。

6) MOM 指标：MOM 指标即动量线，一般 MOM 指标被视为超买超卖指标，而忽略其在“速度”方面的表现。事实上，将 MOM 解释成“速度线”，更符合其实际的作用。理论上，一波健全的股价趋势，其上涨或下跌的过程，应该维持着一定的行进速度。如果行进的速度逐渐减缓，股价很容易转变成整理的格局，甚至于反转。因此，观察股价的速度感，对于股价多空力道的判定，有很大的帮助。

7) BOLL 指标：BOLL 指标利用统计原理，求出股价的标准差及其信赖区间，从而确定股价的波动范围及未来走势，利用波带显示股价的安全高低价位，因而也被称为布林带。其上下限范围不固定，随股价的滚动而变化。布林指标和麦克指标 MIKE 一样同属路径指标，股价波动在上限和下限的区间之内，这条带状区的宽窄，随着股价波动幅度的大小而变化，股价涨跌幅度加大时，带状区变宽，涨跌幅度狭小盘整时，带状区则变窄。

代码如下：


```
# 加入技术指标
# MA
stock['ma_5_data'] = talib.MA(stock['收盘价'].values, timeperiod=5)
stock['ma_30_data'] = talib.MA(stock['收盘价'].values, timeperiod=30)

# KDJ
stock['k_data'], stock['d_data'] = talib.STOCH(stock['最高价'].values, stock['最低价'].values, stock['收盘价'].values, fastk_period=9, slowk_period=3, slowk_matype=0, slowd_period=3, slowd_matype=0)
stock['j_data'] = 3 * stock['k_data'] - 2 * stock['d_data']

# WMS
stock['wms_data'] = -talib.WILLR(stock['最高价'].values, stock['最低价'].values, stock['收盘价'].values, timeperiod=14)

# RSI
stock['rsi'] = talib.RSI(stock['收盘价'].values)

# CCI
stock['cci_data'] = talib.CCI(stock['最高价'].values, stock['最低价'].values, stock['收盘价'].values)

# MOM
stock['mom_data'] = talib.MOM(stock['收盘价'].values)

# BOLL
stock['boll_upper'], stock['boll_middle'], stock['boll_lower'] = talib.BBANDS(stock['收盘价'].values, timeperiod=20, nbdevup=2, nbdevdn=2, matype=0)
```

衍生参数结果：

	日期	股票代码	名称	收盘价	最高价	最低价	开盘价	前收盘	涨跌额	涨跌幅	...	k_data	d_data	j_data	wms_data	rsi	cci_data	mom_data	boll_upper	boll_middle	boll_lower
29	2010-02-12	'000001	上证指数	3018.133	3018.858	2993.437	2996.088	2985.499	32.634	1.0931	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
30	2010-02-22	'000001	上证指数	3003.398	3026.659	3002.811	3016.793	3018.133	-14.735	-0.4882	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31	2010-02-23	'000001	上证指数	2982.575	2998.907	2938.753	2998.907	3003.398	-20.823	-0.6933	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	2010-02-24	'000001	上证指数	3022.177	3023.739	2955.069	2964.984	2982.575	39.602	1.3278	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
33	2010-02-25	'000001	上证指数	3060.618	3063.004	3022.373	3026.656	3022.177	38.441	1.2720	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows x 35 columns

4.2.2 特征选择

在计算技术指标之后，会发现当前的特征过多，因此需要做特征选择。对于上证指数数据，因为已经有技术指标作为特征，而技术指标都是由原始数据得出，所以可以放弃原始的诸如开、高、低、收等数据。对于美元指数数据，在经过多次实验验证后发现，美元指数的开、高、低、收等数据效果不如美元指数的涨跌幅的数据效果好。因此选择美元指数涨跌幅作为特征。对于美元/人民币汇率数据，与美元指数数据同理，选择汇率涨跌幅作为特征，但是需要注意的是，因为该汇率涨，相当于人民币贬值，而人民币贬值往往代表股市不景气，所以该数据需要取反。

