证券研究报告—深度报告

金融工程

数量化投资

金融工程专题研究

2021 年 11 月 16 日

专题报告

深度学习时间序列预测综述

● 用于时间序列预测的深度学习框架

近年来包括股票交易数据在内的大量数据的涌现对于大数据分析提出了越来越多的需求，而在大数据分析中广泛使用的各种学习技术中，深度学习技术由于其出色的预测能力脱颖而出。

本文着重对于当前流行和常用于时间序列预测的深度学习框架进行了综述，着重从深度前馈神经网络、循环神经网络以及卷积神经网络三个大类对于深度学习框架进行了介绍和分析，并且介绍了循环神经网络和卷积神经网络的大量变体网络类型。

● 深度学习实践中的问题

对于深度学习网络在实践过程中可能遇到的问题，本文从框架实现、超参优化、硬件性能三个角度着重分析了工程实践过程中可能遇到的典型问题并给出了详细的分析与建议。

风险提示：本报告基于相关文献，不构成投资建议。

证券分析师：杨怡玲
电话：021-60875176
E-MAIL：yangyiling@guosen.com.cn
证券投资咨询执业资格证书编码：S0980521020001
证券分析师：张欣慰
电话：021-60933159
E-MAIL：zhangxinwei1@guosen.com.cn
证券投资咨询执业资格证书编码：S0980520060001

相关研究报告：
《学术文献研究系列第 22 期：温水煮青蛙：股价信息连续性与动量效应》——2021-11-09
《学术文献研究系列第 21 期：如何度量基金申赎对股票价格的影响》——2021-11-04
《学术文献研究系列第 20 期：如何看待价值因子的至暗时刻》——2021-10-26
《学术文献研究系列第 19 期：管理层与卖空投资者意见为何总是分歧？》——2021-10-19
《学术文献研究系列第 18 期：机构投资者的过度自信与 PEAD》——2021-10-13

内容目录

图表目录

文献来源

文献来源： Torres, José F., et al. "Deep Learning for Time Series Forecasting: A Survey." *Big Data* 9.1 (2021): 3-21..

文献亮点： 深度学习方法正被越来越多的用于时间序列预测。本文对于时间序列预测的各种深度学习框架以及实践过程中的很多工程问题都进行了深入而广泛的讨论，为基于深度学习的时间序列研究的框架选型和使用过程提供了丰富的指导作用。

引言

近年来大量数据的涌现对于大数据分析提出了越来越多的需求，而在大数据分析中广泛使用的各种学习技术中，深度学习技术由于其出色的预测能力脱颖而出。大部分深度学习中的层次化计算都可以通过 GPU 来并行计算，因此我们能够以较高的性能来训练大规模分布式的模型，并且模型复杂度和非线性程度越来越高。

深度学习目前在学界有大量的研究，监督学习和无监督学习都应用广泛。模式识别和分类任务是最先也是最大量使用深度学习方法的，并且在语音识别、文本挖掘、图像分析领域都取得了较大成功。然而，深度学习在回归问题上的应用正变得日益广泛，主要是因为在带有时间标签的数据上开发了越来越多的深度学习架构，这种情况我们一般称为时间序列预测。

时间序列是指按一定时间顺序上以一定的时间间隔收集的数据集合。虽然时间序列上的一些统计方法早在 1970 年代就开始大量研究，但是基于深度学习的模型当前能够获得更强的效果并且新的深度学习框架正在逐步被研究和创新出来。

本文的主要目的是对于时间序列预测领域中深度学习基础设施进行全面广泛的研究和梳理。另外，本文也会介绍当前已经成功应用深度学习方法的一些应用案例

问题定义

时间序列定义

一个时间序列是由一个按一定顺序排列并且在时间上观察得到的一段取值序列组成。虽然时间是连续的，但是时间序列样本的取值通常是以一定固定间隔采样得到。虽然这个定义对于大多数情况是成立的，但是并不是所有的时间序列都能以此定义来建模，主要是因为：

1. 数据收集时不可靠肯定导致数据缺失。为了处理缺失值，通常广泛使用的方式是填充缺失信息或者丢弃样本。
2. 异常数据也是经常遇到的问题，通常采用一些基于稳健统计的方法来剔除这些异常数据或者将数据调整后加入模型。
3. 当以无规律的时间间隔收集数据时，这些数据被称为不均匀间隔的时间序列或者是数据流。

