摘 要

ABSTRACT

目 录

[摘 要 I](#_Toc6945096)

[ABSTRACT II](#_Toc6945097)

[目 录 III](#_Toc6945098)

[第一章 绪 论 1](#_Toc6945099)

[1.1 选题背景与意义 1](#_Toc6945100)

[1.2 国内外研究现状 1](#_Toc6945101)

[1.3 本文主要内容 1](#_Toc6945102)

[1.4 本文的组织结构与技术路线 1](#_Toc6945103)

[第二章 深度学习理论基础 2](#_Toc6945104)

[2.1 人工神经网络 2](#_Toc6945105)

[2.2全连接神经网络 3](#_Toc6945106)

[2.2.1 前馈神经网络 3](#_Toc6945107)

[2.2.2 全连接层和全连接神经网络 4](#_Toc6945108)

[2.2.3 损失函数 4](#_Toc6945109)

[2.2.4 全连接神经网络的训练方法 4](#_Toc6945110)

[2.3 循环神经网络 5](#_Toc6945111)

[2.3.1 循环神经网络的结构 5](#_Toc6945112)

[2.3.2 循环神经网络的训练 6](#_Toc6945113)

[2.4 长短期记忆网络 9](#_Toc6945114)

[2.4.1 长短期记忆网络的结构 9](#_Toc6945115)

[2.4.2 长短期记忆网络的训练 10](#_Toc6945116)

[2.5 支持向量机 14](#_Toc6945117)

[2.6 神经网络训练的优化方法 15](#_Toc6945118)

[2.6.1 梯度下降算法的缺陷 15](#_Toc6945119)

[2.6.2 解决方法 15](#_Toc6945120)

[第三章 小波变换 17](#_Toc6945121)

[3.1 傅里叶变换 17](#_Toc6945122)

[3.2 短时傅里叶变换 17](#_Toc6945123)

[3.3 小波变换 18](#_Toc6945124)

[3.3.1 连续小波变换 18](#_Toc6945125)

[3.3.2 离散小波变换 19](#_Toc6945126)

[第四章 模型构建 20](#_Toc6945127)

[4.1 使用工具 20](#_Toc6945128)

[4.1.1 Python 20](#_Toc6945129)

[4.1.2 Numpy 20](#_Toc6945130)

[4.1.3 PyTorch 20](#_Toc6945131)

[4.1.4 Pandas 21](#_Toc6945132)

[4.1.5 Matplotlib 21](#_Toc6945133)

[4.1.6 PyWavelet 21](#_Toc6945134)

[4.2 基于LSTM的神经网络模型构建 21](#_Toc6945135)

[4.3 基于WT+SAE+LSTM的神经网络模型构建 22](#_Toc6945136)

[4.4 数据处理 22](#_Toc6945137)

[第五章 股票数据的选取 24](#_Toc6945138)

[5.1 数据来源 24](#_Toc6945139)

[5.2 上证指数预测 24](#_Toc6945140)

[5.2.1 上证指数的计算方法 24](#_Toc6945141)

[5.3 个股股价预测 24](#_Toc6945142)

[第六章 实验分析 25](#_Toc6945143)

[6.1 优化方法 25](#_Toc6945144)

[6.2 参数设置 25](#_Toc6945145)

[6.3 整体效果 25](#_Toc6945146)

[6.4 指标分析 25](#_Toc6945147)

[6.5 几种解决方法的对比 25](#_Toc6945148)

[致 谢 26](#_Toc6945149)

[参考文献 27](#_Toc6945150)

[外文资料原文 28](#_Toc6945151)

[外文资料译文 29](#_Toc6945152)

第一章 绪 论

1.1 选题背景与意义

1.2 国内外研究现状

1.3 本文主要内容

1.4 本文的组织结构与技术路线

第二章 深度学习理论基础

2.1 人工神经网络

在生物学中，生物获取外界信息是由它们的神经网络系统来完成的。神经网络系统最基本的结构和功能单位是神经细胞（即神经元）。神经细胞可分为细胞体和突起两部分。其中突起又由树突和轴突两部分构成。树突的主要作用是：接收其他神经元传递来的电信号。轴突的主要作用是：将神经元的电信号传递给其他神经元的树突。