代码如下：

```
# 取技术指标, 美元指数涨跌、USD/CNY汇率涨跌
tech_index=stock.values[:, 23:]
tech_index=np.column_stack((tech_index, stock['d涨跌幅'].values))
tech_index=np.column_stack((tech_index, -stock['r涨跌幅'].values))
tech_index=tech_index.astype('float64')
tech_index
```

特征选择结果:

```
array([[2974.0296, 3098.66296667, 81.44880899, ..., 2841.33264288,
        0.41, 0.01],
       [2987.6744, 3090.6509, 86.24391711, ..., 2853.21250175,
        -0.16, 0.02],
       [2994.421, 3080.6641, 80.50007788, ..., 2874.07798851,
        0.42, -0.],
       ...,
       [3494.00676, 3410.48127, 97.18879219, ..., 3293.92806328,
        0.1, -0.1],
       [3526.35714, 3416.26067333, 99.02595924, ..., 3280.97408321,
        0.32, -0.22],
       [3545.76492, 3423.1867, 96.22471726, ..., 3277.27924023,
        0.31, 0.02]])
```

4.2.3 离散化

通常在机器学习任务中, 通过离散化, 可以将连续特征离散化为许多离散的特征, 然后再输入到模型中。利用离散化有以下优点:

- 1) 离散化特征对异常值有很强的鲁棒性, 可以在一定程度上有去除噪声, 使得模型会更加稳定;
- 2) 增加和减少离散特征都很容易;
- 3) 可以简化模型, 降低模型过拟合的风险。
- 4) 离散数据一般为整数, 运算速度快;