这些问题中有一些能够在后续使用的模型中自动处理完成，但是如果数据采集

无固定规律，这种情况就需要在模型中进行显式处理。时间序列的预处理方法不属于本文讨论范围，在此不做详细介绍。

时间序列构成

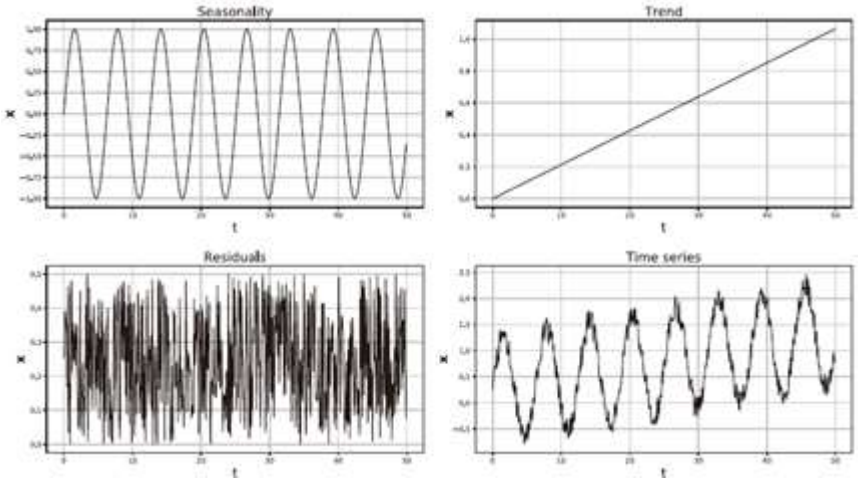
时间序列通常由三个重要的属性来刻画，趋势、季节性和无规律部分即残差。

- 趋势：这是一段时间序列不考虑季节性和残差的基本运动过程，最常见的趋势类型包括线性、指数型以及抛物线型。
- 季节性：这个属性衡量了以特定时间间隔重复的变化特征，它综合表现出在时间、幅度和方向上的稳定性。季节性可能由于气候、经济周期或者节假日等因素而产生。
- 残差：当趋势和季节性被剥离后，剩下的就是残差。有些残差可能足够高而掩盖了趋势和季节性，这时的残差我们通常称其为离群点，并且通常会使用一些稳健统计的方法来处理这些离群点。这些波动可能有多种来源，这使得预测几乎不可能。然而，如果有任务机会可以检测或建模这些波动的来源，他们可以被认为是趋势变化的前兆。

时间序列是这三个组成部分的集合。现实世界的时间序列呈现出有意义的不规则分量并且不是平稳的（均值和方差随时间变化不是恒定的），这使该分量成为最具挑战性的建模分量。 为此，对它们进行准确的预测是极其困难的，许多经典的预测方法试图将目标时间序列分解为这三个分量，并分别对它们进行预测。每种技术的有效性是根据其预测每个特定的分量的能力来评估的。 基于数据挖掘的技术已被证明对于这些分量具有非常强大的分析能力。

时间序列可以用图形表示，x 轴标识时间，而 y 轴标识时间戳 x_t 记录的值。 这种表示允许对系列中最突出的特征进行视觉检测，例如振荡幅度、季节性和周期或异常数据或离群点的存在。图 1 描绘了一个时间序列，它使用具有线性季节性的加性模型，随着时间的推移具有恒定的频率和幅度，由函数 sin(x)表示；线性趋势其随时间的变化始终保持相同的数量，由函数 0.0213x 表示；残差由区间 [0, 0.1] 中的随机数表示。

图 1：一个包含趋势、季节性和残差的时间序列



资料来源：Big Data，国信证券经济研究所整理

数学形式

时间序列模型可以是单变量（一个时间相关变量）或多变量（多个时间相关变

量）。 尽管单变量和多变量系统之间的模型可能有很大不同，但大多数深度学习模型都可以模糊地处理它们。

一方面，用$y = y(t-L), \dots, y(t-1), y(t), y(t+1), \dots, y(t+h)$ 表示一个历史数据中具有 L 个值的单变量时间序列，其中每个$y(t-i), i = 0, \dots, L$ 都表示变量y在时间$t-i$的记录值。预测过程包括估计$y(t+1)$的值，用$\hat{y}(t+1)$表示，目的是最小化误差，误差通常表示为 $y(t+1) - \hat{y}(t+1)$。 当预测范围 h 大于 1 时，也可以进行这种预测，也就是说，当目标是预测 $y(t)$ 之后的h个值时，即$y(t+i), i = 1, \dots, h$。 在这种情况下，当函数 $\Sigma_{i=1}^{h}(y(t+i) - \hat{y}(t+i))$ 最小化时，可以达到最佳预测。

另一方面，多元时间序列可以用矩阵形式表示如下：

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_T \end{pmatrix} = \begin{pmatrix} y_1(t-L) & \dots & y_1(t-1) & y_1(t) & y_1(t+1) & \dots & y_1(t+h) \\ y_2(t-L) & \dots & y_2(t-1) & y_2(t) & y_2(t+1) & \dots & y_2(t+h) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ y_n(t-L) & \dots & y_n(t-1) & y_n(t) & y_n(t+1) & \dots & y_n(t+h) \end{pmatrix}, \quad (1)$$

其中 $y_i(t-m)$ 表示时间序列集，$i = \{1,2, \dots, n\}, m = \{1,2, \dots, L\}$ 表示历史数据和当前样本，而$m = \{-1, -2, \dots, -h\}$ 表示未来的h 个值。通常，有一个目标时间序列（待预测的），其余的表示为独立的时间序列。

短期和长期时间序列预测

另一个关键问题是时间序列的长度。根据样本数量，可以定义长时间序列或短时间序列。众所周知，Box-Jenkins 的模型不适用于长时间序列，主要是由于参数优化的耗时过程和信息的包含，这些信息对当前样本的建模不再有用。

如何处理这些问题与模型的目的密切相关。可以使用灵活的非参数模型，但这仍然假设模型结构将适用于整个数据周期，这并不一定总是正确的。更好的方法包括允许模型随时间变化。这可以通过使用时变参数调整参数模型或使用基于时间的内核调整非参数模型来完成。但是，如果目标只是预测一些观察结果，则使用最新样本拟合模型并将长时间序列转换为短时间序列会更简单。

尽管最近发布了使用分布式 ARIMA 模型的初步方法，但使用经典预测方法处理此类时间序列仍然具有挑战性。 然而，近年来已经发布了许多适用于处理超长时间序列或大数据时间序列的机器学习算法。这些模型利用机器集群或 GPU 来克服前面段落中描述的限制。