（生物神经元图片）

人工神经网络（Artificial Neural Network, ANN）简称神经网络，是一种模仿生物神经网络的结构和功能的数学模型或计算机模型。人工神经网络是由大量的人工神经元组成的，它们联合起来达到计算的目的。人工神经元的实现很大程度上受到了生物神经元的启发。生物神经元需要接收来自之前神经元（往往不止一个）的电信号，还需要给接下来的神经元传递电信号。所以数学家们和计算机科学家们利用了生物神经元的这一特性，提出了人工神经元。在人工神经元中，输入为一系列的特征值，输出为一个结果。在人工神经网络中传递的不是生物神经元中的电信号，而是一系列的特征值或计算结果。讲很多神经元组合起来，就形成了人工神经网络。图1-1为一个最简单的人工神经元。在一个最简单的神经元中，它要接收前一层神经元的输出，记为，其中为前一层第个神经元的输出。当它接收到前一层神经元的输出后，要将乘以一个权值，将所有乘积求和再加上一个偏移量得到，即（假设前一层共有个神经元）。我们计，，那么上式可以写成。在得到了之后，这一个神经元的输出值，其中为一个激活函数（activation function）。

（人工神经元图片）

常用的激活函数有以下几种：

1）Sigmoid函数

Sigmoid函数是一个S型函数（如图1-2）。它将变量映射到区间中，它的函数表达式是

（Sigmoid函数图片）

2）Tanh函数

Tanh函数是双曲函数中的一个，它的函数表达式是

（Tanh函数图片）

3）ReLU函数

ReLU函数（The Rectified Linear Unit）函数表达式是：

（ReLU函数图片）

所以一个神经元的计算过程为。

2.2全连接神经网络

2.2.1 前馈神经网络

神经网络是由很多个神经元连接成的网络。对于一个神经网络来说，如果各个神经元从输入层开始，接收前一层的输出，并输出到下一层，直到输出层，并且整个神经网络中无反馈（即可用一个有向无环图来表示整个神经网络），那么称这样的网络为前馈神经网络（Feedforward Neural Network）。前馈神经网络为最早发明的人工神经网络、是一种最简单的神经网络。通俗地讲，在前馈神经网络中，每一个神经元与前一层的神经元相连接，每一个单独的神经元接收前一层神经元的输出，并将数据处理后，传递给下一个神经元，同一层神经元之间没有连接。

2.2.2.1 单层前馈神经网络

顾名思义，单层前馈神经网络中只有一层神经元。

（数学公式）

（单层前馈神经网络图片）

2.2.2.2 多层前馈神经网络

一般说来，多层前馈神经网络的结构就是将多个单层前馈神经网络的神经元依次连接。一般情况下，将最后一层神经元称为输出层（output layer），前面几层神经元称为隐藏层（hidden layer）。多层前馈神经网络中，存在着一个或多个隐藏层。在多层前馈神经网络中每一层神经元的集合相当于一个单层前馈神经网络。

（数学公式）

（多层前馈神经网络图片）

2.2.2 全连接层和全连接神经网络

前馈神经网络是最简单的神经网络类型，而全连接神经网络则是最简单的前馈神经网络类型。

2.2.2.1 全连接层

全连接层（Fully Connected layer）是指神经网络中的这一层的所有神经元都和上一层的所有神经元相连接，即上一层的所有神经元的输出都会影响到这一层的所有的神经元的输出。

2.2.2.2 全连接神经网络

全连接神经网络（fully connected neural network）是仅由全连接层组成的前馈神经网络。

2.2.3 损失函数

在深度学习中，损失函数（loss function）是用来描述神经网络预测结果与实际结果的差距的一个函数。它有着不同的形式，损失函数越小，代表着神经网络预测结果与实际结果越接近。在本论文中，我们用来表示损失函数。

常见的损失函数有以下几种：

1）损失（ loss）

损失又称为“平均偏差”（Mean Absolute Deviance, MAD）。

损失函数的数学表达式是：

其中代表神经网络预测结果的集合，代表样本真实结果的集合，表示样本集合中元素的个数。

2）损失（ loss）

损失又称为“MSE损失”（均方误差，Mean Squared Error）。

损失函数的数学表达式是：

其中代表神经网络预测结果的集合，代表样本真实结果的集合，表示样本集合中元素的个数。

2.2.4 全连接神经网络的训练方法

对于全连接神经网络来讲，它所有神经元中的权值在最初都是随机分配的。人工神经网络之所以称为“神经网络”是因为它有训练和学习的能力。简单地说，它的所有神经元中的权值是在学习的过程中不断变化的。