本次大作业, 针对上述的这些技术指标, 根据每个指标的具体含义, 进行离散化操作, 将数据变成-1 或者 0 或者 1。

代码如下:

```
# 离散化
feature_tech_index = tech_index.copy()
for i in range(tech_index.shape[0]):
    # MA
    feature_tech_index[i][0]=0
    if i>0 and tech_index[i-1][0]<tech_index[i-1][1] and tech_index[i][0]>tech_index[i][1] :
        feature_tech_index[i][0]=1
    elif i>0 and tech_index[i-1][0]>tech_index[i-1][1] and tech_index[i][0]>tech_index[i][1] and
tech_index[i][0]<tech_index[i-1][0] :
        feature_tech_index[i][0]=1
    elif i>0 and tech_index[i-1][0]>tech_index[i-1][1] and tech_index[i][0]<tech_index[i][1] :
        feature_tech_index[i][0]=-1
    elif i>0 and tech_index[i-1][0]<tech_index[i-1][1] and tech_index[i][0]<tech_index[i][1] and
tech_index[i][0]>tech_index[i-1][0] :
        feature_tech_index[i][0]=-1
    feature_tech_index[i][1]=feature_tech_index[i][0]

    # K
    feature_tech_index[i][2]=0
    if tech_index[i][2]>90 :
        feature_tech_index[i][2]=-1
    elif tech_index[i][2]<10 :
        feature_tech_index[i][2]=1
    elif i>0 and tech_index[i-1][2]<tech_index[i-1][3] and tech_index[i][2]>tech_index[i][3] :#gold
        feature_tech_index[i][2]=1
    elif i>0 and tech_index[i-1][2]>tech_index[i-1][3] and tech_index[i][2]<tech_index[i][3] :#death
        feature_tech_index[i][2]=-1

    # D
    feature_tech_index[i][3]=0
    if tech_index[i][3]>80 :
        feature_tech_index[i][3]=-1
    elif tech_index[i][3]<20 :
        feature_tech_index[i][3]=1
    elif i>0 and tech_index[i-1][2]<tech_index[i-1][3] and tech_index[i][2]>tech_index[i][3] :#gold
        feature_tech_index[i][3]=1
    elif i>0 and tech_index[i-1][2]>tech_index[i-1][3] and tech_index[i][2]<tech_index[i][3] :#death
        feature_tech_index[i][3]=-1

    # J
    feature_tech_index[i][4]=0
    if tech_index[i][4]>100 :
        feature_tech_index[i][4]=-1
    elif feature_tech_index[i][4]<0 :
        feature_tech_index[i][4]=1
```

```

# WMS
feature_tech_index[i][5]=0
if tech_index[i][5]>80 :
    feature_tech_index[i][5]=1
elif tech_index[i][5]<20 :
    feature_tech_index[i][5]=-1

# RST
feature_tech_index[i][6]=0
if i>0 and tech_index[i-1][6]<50 and tech_index[i][6]>50 :
    feature_tech_index[i][6]=1

elif i>0 and tech_index[i-1][6]>50 and tech_index[i][6]<50 :
    feature_tech_index[i][6]=-1

elif tech_index[i][6]>80 :
    feature_tech_index[i][6]=-1

elif tech_index[i][6]<20 :
    feature_tech_index[i][6]=1

# CCI
feature_tech_index[i][7]=0
if i>0 and tech_index[i-1][7]>100 and tech_index[i][7]<100 :
    feature_tech_index[i][7]=-1
elif i>0 and tech_index[i-1][7]<100 and tech_index[i][7]>100 :
    feature_tech_index[i][7]=1
elif i>0 and tech_index[i-1][7]>-100 and tech_index[i][7]<-100 :
    feature_tech_index[i][7]=-1
elif i>0 and tech_index[i-1][7]>-100 and tech_index[i][7]<-100 :
    feature_tech_index[i][7]=1

# MOM
feature_tech_index[i][8]=0
if i>0 and tech_index[i-1][8]>600 and tech_index[i][8]>600 and tech_index[i][8]<tech_index[i-1][8] :
    feature_tech_index[i][8]=-1
elif i>0 and tech_index[i-1][8]<-600 and tech_index[i][8]<-600 and tech_index[i][8]>tech_index[i-1][8] :
    feature_tech_index[i][8]=1
elif i>0 and tech_index[i-1][8]>400 and tech_index[i][8]>400 and tech_index[i][8]>tech_index[i-1][8] :
    feature_tech_index[i][8]=-1
elif i>0 and tech_index[i-1][8]<-400 and tech_index[i][8]<-400 and tech_index[i][8]<tech_index[i-1][8] :
    feature_tech_index[i][8]=1
elif i>0 and tech_index[i-1][8]>200 and tech_index[i][8]>200 and tech_index[i][8]>tech_index[i-1][8] :
    feature_tech_index[i][8]=-1
elif i>0 and tech_index[i-1][8]<-200 and tech_index[i][8]<-200 and tech_index[i][8]<tech_index[i-1][8] :
    feature_tech_index[i][8]=1

# BOLL
feature_tech_index[i][9]=0
if i>0 and stock['开盘价'].values[i]>tech_index[i-1][9] :
    feature_tech_index[i][9]=1
elif i>0 and stock['开盘价'].values[i]<tech_index[i-1][11] :
    feature_tech_index[i][9]=-1
feature_tech_index[i][10]=feature_tech_index[i][9]
feature_tech_index[i][11]=feature_tech_index[i][9]

```

离散化结果:

```

array([[ 0. ,  0. , -1. , ...,  0. , -0.36,  0.01],
       [ 0. ,  0. , -1. , ..., -1. ,  0.63, -0.01],
       [ 0. ,  0. , -1. , ..., -1. ,  0.37,  0. ],
       ...,
       [ 0. ,  0. , -1. , ..., -1. ,  0.1 ,  0.1 ],
       [ 0. ,  0. , -1. , ..., -1. ,  0.32,  0.22],
       [ 0. ,  0. , -1. , ..., -1. ,  0.31, -0.02]])

```

4.2.4 归一化

如果不进行归一化，那么由于特征向量中不同特征的取值相差较大，会导致目标函数变“扁”。这样在进行梯度下降的时候，梯度的方向就会偏离最小值的方向，走很多弯路，即训练时间过长。另外由于各类特征之间具有不同的量纲，同时数据范围也具有较大差异，所有需要对不同量纲和不同数据范围的数据进行归一化处理，即针对其他未进行离散化的特征，进行归一化操作。

本次大作业，技术指标之外是数据，进行归一化操作。

代码如下：

```
# 归一化
feature_tech_index[:,12:]=(feature_tech_index[:,12:]-
np.mean(feature_tech_index[:,12:],axis=0))/np.std(feature_tech_index[:,12:],axis=0)
```