深度学习模型可以以可扩展的方式处理时间序列并提供准确的预测。 集成学习也可用于预测大数据时间序列，甚至基于非常成熟的方法（如最近邻或模式序列相似性）的方法。

深度学习架构

本节提供了大数据环境中用于时间序列预测的深度学习的理论导览，首先介绍一下文献中最常用的用于预测时间序列的架构，然后介绍最前沿的大数据的深度学习工作和框架分析。

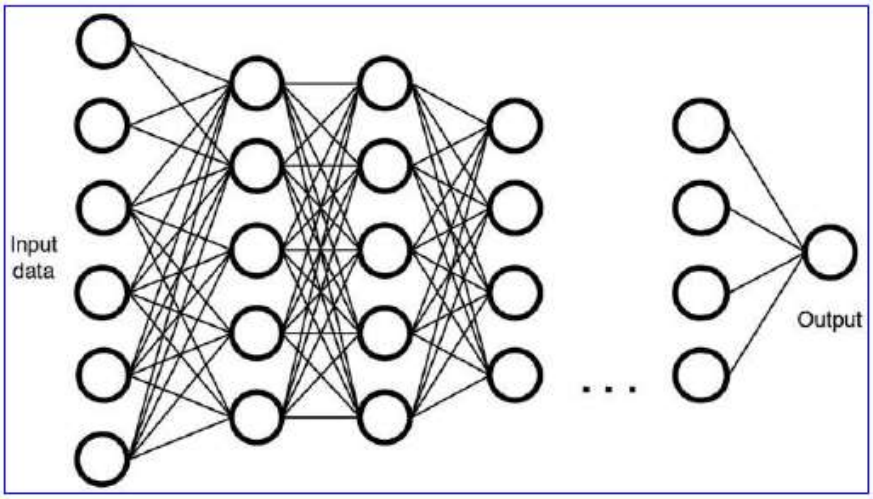
深度前馈神经网络

深度前馈神经网络 (DFFNN)，也称为多层感知器，是由于单层神经网络无法学习某些功能而产生的。 DFFNN 的架构由输入层、输出层和不同的隐藏层组成，

如图 2 所示。此外，每个隐藏层都有一定数量的神经元需要确定。

两个连续层的神经元之间的关系通过权重建模，权重是在网络训练阶段计算的。

图 2：深度前馈神经网络



资料来源：Big Data，国信证券经济研究所整理

特别是，权重是通过梯度下降优化方法最小化成本函数来计算的。 然后，使用反向传播算法计算代价函数的梯度。一旦计算出权重，网络输出神经元的值就通过使用由以下等式定义的前馈过程获得:

$$a^l = g(W_a^l a^{l-1} + b_a^l)$$

其中 a^l 是第 l 层的激活值，即第 l 层神经元的值组成的向量，W_a^l 和 b_a^l 是第 l 层对应的权重和偏差，g 是激活函数。因此，a^l 的值是通过使用 $l-1$ 层的激活值 a^{l-1} 作为输入来计算的。在时间序列预测中，除了输出层获得预测值外，校正线性单元函数通常用作所有层的激活函数，一般使用双曲正切函数（tanh）。

对于所有网络架构，必须提前选择一些超参数的值。这些超参数，如层数和神经元数量定义了网络架构，而其他超参数如学习率、动量、迭代次数或小批量大小等，对梯度下降方法的收敛性有很大影响。这些超参数的最佳选择很重要，因为这些值极大地影响了网络获得的预测结果。 超参数将在超参数优化部分更详细地讨论。

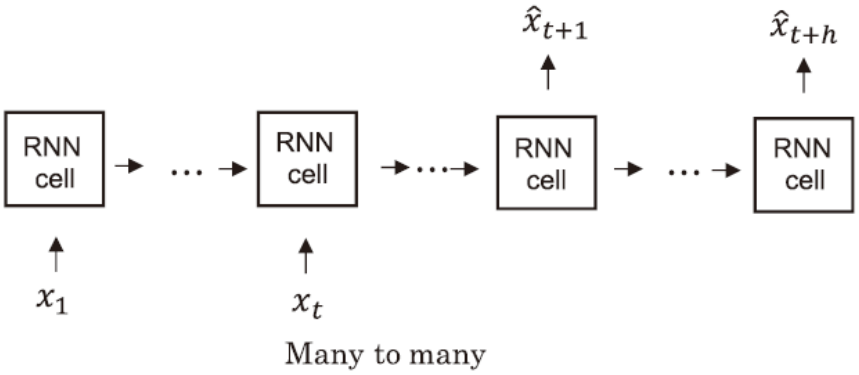
循环神经网络

循环神经网络 (RNN) 专门用于处理序列数据，例如与机器翻译相关的问题中的单词序列、语音识别中的音频数据或预测问题中的时间序列。所有这些问题都呈现出一个共同特征，即数据之间存在时间依赖性。传统的前馈神经网络无法考虑这些依赖性，而 RNN 正是为了解决这个问题而出现的。因此，RNN 架构中的输入数据既是过去数据，也是当前数据。有不同类型的架构，取决于网络中数据输入和输出的数量，例如一对一（一输入一输出）、一对多（一输入多输出）、多对一（多输入和一个输出），以及多对多（多输入和输出）。最常见的 RNN 是多对一的分类问题，或多对多的机器翻译或时间序列预测。另外，对于时间序列的情况，输入数据序列的长度通常与输出数据序列的大小不同，输出

数据序列的大小通常是要预测的样本数。解决时间序列预测的基本 RNN 架构如图 3 所示。x_i 和 $\hat{x}_i$是时间序列在时间 i 的实际值和预测值，h 是要预测的样本数，称为预测范围。