2.2.4.1 梯度

梯度（gradient）是表示函数上升（或下降）趋势的量。

考虑神经网络中的权值，那么针对函数，梯度是

梯度有着以下性质：

1）梯度的方向就是函数上升的方向

2）梯度的范数越大，函数上升地越快

2.2.4.2 梯度下降算法

梯度下降算法是一个迭代的算法。算法的目的是调整神经网络中的权重值，使这个权重值计算出的结果更贴近我们想要的结果。这个目的使用过让损失函数的计算结果不断变小实现的。开始时，算法随机指定一个权重值，当然这个权重值不是我们想要的。接下来时刻，我们对损失函数求导，得到梯度。这样我们就得到了损失函数下降的方向，通过计算就得到了新的权重值。在上式中，代表了学习率（learning rate），它的大小大部分情况下决定了学习效率的大小。在我们不断重复这个过程的时候，损失函数不断变小，也就是神经网络计算出的结果越来越接近我们想要的结果。这样就达到了训练、学习的目的。

2.3 循环神经网络

循环神经网络（Recurrent Neural Network, RNN）是神经网络中的一种。它是在自然语言处理领域最先被利用起来的。循环神经网络可以用来描述与时间有关的行为。循环神经网络的结构和全连接神经网络不同。在全连接神经网络中，样本之间相互独立，但是在循环神经网络中，样本之间可以相互影响，所以循环神经网络和全连接神经网络、前馈神经网络不同。循环神经网络更加适用于处理时间序列数据。最早利用循环神经网络成功的是手写识别。

2.3.1 循环神经网络的结构

图x-x是一个最简单的循环神经网络，它由一个输入层、一个隐藏层、一个输出层构成。其中表示输入层的值，它是一个向量；表示隐藏层的值，它也是一个向量；也是一个向量，它表示输出层的值；是输入层到隐藏层的权重矩阵；是隐藏层到输出层的权重矩阵。循环神经网络的隐藏层的值不仅仅取决于当前这一次的输入，还取决于上一次隐藏层的值。所以，也是一个权重矩阵，它表示隐藏层上一次的值作为隐藏层这一次的输入的权重。我们把图x-x展开，如图x-x。

（RNN图片）

（RNN展开图片）

循环神经网络在时刻接收到的输入后，计算出的隐藏层的值是，计算出的输出值是。重要的是，计算的值除了需要取决于，还需要取决于。这样我们就可以写出循环神经网络的计算公式：

1）隐藏层计算公式：。这一层是循环层。是输入层到隐藏层的权重矩阵；是隐藏层上一次的值作为隐藏层这一次输入的权重矩阵；是激活函数。

2）输出层计算公式：。这一层是全连接层，是输出层的权重矩阵，是激活函数。

所以我们得到：

2.3.2 循环神经网络的训练

循环神经网络的训练方法：BPTT（Back Propagation Trough Time）算法。首先，我们记

BPTT算法是针对循环神经网络中循环层的训练算法，它包含4个步骤：

1）前向计算每个神经元的输出值

2）反向计算每个神经元的误差项

它是损失函数对神经元中的中间计算结果的偏导数

3）计算每个神经元中权重的梯度

4）利用梯度下降算法更新权重

2.3.2.1 前向计算

利用上面的计算公式得到循环神经网络中循环层的所有。

2.3.2.2 误差项计算

首先，我们记，所以，因此，

其中，

上式中，代表中的个元素，表示对角矩阵。

所以，

上式描述了沿时间传递的规律，有了这个规律我们可以求的任意时刻的误差项：

2.3.2.3 权重梯度计算

接下来，我们就可以计算出权重矩阵在时刻的梯度。

我们考虑，

我们对权重求导，

所以我们得到下面的公式：

在上式中，表示时刻的权重矩阵，表示的第个元素，表示的第个元素。

而是所有时刻的和，所以，

同样我们可以用类似的方法得到权重矩阵的梯度，

和权重矩阵一样，权重矩阵的梯度也是所有时刻梯度之和，

2.3.2.4 循环神经网络的梯度爆炸问题

这种循环神经网络不能很好地处理较长序列。一个主要的原因是：循环神经网络在训练的过程中很容易发生梯度爆炸或梯度消失的问题，这导致训练时梯度不能在较长的序列中传递下去，使得这种循环神经网络无法捕捉到长距离的影响。