归一化结果：

```
array([[ 0.          ,  0.          , -1.          , ...,  0.          ,
        -0.81474723,  0.05830435],
       [ 0.          ,  0.          , -1.          , ..., -1.          ,
        1.39979291, -0.04229321],
       [ 0.          ,  0.          , -1.          , ..., -1.          ,
        0.81819651,  0.00800557],
       ...,
       [ 0.          ,  0.          , -1.          , ..., -1.          ,
        0.21423102,  0.51099334],
       [ 0.          ,  0.          , -1.          , ..., -1.          ,
        0.70635105,  1.11457867],
       [ 0.          ,  0.          , -1.          , ..., -1.          ,
        0.68398196, -0.09259199]])
```

4.2.5 PCA 降维

主成分分析算法是最常用的线性降维方法，它的目标是通过某种线性投影，将高维的数据映射到低维的空间中，并期望在所投影的维度上数据的信息量最大（方差最大），以此使用较少的数据维度，同时保留住较多的原数据点的特性。

PCA 降维的目的，就是为了在尽量保证“信息量不丢失”的情况下，对原始特征进行降维，也就是尽可能将原始特征往具有最大投影信息量的维度上进行投影。将原特征投影到这些维度上，使降维后信息量损失最小。

本次大作业在进行上述操作之后，共有 14 个特征，特征数量大，且难以区分那个特征的影响最大，因此，选择采用 PCA 的方法降维到 5 个特征。

代码如下：

```
# PCA
estimator = PCA(n_components=5)
pca_tech_index = estimator.fit_transform(feature_tech_index)
print(pca_tech_index.shape)
```

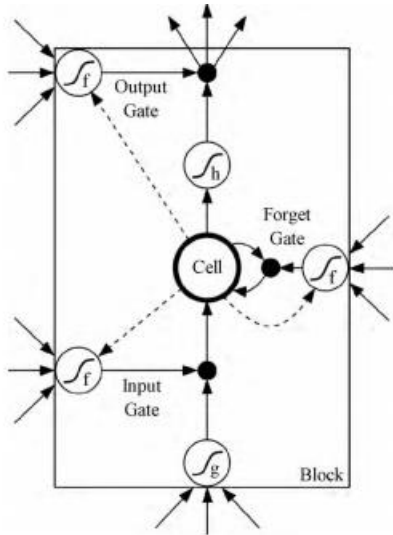
PCA 后效果：

```
array([[ 0.58354639, -0.08766052, -0.47184584, -0.49663733, -0.15726437],
       [-0.62742552, -0.24893585, -1.23486869,  1.04560543, -0.90368448],
       [ 0.48274486, -1.0149992 , -1.3695898 ,  0.25196581,  1.30996195],
       ...,
       [ 0.15047793, -2.03409971,  0.18172597,  0.42882728, -0.31972703],
       [ 0.88381524, -2.25147134,  0.09285191,  0.60553042, -0.50381367],
       [ 0.03755963, -2.11943371, -0.20653311, -0.2168622 , -0.29372195]])
```

5、模型算法设计

本次大作业采用 LSTM 模型。LSTM 是一种时间循环神经网络，是为了解决一般的 RNN（循环神经网络）存在的长期依赖问题而专门设计出来的，所有的 RNN 都具有一种重复神经网络模块的链式形式。

在本次大作业中，经过多次实验验证，最终采用两层 LSTM，同时在每层 LSTM 后加入 Dropout 技术，并将失活概率设为 0.2，可以通过阻止神经元共适应能够缓解过拟合问题。



图表 5: LSTM 神经元示意图

代码如下：

```
# LSTM模型搭建
model=Sequential()
model.add(LSTM(units=50, activation='relu',input_shape=(x_train.shape[1],
x_train.shape[2]),return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50, activation='relu', return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=25))
model.add(Dense(y_train.shape[1], activation='softmax'))
model.summary()
```

模型如下：

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 15, 50)	11200
dropout (Dropout)	(None, 15, 50)	0
lstm_1 (LSTM)	(None, 50)	20200
dropout_1 (Dropout)	(None, 50)	0
dense (Dense)	(None, 25)	1275
dense_1 (Dense)	(None, 2)	52
Total params: 32,727		
Trainable params: 32,727		
Non-trainable params: 0		