用于时间序列预测的最广泛使用的 RNN 将在后面简要介绍。

图 3：循环神经网络



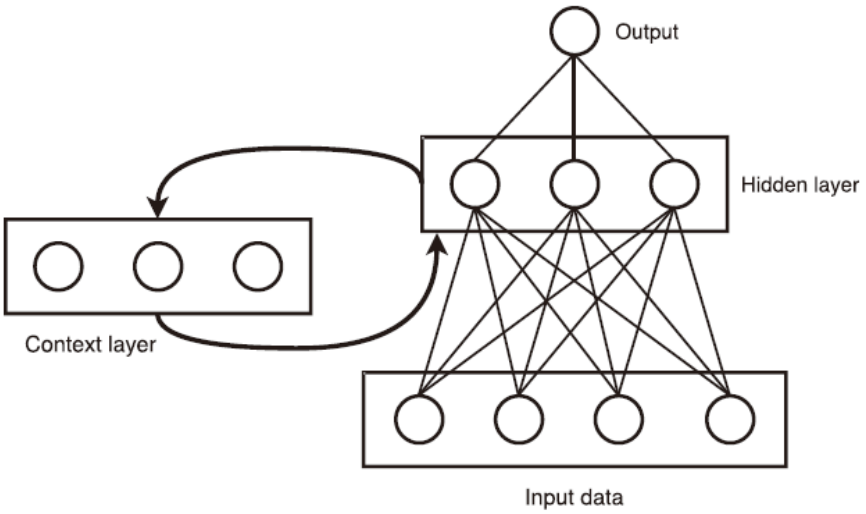
资料来源：Big Data，国信证券经济研究所整理

Elman RNN: Elman 神经网络（ENN）是第一个 RNN，它结合了隐藏单元的 t 状态来对数据序列进行预测。ENN 由经典的单层前馈网络组成，但隐藏层连接到一个新的层，称为上下文层，使用固定权重 1，如图 4 所示。这个上下文层的神经元的主要功能是保存隐藏层神经元激活值的副本。模型定义为：

$$a_t = g(W_a x_t + U_a a_{t-1} + b_a)$$

其中 a_t 是隐藏层中 t 状态的神经元值，x_t 是当前输入，a_{t-1} 是上下文隐藏单元中保存的信息，W_a, U_a, b_a 是权重和偏差，g 是激活函数。

图 4：ENN



资料来源：Big Data，国信证券经济研究所整理

长短期记忆网络：标准的基本 RNN 会遇到梯度消失的问题，梯度随着层数的增加而降低。事实上，对于具有大量层数的深度 RNN，梯度实际上为零，从而

阻止了网络的学习。出于这个原因，这些网络具有短期记忆，并且在处理需要记住完整序列中包含的所有信息的长序列时并不能获得好的结果。长短期记忆(LSTM) 循环网络的出现是为了解决梯度消失问题。为此，LSTM 使用三个门来保留长期存在的相关信息并丢弃不相关的信息。这些门是 Γ^f 遗忘门、Γ^u 更新门和 Γ^0 输出门。 Γ^f 决定应该丢弃或保存哪些信息。接近 0 的值意味着过去的信息被遗忘，而接近 1 的值意味着它仍然存在。 Γ^u 决定使用什么新信息 $\tilde{c}_t$ 来更新 c_t 内存状态。因此，c_t 是同时使用 Γ^f 和 Γ^u 来更新。最后，Γ^0 决定哪个输出值将作为下一个隐藏单元的输入。

a_{t-1}前一个隐藏单元的信息和 x_t 当前输入的信息通过 σ sigmoid 激活函数计算所有门值，并通过 tanh 激活函数计算 $\tilde{c}_t$ 新信息，这些信息将被用于更新。定义 LSTM 单元的方程是:

$$\tilde{c}_t = \tanh(W_c[a_{t-1}, x_t] + b_c)$$

$$\Gamma^u = \sigma(W_u[a_{t-1}, x_t] + b_u)$$

$$\Gamma^f = \sigma(W_f[a_{t-1}, x_t] + b_f)$$

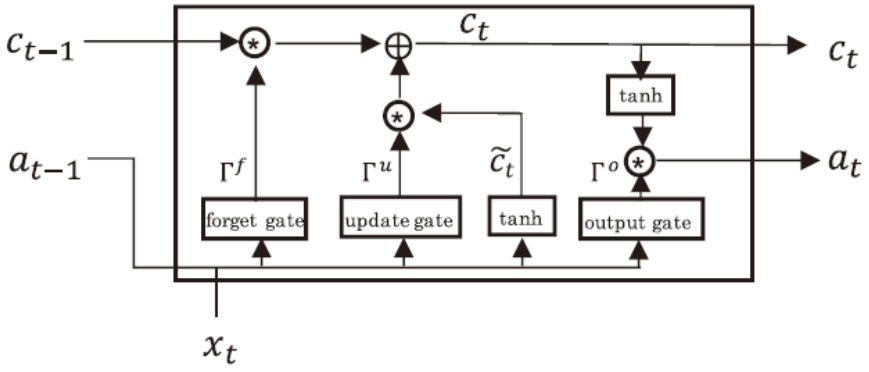
$$\Gamma^0 = \sigma(W_0[a_{t-1}, x_t] + b_0)$$

$$c_t = \Gamma^u \times \tilde{c}_t + \Gamma^f \times c_{t-1}$$

$$a_t = \Gamma^0 \times \tanh(c_t)$$

其中W_u, W_f, W_0，以及 b_u, b_f, b_0 是权重和偏差,分别控制 $\Gamma^u, \Gamma^f, \Gamma^0$ 门的行为，W_C 和 b_c 是 $\tilde{c}_t$ 存储单元候选者的权重和偏差。图 5 显示了 LSTM 循环网络中一个隐藏单元如何工作。× 和 + 运算符表示逐元素向量乘法和求和。

图 5: LSTM 中的隐藏单元



资料来源：Big Data，国信证券经济研究所整理

门控循环单元: 带有门控循环单元(GRU)的循环网络是 LSTM 等长期记忆网络，但由于 LSTM 网络的高计算成本，它们在 2014 年作为 LSTM 的简化而出现。GRU 是研究人员所关注的最常用的版本之一，并发现它对许多不同的问题都具有鲁棒性和实用性。在 RNN 中使用门可以改进对非常长距离的依赖关系的捕获，从而使 RNN 更加有效。LSTM 更强大、更有效，因为它有三个门而不是两个，但 GRU 是一个更简单的模型，计算速度更快，因为它只有两个门，Γ^u 更新门和 Γ^r 相关门，如图 6 所示。Γ^u 门将使用 $\tilde{c}_t$ 内存状态候选来决定是否更新 c_t 内存状态。Γ^r 门决定如何使用 c_{t-1} 来计算 c_t 的下一个候选者，即 $\tilde{c}_t$。