产生梯度爆炸或梯度消失的原因是：

上式中的定义为矩阵模的上界。因为上式是一个指数函数，所以如果很大的话，会导致误差项增长或缩小的非常快，这样就会导致相应的梯度爆炸或梯度消失的问题。

2.4 长短期记忆网络

长短期记忆网络（Long Short-Term Memory, LSTM）属于一种循环神经网络，LSTM适合预测和处理时间序列中延迟和间隔非常长的事件。LSTM的表现通常比上文提到的循环神经网络要好。LSTM区别与RNN之处在于它加入了一个用于“遗忘”的“处理器”。

2.4.1 长短期记忆网络的结构

长短期记忆网络思路也比较简单，原始循环神经网络的隐藏层只有一个状态，它对短期输入非常敏感。如果我们再增加一个量，用来保存长期的状态，那么，循环神经网络中的问题就迎刃而解了。所以LSTM网络中的LSTM神经元如图x-x。

（LSTM细胞图片）

记原始循环神经网络中的状态为，新增的状态为，称为单元状态（cell state）。我们把图x-x展开，如图x-x

（LSTM细胞展开的图片）

我们可以从图x-x中看出，在时刻，LSTM神经元的输入有三个：当前时刻神经网络的输入值、上一时刻LSTM神经元的输出值、上一时刻LSTM神经元的单元状态。LSTM神经元的输入有两个：当前时刻LSTM神经元的输出值、当前时刻LSTM神经元的单元状态。

在LSTM神经元中有三个控制开关，如图x-x所示：

1）负责控制是否继续保持长期状态。

2）负责控制把当前状态加入到长期状态中。

3）负责控制是否把长期状态作为当前时刻LSTM神经元的输出。

在LSTM神经元中，使用“门（gate）”的概念来实现这三个控制开关。门实际上是一层全连接层，它输入是一个向量，输出也是一个向量，并且元素值在0到1之间。门可以表示为：，其中是权重向量，是偏移项。为Sigmoid函数（激活函数的一种，上文提到过）。由于Sigmoid函数返回的值在区间上，所以门总是半开半闭的。

在LSTM神经元中，使用两个门来控制单元状态，一个是遗忘门（forget gate），另一个是输入门（input gate）。另外，LSTM神经元用输出门(output gate)来控制单元状态有多少可以输出到当前的输出值中。

1）遗忘门：，其中代表遗忘门的权重矩阵，代表遗忘门的偏移项，为两个向量的拼接。的维度是，其中，是单元状态的维度，是输出的维度，是输入的维度。我们可以把权重矩阵看作两个矩阵的拼接：，其中对应输入，它的维度为，对应输入，它的维度为。所以，

2）输入门：，其中代表输入门的权重矩阵，代表输入门的偏移项。

3）输出门：我们需要考虑当前输入的单元状态：它是根据上一次输出，和本次的输入来计算的，；当前时刻的单元状态：；长期记忆对当前输入的作用：。利用以上计算，输出门的数学表达式为。