6、实验分析与模型效果评价

在模型训练过程中，采用自 2010 年起共约 10 年的数据进行训练，80%的数据作为训练集，20%的数据作为验证集。模型预测方式为使用前 15 个交易日的数据预测第 16 日的涨跌情况。模型训练中，采用 Mini-Batch 方法训练 LSTM 网络。损失函数采用多类的对数损失。优化器方面，本文采用 Adam 优化器，与其他自适应学习率算法相比，Adam 算法收敛速度更快、学习效果更为有效。为防止发生过拟合，共训练 5 个 Epoch。

最终从模型效果来看，训练集准确率最高大约 54%，验证集准确率最高大约 52%。经过多次实验验证，模型情况较为稳定。

模型情况基本符合预期，也符合生活经验。因为在长期情况下，对于每天的股市涨跌情况进行预测本身就是一个难以完成的任务，尤其是在忽略了经济基本面情况、国家货币政策、财政政策、地缘政治情况等一系列重要因素的情况下。

代码如下：

```
# 模型训练
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
h = model.fit(x_train, y_train, validation_split=0.2, batch_size=1, epochs=5)
```


训练效果如下：

```
Epoch 1/5  
1896/1896 [=====] - 7s 4ms/step - loss: 0.6945 - accuracy: 0.5116 - val_loss: 0.6930 - val_accuracy: 0.5042  
Epoch 2/5  
1896/1896 [=====] - 7s 4ms/step - loss: 0.6927 - accuracy: 0.5301 - val_loss: 0.6925 - val_accuracy: 0.5105  
Epoch 3/5  
1896/1896 [=====] - 7s 4ms/step - loss: 0.6918 - accuracy: 0.5253 - val_loss: 0.6925 - val_accuracy: 0.5211  
Epoch 4/5  
1896/1896 [=====] - 7s 4ms/step - loss: 0.6928 - accuracy: 0.5380 - val_loss: 0.6946 - val_accuracy: 0.5000  
Epoch 5/5  
1896/1896 [=====] - 7s 4ms/step - loss: 0.6913 - accuracy: 0.5285 - val_loss: 0.6926 - val_accuracy: 0.5211
```

7、预测结果分析

本次大作业采用自 2020. 12. 21 开始到 2021. 1. 7 日结束，共三周，12 天的测试情况，进行测试。最终测试集准确率大约 75%，即在这 12 天中，共预测正确 9 天。之所以在训练集上只有 54% 的准确率，而在测试集上有约 75% 的准确率，个人猜测原因有两点。一是因为测试集天数较少，情况并不全面。二是可能构造的模型整体偏乐观，偏向于买多，而同时最近股市也是上涨的较多，所以准确率较高。

代码如下：

```
# 测试  
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)  
print(f'测试集损失值: {test_loss}, 测试集准确率: {test_acc}')
```

测试结果如下：

测试集损失值：0.65190190076828，测试集准确率：0.75

8、大作业代码

```
# 引入相关的包  
import pandas as pd  
import numpy as np  
import talib  
import matplotlib.pyplot as plt  
from datetime import datetime, date  
from keras.models import Sequential  
from keras.layers import Dense, LSTM, Dropout  
from keras import utils
```

```

from keras import optimizers
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from collections import Counter
from sklearn.decomposition import PCA

# 从 csv 中读取数据
np.set_printoptions(suppress=True)
stock = pd.read_csv('stock.csv', encoding='gb2312')
dollar = pd.read_csv('dollar.csv')
rate = pd.read_csv('rate.csv')

# 去除不合理数据
stock=stock.mask(stock.eq('None')).dropna()

# 时间格式转换
stock['日期']=pd.to_datetime(stock['日期'], format='%Y/%m/%d')
dollar['日期']=pd.to_datetime(dollar['日期'], format='%Y 年%m 月%d 日')
rate['日期']=pd.to_datetime(rate['日期'], format='%Y 年%m 月%d 日')

# 去除%
dollar['d 涨跌幅']=dollar['d 涨跌幅'].str.strip("%").astype(float)
rate['r 涨跌幅']=rate['r 涨跌幅'].str.strip("%").astype(float)

# 合并
stock=pd.merge(stock, dollar, on='日期')
stock=pd.merge(stock, rate, on='日期')
stock.head()

# 加入技术指标
# MA
stock['ma_5_data'] = talib.MA(stock['收盘价'].values, timeperiod=5)
stock['ma_30_data'] = talib.MA(stock['收盘价'].values, timeperiod=30)

# KDJ
stock['k_data'], stock['d_data'] = talib.STOCH(stock['最高价'].values, stock['最低价'].values, stock['收盘价'].values, fastk_period=9, slowk_period=3, slowk_matype=0, slowd_period=3, slowd_matype=0)
stock['j_data'] = 3 * stock['k_data'] - 2 * stock['d_data']

# WMS