GRU 由以下等式定义:

$$\Gamma^u = \sigma(W_u[c_{t-1}, x_t] + b_u)$$

$$\Gamma^r = \sigma(W_r[c_{t-1}, x_t] + b_r)$$

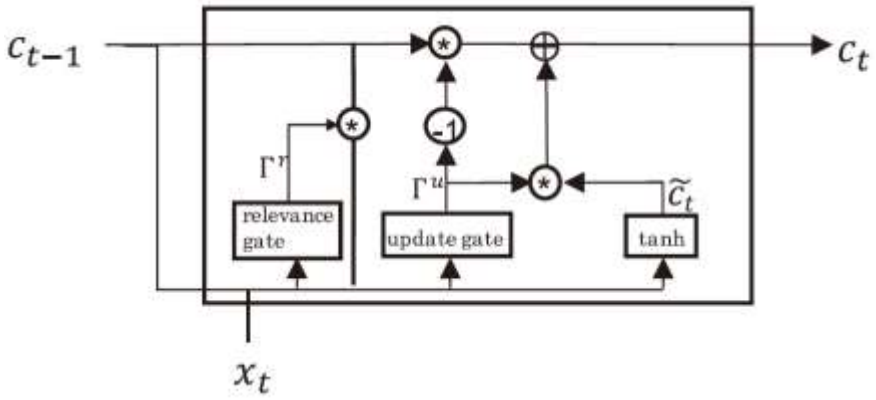
$$\tilde{c}_t = \tanh(W_c[\Gamma^r \times c_{t-1}, x_t] + b_c)$$

$$c_t = \Gamma^u \times \tilde{c}_t + (1 - \Gamma^u) \times c_{t-1}$$

$$a_t = c_t$$

其中 W_u, W_r 以及 b_u, b_r 分别是控制 Γ^u, Γ^r 门行为的权重和偏差，而 W_c, b_c 是 $\tilde{c}_t$ 存储单元候选者的权重和偏差。

图 6: GRU 中的隐藏单元



资料来源：Big Data，国信证券经济研究所整理

双向 RNN: 例如在自然语言处理(NLP)领域存在一些问题，在给定时刻预测数据序列的值时，需要该时刻前后的序列信息。 双向循环神经网络(BRNN)解决了这个问题。BRNN 的主要缺点是在进行预测之前需要整个数据序列。标准网络使用单向前馈过程计算隐藏单元的激活值。然而，在 BRNN 中，预测使用来自过去的信息以及来自现在和未来的信息作为输入，使用前向和后向处理。

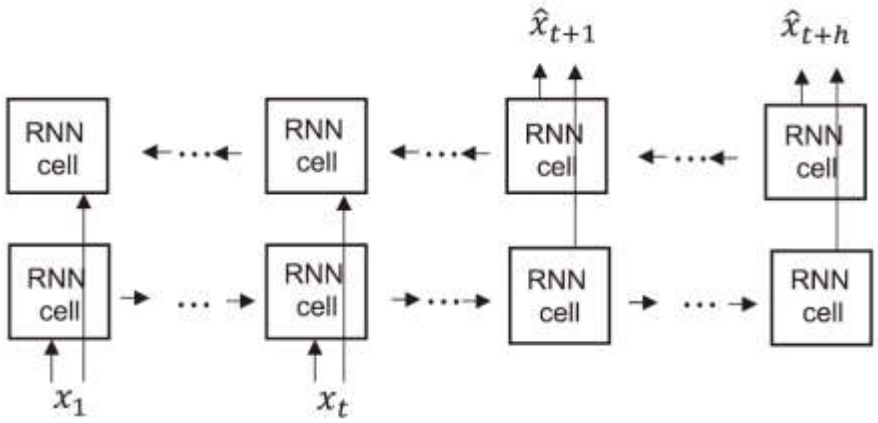
因此，时间 t 的预测 $\hat{x}_t$ 是通过使用 g 激活函数获得的，该函数同时在时间 t 进行前向和后向激活取值上应用对应的权重。

$$\hat{x}_t = g(W_x[a_t^f, a_t^b] + b_x)$$

其中 W_x, b_x 为权重和偏差，a_t^f, a_t^b 是分别通过前向和后向处理过程中隐藏层计算得到的激活值，g 是激活函数。

图 7 展示了 BRNN 的基本架构。 一个 BRNN 可以被看作是两个 RNN，其中不同的隐藏单元有两个值，一个由前向计算，另一个由后向计算。此外，BRNN 单元可以是标准 RNN 单元或 GRU 或 LSTM 单元。事实上，带有 LSTM 单元的 BRNN 通常用于很多 NLP 问题。

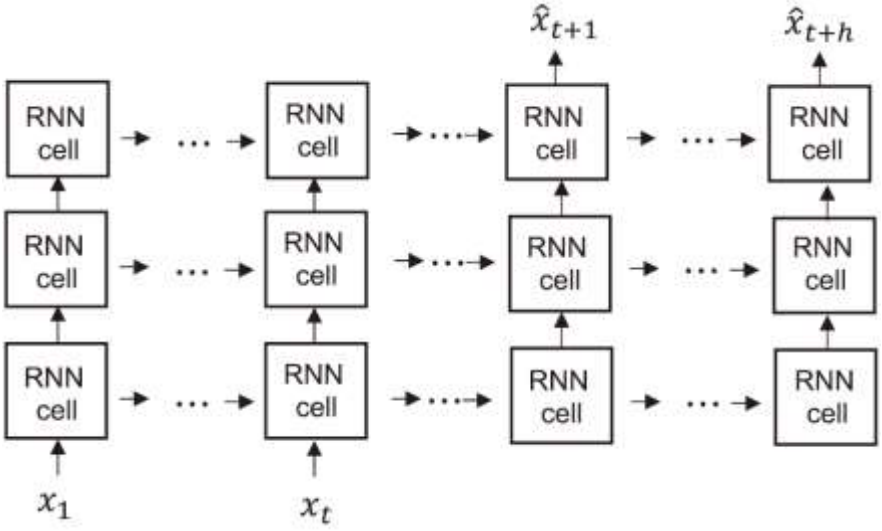
图 7：BRNN 基本架构



资料来源：Big Data，国信证券经济研究所整理

深度循环神经网络：深度循环神经网络 (DRNN) 可以被视为具有不止一层的 RNN，也称为堆叠 RNN。隐藏单元可以是标准的 RNN、GRU 或 LSTM 单元，并且可以是单向或双向的，如前几节所述。 图 8 说明了具有三层的 DRNN 的架构。