2.4.2 长短期记忆网络的训练

长短期记忆网络的训练方法与循环神经网络类似，主要有下面三个步骤：

1）前向计算每个LSTM神经元的输出值。

2）反向计算每个LSTM神经元的误差项的值。

3）计算LSTM神经元中每个权重的梯度

4）利用梯度下降算法更新每个权重值

2.4.2.1 前向计算

对于LSTM来讲，前向计算就是计算的值

2.4.2.2 反向计算误差项的值

和循环神经网络不同，我们定义

这里我们假设误差项是损失函数对输出的导数。接下来我们记

，，，

下面我们计算

在上文中可以看出，都是的自变量，利用全导数公式可得：

根据前面的公式我们可以得到：

由此我们得到，

根据，可知：

这样我们就可以写出，

下面我们将误差项传递到上一层，假设当前为第层，定义层的误差项：

它代表损失函数对层加权输入的导数。第层的输入由公式计算，其中是第层的激活函数。

我们用全导数公式将上式化为：

2.4.2.3 权重矩阵梯度的计算

我们已经求了误差项，接下来很容易求出的梯度：

各个时刻的梯度加和，求得最终梯度：

偏移项的梯度：

各个时刻的梯度加和，求得最终梯度：

权重的梯度：

最后更新权重值，LSTM神经网络的训练就完成了。

2.6 神经网络训练的优化方法

2.6.1 梯度下降算法的缺陷

在上文中，我们利用来更新权值矩阵，这个算法有些明显的缺陷：

1）最基本的梯度下降算法很容易求得局部最小值或者陷入驻点。

2）最基本的梯度下降算法很容易在最优点附近震荡而无法求得最优点。

2.6.2 解决方法

针对第一种缺陷我们引入“动量”来解决，针对第二种缺陷我们通过调整学习率来解决。

2.6.2.1 动量

动量就是将历次梯度和最新的梯度做线性叠加。通常，动量的表达式可以是，其中，和是两个预设的常数，用来更新：

2.6.2.2 调整学习率

为了让循环开始时的权重矩阵变化尽可能大，到快接近最优点时权重矩阵变化尽可能小，我们可以在调整权重矩阵时使用不同的学习率，让学习率开始时很大，在多次循环之后逐渐减小。

2.6.2.3 Adam算法

Adam算法是一种随机优化的算法。在Adam算法中，实现了上文提及的两个方面，也补足了梯度下降算法的两个不足。下面是Adam算法的计算步骤：

1）初始化：初始化

2）计算：

其中，，一般为，一般为，一般为，一般为

第三章 模型构建

3.1 使用工具

3.1.1 Python

Python语言在1991年由Guido van Rossum创立，它是一个解释性的高级编程语言。Python语言极其强调代码的可读性以及代码的优美，在Python语言中，使用强制空格的方式划分模块。这种方法代替了C语言中的花括号、Matlab中的关键词等，使得代码可读性更高，更加优美。

因为Python语言是一个解释型语言，所以Python在运行时需要解释器。Python的解释器可以在很多平台上运行（几乎是所有平台），这使得Python可以在更广泛的范围内传播。如今Python语言已经是世界上常用语言之一。如Linux操作系统中就默认装有Python语言模块，在Linux中也有部分功能使用Python实现。

在本文中，我们使用Python3.5.2（Ubuntu中自带的Python版本），我们使用pip安装以下几个模块（下文也会提及）：

1）Numpy

2）PyTorch

3）Pandas

4）Matplotlib

3.1.2 Numpy

Numpy是Python的一个库。它支持大型的、多维的向量和矩阵的运算，同时也实现了很多数学函数，我们用起来特别方便。

在Numpy中比较核心的功能是“ndarray”，它实现了维数组的数据结构。它和Python中的list、tuple等不太相同的是：Numpy数组的数据必须是同一类型的，在Python数组中数据可以是不同类型的。

3.1.3 PyTorch

PyTorch是Python的一个库，它是一个开源的机器学习库。它最初是由Facebook的人工智能研究团队开发的。PyTorch的显著特点是可以利用GPU的快速运算来加速。

3.1.3.1 Tensor

在Python中，Tensor是一个最基本的单位，一切数据都存储在Tensor中，它类似于Numpy的ndarray。Tensor同样可以存储多维数组、矩阵、向量。Tensor的中文为“张量”，张量可以是维的。

3.1.3.2 PyTorch中的模块

1）Autograd模块

PyTorch使用自动微分（automatic differentiation）技术。在这个模块中使用一个记录器来记录函数的表现，然后进行计算梯度。这个模块在神经网络的训练中特别有用，因为它可以实现神经网络中的梯度下降法。

2）Optim模块

PyTorch中的Optim模块实现了很多神经网络中的优化算法，包括上文提到的Adam算法，我们在进行训练时，只需调用Optim模块，而不需要重新构建优化算法。

3）nn模块

nn模块中实现了现如今主流的神经网络模型，同样我们也可以利用nn模块来重新搭建一个我们想要的神经网络模型，nn模块非常好用且高效。

3.1.4 Pandas

Pandas是Python的一个库，它的作用是数据处理和数据分析，它同Numpy类似，都是用来处理数据的库。不同的是，Numpy更加注重数学运算，而Pandas更擅长处理时间序列，因为本文的数据是股票市场的数据，它是一个时间序列数据，所以我们将Pandas和Numpy结合来处理股票数据。

3.1.5 Matplotlib

Matplotlib是Python的一个库，它的功能是将数据以图表的形式展示出来。它结合Numpy可以轻松地画出股票市场的曲线图。它的图表形式与Matlab类似。虽然它是Python的一个库，但是它的使用风格与Matlab相似。