```

```

stock['wms_data'] = -talib.WILLR(stock['最高价'].values, stock['最低价'].values, stock['收盘价'].values, timeperiod=14)

# RSI
stock['rsi'] = talib.RSI(stock['收盘价'].values)

# CCI
stock['cci_data'] = talib.CCI(stock['最高价'].values, stock['最低价'].values, stock['收盘价'].values)

# MOM
stock['mom_data'] = talib.MOM(stock['收盘价'].values)

# BOLL
stock['boll_upper'], stock['boll_middle'], stock['boll_lower'] = talib.BBANDS(stock['收盘价'].values, timeperiod=20, nbdevup=2, nbdevdn=2, matype=0)

stock.head()

#去除 Nan
stock=stock.dropna()
stock.head()

# 取技术指标, 美元指数涨跌、USD/CNY 汇率涨跌
tech_index=stock.values[:, 23:]
tech_index=np.column_stack((tech_index, stock['d 涨跌幅'].values))
tech_index=np.column_stack((tech_index, -stock['r 涨跌幅'].values))
tech_index=tech_index.astype('float64')
tech_index

# 离散化
feature_tech_index = tech_index.copy()
for i in range(tech_index.shape[0]):
    # MA
    feature_tech_index[i][0]=0
    if i>0 and tech_index[i-1][0]<tech_index[i-1][1] and tech_index[i][0]>tech_index[i][1] :
        feature_tech_index[i][0]=1
    elif i>0 and tech_index[i-1][0]>tech_index[i-1][1] and tech_index[i][0]<tech_index[i-1][0] :
        feature_tech_index[i][0]=1

```

```

        elif i>0 and tech_index[i-1][0]>tech_index[i-
1][1] and tech_index[i][0]<tech_index[i][1] :
            feature_tech_index[i][0]=-1
        elif i>0 and tech_index[i-1][0]<tech_index[i-
1][1] and tech_index[i][0]<tech_index[i][1] and tech_index[i][0]>tech_i
ndex[i-1][0] :
            feature_tech_index[i][0]=-1
            feature_tech_index[i][1]=feature_tech_index[i][0]

# K
feature_tech_index[i][2]=0
if tech_index[i][2]>90 :
    feature_tech_index[i][2]=-1
elif tech_index[i][2]<10 :
    feature_tech_index[i][2]=1
elif i>0 and tech_index[i-1][2]<tech_index[i-
1][3] and tech_index[i][2]>tech_index[i][3] :#gold
    feature_tech_index[i][2]=1
elif i>0 and tech_index[i-1][2]>tech_index[i-
1][3] and tech_index[i][2]<tech_index[i][3] :#death
    feature_tech_index[i][2]=-1

# D
feature_tech_index[i][3]=0
if tech_index[i][3]>80 :
    feature_tech_index[i][3]=-1
elif tech_index[i][3]<20 :
    feature_tech_index[i][3]=1
elif i>0 and tech_index[i-1][2]<tech_index[i-
1][3] and tech_index[i][2]>tech_index[i][3] :#gold
    feature_tech_index[i][3]=1
elif i>0 and tech_index[i-1][2]>tech_index[i-
1][3] and tech_index[i][2]<tech_index[i][3] :#death
    feature_tech_index[i][3]=-1

# J
feature_tech_index[i][4]=0
if tech_index[i][4]>100 :
    feature_tech_index[i][4]=-1
elif feature_tech_index[i][4]<0 :
    feature_tech_index[i][4]=1

# WMS
feature_tech_index[i][5]=0