图 8：DRNN 基本架构



资料来源：Big Data，国信证券经济研究所整理

一般来说，DRNN 在时间序列预测方面效果很好，但是当使用很长的数据序列作为输入时，其性能会下降。为了解决这个问题，可以将注意力机制纳入模型，这是深度学习中最强大的思想之一。注意模型允许神经网络在生成输出时只注意输入数据序列的一部分。这种注意力是通过使用权重建模的，权重由单层前馈神经网络计算。

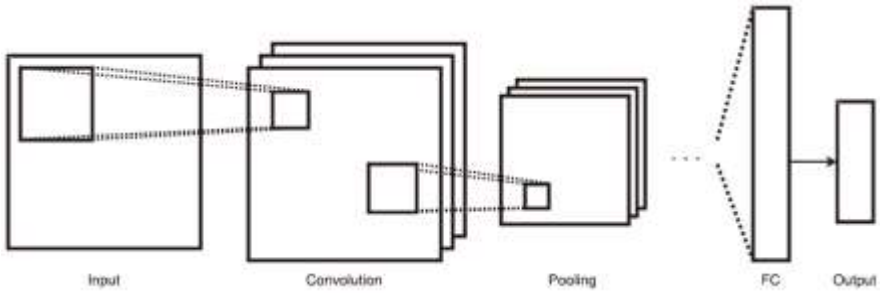
卷积神经网络

卷积神经网络 (CNN) 由 Fukushima 提出，是图像处理和计算机视觉中最常见的架构之一。CNN 具有三种层：卷积层、池化层和全连接层。卷积层的主要任务是从数据输入中学习特征。为此，通过使用矩阵之间的卷积运算将预定义大

小的过滤器应用于数据。卷积是所有元素乘积的总和。池化减少了输入的大小，加快了计算速度并防止了过拟合。最流行的池化方法是平均值和最大池化，它们分别使用平均值或最大值来汇总值。

一旦卷积层提取了特征，就可以使用全连接层（也称为密集层）进行预测，如 DFFNN。最后这些全连接层的输入数据是卷积层和池化层产生的扁平特征。图 9 描绘了 CNN 的整体架构。

图 9：CNN 基本架构



资料来源：Big Data，国信证券经济研究所整理

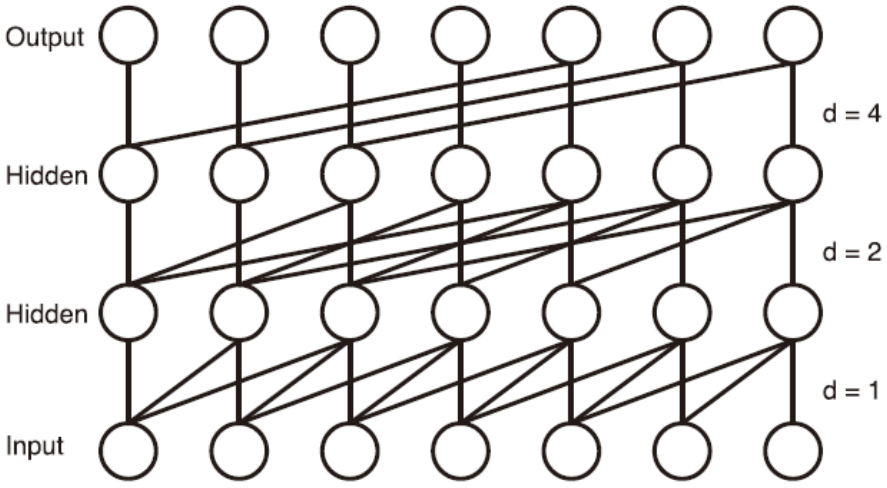
最近，一种称为时间卷积网络(TCN)的 CNN 变体出现，在执行时间和内存要求方面与 DRNN 形成了有力竞争。TCN 与 DFFNN 具有相同的架构，但每层的激活值是通过使用前一层的早期值计算的。膨胀卷积用于选择上一层神经元的哪些值将对下一层神经元的值有贡献。因此，这种扩张的卷积操作同时捕获了局部和时间信息。

膨胀卷积 F_d 是一个定义如下的函数：

$$F_d(x) = \Sigma_{i=0}^{K-1} f(i) \cdot x_{t-d \cdot i}$$

其中 d 是扩张因子参数， f 是大小为 K 的过滤器。

图 10：TCN 基本架构



资料来源：Big Data，国信证券经济研究所整理

图 10 展示了通过对每层使用大小为 3 的滤波器和 1、2 和 4 的扩张因子来应用膨胀卷积时 TCNN 的架构。此外，当使用更深更大的 TCN 来实现进一步的

稳定性时，除了卷积层之外，还需要使用通用残差模块。这些通用残差块包括在应用激活函数之前将数据输入添加到输出中。那么，TCN 模型可以定义如下：

$$a_t^l = g(W_a^l F_a(a_t^{l-1}) + b_a^l + a_t^{l-1})$$

其中 $F_a(\cdot)$ 是膨胀卷积，a_t^l 是时刻 t 第 l 层的神经元取值，W_a^l, b_a^l 是第 l 层对应的权重和偏差，g 是激活函数。