3.2 基于LSTM的神经网络模型构建

在构建神经网络前，我们需要设定一个窗口，它代表我们预测股票市场时间序列所基于的数据，如图x-x所示，我们想要预测某一天的股票收盘价，我们需要知道它前天的数据，这段数据就是我们要设定的窗口，我们称它为。

（window图片）

在神经网络中就是我们神经网络的输入的维度，即，在PyTorch中需要将input\_size设定为这个值。接下来，我们需要确定在LSTM中的输出的长度，在PyTorch中需要将hidden\_size设定为这个值。最后，我们需要确定PyTorch中的num\_layers，它代表循环层的层数，也就是我们使用多少个LSTM神经元。因为LSTM的输出的维度是hidden\_size，所以我们需要一个全连接层来将这个输出变为维度为的值。我们使用torch.nn中的Sequential。Sequential是一个容器，我们可以将神经元加入到其中，整合成一个神经网络。这里我们在Sequential中加入一个全连接层，使得数据通过全连接层后，变成我们想要的股价预测数据。

我们可以构建一个最基本的基于LSTM的神经网络，

（神经网络模型图片）

在这个神经网络中有一个LSTM神经元和一个普通神经元，它们分别在循环层和全连接层中。

（未完待续）

3.3 基于WT+SAE+LSTM的神经网络模型构建

3.4 数据处理

在我们获取到股票数据时，我们需要进行标准化（normalization）处理。目的是使得数据更加的正常或者说更加的规则。我们使用的标准化方法是z-score标准化方法：

其中，是标准化前的数据，是标准化后的数据，是数据整体的均值，是数据整体的标准差。

在我们把数据标准化之后，我们就要考虑如何把数据输入至神经网络中了。当我们把数据输入神经网络时，并不是一条一条的输入进入神经网络的，我们需要把数据重新编排，把数据的输入特征和标签分开（标签就是我们想要的数据的输出结果）。这里我们使用Pandas工具将数据进行重新编排。我们在预测第天的股票数据时，需要第天至第天的数据作为窗口。在Pandas中，我们需要先创建一个DataFrame，接下来我们需要设置这个DataFrame的列，其中，前条数据代表窗口，第条数据代表标签。

之后，我们把这个DataFrame转化为Numpy的Array，再转换为PyTorch中的Tensor。

最后，需要用到PyTorch中的两个模块：Dataset和DataLoader。它们都在torch.utils.data中。Dataset是一个抽象类，所以我们在使用Dataset时，需要继承这个类，然后重写Dataset类中的\_\_getitem\_\_()方法和\_\_len\_\_()方法，它们分别实现了取出数组中第个元素和求数组长度的功能。DataLoader类是用来加载数据的。然后返回一个迭代器，在训练时，我们只需要使用一个循环就可以完成一次训练。

第四章 股票数据的选取

4.1 数据来源

本文中实验的数据有两部分，第一部分是上证指数的数据，第二部分是个股股价的数据。本文用这两种数据来进行训练和预测，以及对算法的评估。这两部分数据都是从RESSET金融研究数据库获取的。RESSET金融研究数据库是为模型检验、投资研究、实证研究、学科与实验室建设等提供专业服务的数据库。包括股票、债券、基金、外汇、期货、黄金、研究报告、融资融券、宏观统计、行业统计、金融统计11大系列数据。

4.2 上证指数预测

上证指数指的是上海证券综合指数。它是用来反应上海证券交易所的所有上市股票（包括A股和B股）的价格增长或下跌情况的。

5.2.1 上证指数的计算方法

1）抽样：在上海证券交易所上市的股票中进行抽样，选出具有代表性的一批成分股。由于在上海证券交易所上市的股票种类很多，如果考虑所有股票的情况是非常复杂的，所以采用选取一些具有代表性的股票作为样本。

2）加权：按照股票价格或总金额进行加权，或采用不加权的方法，即平均。

3）计算程序：计算算数平均数或者是集合平均数

4.3 个股股价预测

个股股价的预测比上证指数更难做到，因为存在着多方因素的影响，比如政策、人们的心理、一些机构的操纵等等。我们把算法用在个股的预测上是为了对比上证指数的预测。

第五章 实验分析

5.1 参数设置

5.2 整体效果

5.3 评估指标

5.4 综合评价

5.5 几种解决方法的对比

致 谢

参考文献

外文资料原文

外文资料译文