```

```

if tech_index[i][5]>80 :
    feature_tech_index[i][5]=1
elif tech_index[i][5]<20 :
    feature_tech_index[i][5]=-1

# RST
feature_tech_index[i][6]=0
if i>0 and tech_index[i-1][6]<50 and tech_index[i][6]>50 :
    feature_tech_index[i][6]=1

elif i>0 and tech_index[i-1][6]>50 and tech_index[i][6]<50 :
    feature_tech_index[i][6]=-1

elif tech_index[i][6]>80 :
    feature_tech_index[i][6]=-1

elif tech_index[i][6]<20 :
    feature_tech_index[i][6]=1

# CCI
feature_tech_index[i][7]=0
if i>0 and tech_index[i-1][7]>100 and tech_index[i][7]<100 :
    feature_tech_index[i][7]=-1
elif i>0 and tech_index[i-1][7]<100 and tech_index[i][7]>100 :
    feature_tech_index[i][7]=1
elif i>0 and tech_index[i-1][7]>-100 and tech_index[i][7]<-100 :
    feature_tech_index[i][7]=-1
elif i>0 and tech_index[i-1][7]>-100 and tech_index[i][7]<-100 :
    feature_tech_index[i][7]=1

# MOM
feature_tech_index[i][8]=0
if i>0 and tech_index[i-
1][8]>600 and tech_index[i][8]>600 and tech_index[i][8]<tech_index[i-
1][8] :
    feature_tech_index[i][8]=-1
elif i>0 and tech_index[i-1][8]<-600 and tech_index[i][8]<-
600 and tech_index[i][8]>tech_index[i-1][8] :
    feature_tech_index[i][8]=1
elif i>0 and tech_index[i-
1][8]>400 and tech_index[i][8]>400 and tech_index[i][8]>tech_index[i-
1][8] :
    feature_tech_index[i][8]=-1

```

```

        elif i>0 and tech_index[i-1][8]<-400 and tech_index[i][8]<-
400 and tech_index[i][8]<tech_index[i-1][8] :
            feature_tech_index[i][8]=1
        elif i>0 and tech_index[i-
1][8]>200 and tech_index[i][8]>200 and tech_index[i][8]>tech_index[i-
1][8] :
            feature_tech_index[i][8]=-1
        elif i>0 and tech_index[i-1][8]<-200 and tech_index[i][8]<-
200 and tech_index[i][8]<tech_index[i-1][8] :
            feature_tech_index[i][8]=1

```

BOLL

```

feature_tech_index[i][9]=0
if i>0 and stock['开盘价'].values[i]>tech_index[i-1][9] :
    feature_tech_index[i][9]=-1
elif i>0 and stock['开盘价'].values[i]<tech_index[i-1][11] :
    feature_tech_index[i][9]=1
feature_tech_index[i][10]=feature_tech_index[i][9]
feature_tech_index[i][11]=feature_tech_index[i][9]

```

feature_tech_index

归一化

```

feature_tech_index[:,12:]=(feature_tech_index[:,12:]-
np.mean(feature_tech_index[:,12:],axis=0))/np.std(feature_tech_index[:,
12:],axis=0)
# feature_tech_index=(feature_tech_index-
np.mean(feature_tech_index,axis=0))/np.std(feature_tech_index,axis=0)
feature_tech_index

```

PCA

```

estimator = PCA(n_components=5)
pca_tech_index = estimator.fit_transform(feature_tech_index)
print(pca_tech_index.shape)
pca_tech_index

```

划分训练集、测试集

```

x_train=[]
y_train=[]
x_test=[]
y_test=[]
divide = pca_tech_index.shape[0]*9/10
for i in range(15, tech_index.shape[0]):
    if i<divide:

```

```

        tmp = pca_tech_index[i-15:i, :]
        x_train.append(tmp)
        if float(stock.iloc[i]['涨跌幅'])>0:
            y_train.append(1)
        else:
            y_train.append(0)
    elif i>2635:
        tmp=pca_tech_index[i-15:i,:]
        x_test.append(tmp)
        if float(stock.iloc[i]['涨跌幅'])>0:
            y_test.append(1)
        else:
            y_test.append(0)

x_train,y_train,x_test,y_test=np.array(x_train),np.array(y_train),np.array(x_test),np.array(y_test)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
print(x_train.dtype)
print(x_train[0])
print(y_train)

# 独热化
label_cnt=len(Counter(y_train))
y_train=utils.to_categorical(y_train, label_cnt)
y_test=utils.to_categorical(y_test, label_cnt)
print(y_train)

# LSTM 模型搭建
model=Sequential()
model.add(LSTM(units=50, activation='relu',input_shape=(x_train.shape[1], x_train.shape[2]),return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50, activation='relu', return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=25))
model.add(Dense(y_train.shape[1], activation='softmax'))
model.summary()

# 模型训练
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

```
h = model.fit(x_train, y_train, validation_split=0.2, batch_size=1, epochs=5)

# 测试
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f'测试集损失值: {test_loss}, 测试集准确率: {test_acc}')
```

9、学习体会

对于本次大作业，首先的一点感受就是对于数据挖掘、对于机器学习和特征处理以及 Python 语言的理解更加深刻了。

在本次实验过程中，涉及到大量与股票相关的数据获取、数据清洗、特征处理等操作，因此，需要自主学习很多和股票相关的知识，尤其是与股票技术指标相关的知识。

同时，在本次实验过程中，我通过查询网上资料，尤其是 talib 官网提供的 API，了解了一系列股市技术指标，然后依次查询各个技术指标，了解这些技术指标的原理，然后利用 talib 的 API 获取这些技术指标，并且在离散化时将这些指标进行离散处理。

另外，我在网上搜索了许多与股票预测相关的论文，然后决定采用 LSTM 模型，并且在每层 LSTM 后添加 Dropout 层防止过拟合。

在实验过程中，我尽量采用了高效的设计方式进行编码，通过对每一个模块的设计，使得我对理论课程所学习的内容有了更深的理解和体会，使得我对于数据挖掘这门课的认识更加深刻了。当最终结果稳定的出现时，我如释重负，感觉自己的努力终究没有白费。

本次实验也大大增加了我对数据挖掘的兴趣。我其实本来就对于机器学习比较感兴趣，并一直有志于为我国机器学习的发展贡献出自己的一份力。在这次实验中，我感觉到机器学习过程的乐趣，并对于当前这个领域面临的问题与解决方案有了一个基本的了解。虽然花费了不少的时间，但是有着巨大的收获，非常值得。

10、参加课外比赛

在课程之外，我还参与了 CCF 大数据与计算智能大赛，选择了房产行业聊天问答匹配赛道。在这个比赛中，我们采用了 Bert+Transfer Learning 的方法，搭建了模型。模型代码如下：


```

#加载预训练bert模型
bert_model = load_trained_model_from_checkpoint(config_path, checkpoint_path, seq_len=None)

for l in bert_model.layers:
    l.trainable = True

x1_in = Input(shape=(None,))
x2_in = Input(shape=(None,))

x = bert_model([x1_in, x2_in])
x = Lambda(lambda x: x[:, 0])(x) #模型训练结果的第一位 [cls] 进行预测
p = Dense(1, activation='sigmoid')(x) #加sigmoid线性层

model = Model([x1_in, x2_in], p)
model.compile(
    loss='binary_crossentropy', #binary_crossentropy与sigmoid对应
    optimizer=Adam(1e-5), # 用足够小的学习率
    metrics=['accuracy']
)
model.summary()

```

比赛相关截图：



黄金矿工分工


挖矿了

190268 | 导师信息

☞ 向阳

🏫 同济大学-电子信息与工程学院

队伍成员



赵旭豪

没吃饱啊今天

C/C++

发送私信



王家振

???脸

尚未填写技能标签

A 榜

B 榜

我的成绩

到目前为止，您的最好成绩为 **0.75393103** 分，第 **417** 名，在本阶段中，您已超越 **156** 支队伍。