深度学习实践

实现

多层感知器的实现比较简单。然而，深度学习模型更为复杂，其实施需要高水平的技术专长和大量的实施时间。为了更轻松地实施并减少设计和训练模型所需的时间，一些公司将工作重点放在开发允许实施、训练和使用深度学习模型的框架上。

深度学习框架的主要思想是提供一个接口，提供模型的实现而不必过多关注它们背后的数学复杂性。学术文献中有几种可用的框架。选择哪个框架主要取决于几个重要因素，例如可以实现的架构类型、对分布式编程环境的支持，或者它是否可以在 GPU 上运行。从这个意义上讲，下图总结了学术文献中使用最广泛的框架，其中术语 all 包括 DFFNN、CNN、TCN、RNN、LSTM、GRU 或 BRNN 架构，CPU 是中央处理单元。

图 11：深度学习框架

Framework	Core language	Available interfaces	Architecture	Distributed	CPU | GPU
TensorFlow[40]	C++	Python, JavaScript, C++, Java, Go, C#, Julia	All	✓	✓|✓
H2O[42]	Java	Python, R, Scala, REST	DFFNN	✓	✓|✓
DI4J[41]	Java	Python, Scala, Clojure, Kotlin, C, C++	All	✓	✓|✓
PyTorch[43]	Lua	Python, C, C++	All	✓	✓|✓
Caffe[43]	C++	Python, MATLAB	CNN	✗	✓|✓
Neon[44]	Python	Python	All	✗	✓|✓
Chainer[45]	Python	Python	All	✓	✓|✓
Theano[46]	Python	Python	All	✗	✓|✓
MXNet[47]	Python	Python, Scala, Julia, Clojure, Java, C++, R, Perl	All	✓	✓|✓
ONNX[48]	Python	Python	CNN, DFFNN	✗	✓|✗
PaddlePaddle	Python	Python	CNN	✓	✓|✓
CNTK[49]	C++	Python, C++, C#	DFFNN, CNN, RNN	✓	✓|✓

资料来源：Big Data，国信证券经济研究所整理

上图显示用于开发深度学习模型的主要编程语言是 Python。此外，大多数框架都支持分布式执行和 GPU 的使用。尽管这里的框架有助于模型的开发，但其中一些框架需要实现额外较多的代码才能获得完整的实现。为此，开发者开发了基于框架核心的高级库，例如 Keras、Sonnet、Swift、Gluon，使上层编程更加容易。使用高级库的主要优点是，除了便于实现之外，语法还可以重用于另一个基础框架，然而可能缺乏灵活性。

超参优化

框架和高级库的结合极大地方便了模型的实现。然而，有一个重要的研究空白：模型优化。这种优化将决定模型的好坏，必须根据其超参数的调整来进行。在深度学习中有两种类型的超参数：模型参数和优化参数。必须在模型定义中调整模型参数以获得最佳性能。在模型的训练阶段使用数据集调整优化参数。一些最相关的超参数在下图中按网络架构进行了描述和分类。

图 12：深度学习中的超参

Hyper-parameter	Architectures	Description
Optimizer	All	Algorithm used to update the weights of each layer after each iteration.[53]
Learning rate	All	It determines the size of the step at each iteration of the optimization method.[54]
Number of epochs	All	Number of passes made in the whole training set.[55]
Batch size	All	Number of sub-samples that the network uses to update the weights.[56]
Hidden layers	All	It determines the depth of the neural network.[57]
Activation function	All	Introduces nonlinearity in the model, which allows the extraction of more complex knowledge.[58]
Momentum	All	It prevents oscillations in the convergence of the method.[59]
Weight initialization	All	It prevents the explosion or vanishing of the activation in the layers.[60]
Dropout	All	It eliminates certain connections between neurons in each iteration. It is used to prevent over-fitting.[61]
L1/L2 Regularization	All	It prevents over-fitting, stopping weights that are too high so that the model does not depend on a single feature.[62]
Units	RNN, DFFNN	It determines the level of knowledge that is extracted by each layer. It is highly dependent on the size of the data used.[57]
Kernel/filter	CNN	Matrix that moves over the input data. It allows the extraction of characteristics.[64]
Stride	CNN	The number of pixels that move over the input matrix for each filter.[64]
Padding	CNN	Number of null samples added to a dataset when it is processed by the kernel.[63]
Number of channels	CNN	Depth of the matrices involved in the convolutions.[65]
Pooling	CNN	It allows to reduce the number of parameters and calculations in the network.[57]
nb_stacks	TCN	Number of stacks of residual blocks.
Dilations	TCN	A deep stack of dilated convolutions to capture long-range temporal patterns.

资料来源：Big Data，国信证券经济研究所整理

超参数的数量将取决于要使用的网络架构。此外，每一项的价值都会受到问题特征和数据的影响。这使得优化模型的任务成为研究界的挑战。此外，考虑到参数可以推断出大量可能的组合。为此，各种元启发式和优化策略得以应用。根据文献，有几种策略可以优化深度学习模型的一组超参数，如下图所示。

图 13：参数寻优策略

Strategy	Deep learning	Cost	Search space
Trial-error	✗	Low	Low
Grid	✗	High	High
Random	✓	Medium	High
Probabilistic	✓	Medium	Medium-driven

资料来源：Big Data，国信证券经济研究所整理

因此，超参数优化方法可以分为四大块：

1. 试错：这种优化方法基于手动调整每个超参数。因此，该方法意味着高时间投资，具有相对较低的计算成本和低搜索空间，因为它需要用户在每次运行完成时手动修改值的动作。由于在深度学习中有大量的超参数，并且它们可以设置的值是无限的，因此不建议使用这种优化方法。

2. 网格：网格方法探索一组已建立的超参数的不同可能组合。这种方法覆盖了很高的搜索空间，尽管它具有与之相关的高计算成本，这使得这种方法无法应用于深度学习，更不用说在大数据环境中了。

3. 随机：随机搜索允许覆盖高搜索空间，因为可以生成无限的超参数组合。在这类方法中，我们可以区分完全随机或引导搜索策略，例如基于元启发式的搜索策略。此类搜索算法包括遗传算法、粒子群优化或增强拓扑算法的神经进化等。广泛的搜索范围加上该搜索策略所涉及的中等成本，使其成为优化深度学习模型的最佳方法之一。

4. 概率：这种优化方法跟踪每次评估的结果。这些评估用于生成概率模型，为不同的超参数分配值。使用概率方法优化超参数的最常见算法是基于贝叶斯方法的算法。

有许多库用于以自动化方式优化超参数。 然而，很少有专门为深度学习模型超

参数优化设计的。下图总结了一组用于深度学习模型超参数优化的库，按搜索策略分类。

图 14：超参优化库列表

Library	Search strategy	Distributed	Language	Framework
Elephas	Random, Probabilistic	Yes	Python	Keras
Hyperas	Random, Probabilistic	Yes	Python	Keras
Hyperopt[74]	Random, Probabilistic	Yes	Python	—
Dlopt[75]	Random	No	Python	Keras
Talos[76]	Grid, Random	Yes	Python	keras
Keras-tuner	Random	Yes	Python	Keras
H_2O[80]	Grid, Random	Yes	Python, R	H_2O
BoTorch[77]	Probabilistic	Yes	Python	PyTorch
HPOLib[78]	Probabilistic	—	Python	—

资料来源：Big Data，国信证券经济研究所整理

硬件性能

研究人员必须做出的最重要的决定之一是决定确保深度学习算法能够找到准确模型所需的物理资源。因此，鉴于对更好、更复杂的硬件的需求不断增加，本节概述了通常用于深度学习环境的不同硬件基础设施。

虽然可以使用 CPU 来执行深度学习算法，但密集的计算需求通常会导致 CPU 物理资源不足（线性架构）。因此，深度学习框架通常使用三种不同的硬件架构来挖掘信息：GPU、张量处理单元 (TPU) 和智能处理单元 (IPU)。

GPU 是分配在 CPU 中的协处理器，专门设计用于处理计算环境中的图形。GPU 可以拥有比 CPU 多数百甚至数千个内核，但运行速度较低。 GPU 以单指令、多数据架构实现高数据并行性，在当前人工智能领域发挥着重要作用，应用范围广泛。

第一代 TPU 于 2016 年在 Google I/O 大会上推出，它们专门设计用于运行已经训练好的神经网络。TPU 是专门为机器学习构建的定制专用集成电路。与 GPU（自 2016 年以来经常用于相同的任务）相比，TPU 被隐式地设计用于大量降低精度的计算（例如，从 8 位精度开始）和缺乏用于光栅化/纹理映射的硬件。该术语是为专为 Google 的 TensorFlow 框架设计的特定芯片而创造的。一般来说，与在普通 CPU 或 GPU 上执行的计算相比，TPU 的准确性较低，但对于它们必须执行的计算来说已经足够了（单个 TPU 每天可以处理超过 1 亿张图片）。此外，TPU 针对大批量和 CNN 进行了高度优化，并具有最高的训练吞吐量。

IPU 与今天的 CPU 和 GPU 处理器完全不同。它是一种高度灵活、易于使用的并行处理器，经过全新设计，可在当前机器学习模型上提供最先进的性能。但更重要的是，IPU 旨在实现新出现的机器智能工作负载。 IPU 在训练和推理的小批量上提供更好的算术效率，这导致训练中模型收敛速度更快，模型泛化能力更好，能够并行化更多 IPU 处理器以减少给定批量大小的训练时间，并且还以更低的推理延迟提供更高的吞吐量。另一个有趣的特性是与 GPU 或 TPU 相比功耗更低（最多降低 20%）。

结论

深度学习已被证明是解决处理大数据的复杂问题的最强大的机器学习技术之一。目前通过智能设备或者金融市场产生的数据大多是时间序列，对其预测是几乎所有研究领域中最常见和重要的问题之一。因此，本文对应用于时间序列预测的深度学习技术进行了综述。本文描述了过去几年最常用的时间序列数据深度学习架构，并且对于深度学习实践过程中的多个工程问题进行了深入探讨。

国信证券投资评级

类别	级别	定义
股票投资评级	买入	预计 6 个月内，股价表现优于市场指数 20%以上
	增持	预计 6 个月内，股价表现优于市场指数 10%-20%之间
	中性	预计 6 个月内，股价表现介于市场指数 ±10%之间
	卖出	预计 6 个月内，股价表现弱于市场指数 10%以上
行业投资评级	超配	预计 6 个月内，行业指数表现优于市场指数 10%以上
	中性	预计 6 个月内，行业指数表现介于市场指数 ±10%之间
	低配	预计 6 个月内，行业指数表现弱于市场指数 10%以上

分析师承诺

作者保证报告所采用的数据均来自合规渠道，分析逻辑基于本人的职业理解，通过合理判断并得出结论，力求客观、公正，结论不受任何第三方的授意、影响，特此声明。

风险提示

证券投资咨询业务的说明

本公司具备中国证监会核准的证券投资咨询业务资格。证券投资咨询业务是指取得监管部门颁发的相关资格的机构及其咨询人员为证券投资者或客户提供证券投资的相关信息、分析、预测或建议，并直接或间接收取服务费用的活动。

证券研究报告是证券投资咨询业务的一种基本形式，指证券公司、证券投资咨询机构对证券及证券相关产品的价值、市场走势或者相关影响因素进行分析，形成证券估值、投资评级等投资分析意见，制作证券研究报告，并向客户发布的行为。

国信证券经济研究所

深圳
深圳市罗湖区红岭中路 1012 号国信证券大厦 18 层
邮编：518001　总机：0755-82130833

上海
上海浦东民生路 1199 弄证大五道口广场 1 号楼 12 楼
邮编：200135

北京
北京西城区金融大街兴盛街 6 号国信证券 9 层
邮编：100032