|  |  |  |  |
| --- | --- | --- | --- |
| 密级： | 非密 | 编 号： | \*\*\*\*\*040403-SJ1 |
| 定密责任人： | 雷印杰 | 页 数： | 76页 |
| 版本： | A | 保管期限： | 10年 |

设备故障趋势预测算法模块设计报告

|  |  |  |  |
| --- | --- | --- | --- |
|  | **姓名** | **签名** | **日期** |
| **编制** | **李文骁** |  | **2023.3.20** |
| **校对** | **李勇成** |  | **2023.3.20** |
| **审核** | **赵成萍** |  | **2023.3.20** |
| **批准** | **雷印杰** |  | **2023.3.20** |
| **相关部门会签** |  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**四川大学**

**文件审签历史**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 版本 | 日期 | 状态 | 编写 | 校对 | 审核 | 审定 | 批准 |
| A | 2023.03.20 |  | 李文骁 | 李勇成 | 赵成萍 |  | 雷印杰 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**文件修改记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 日期 | 章节 | 页码 | 修改内容 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1 概述 1](#_Toc152254201)

[1.1 编写目的 1](#_Toc152254202)

[1.2 参考资料和设计依据 1](#_Toc152254203)

[1.2.1 参考资料 1](#_Toc152254204)

[1.2.2 设计依据 1](#_Toc152254205)

[2 使用数据集介绍 2](#_Toc152254206)

[2.1 C-MAPSS寿命预测数据集 2](#_Toc152254207)

[2.1.1 数据预处理 3](#_Toc152254208)

[2.1.2 时间窗处理 3](#_Toc152254209)

[2.2 FEMTO-ST轴承数据集 4](#_Toc152254210)

[2.2.1 数据预处理 5](#_Toc152254211)

[2.2.2 时间窗处理 5](#_Toc152254212)

[3 算法设计框架 6](#_Toc152254213)

[3.1 概述 6](#_Toc152254214)

[3.2 算法总体架构 6](#_Toc152254215)

[4 软件设计框架 7](#_Toc152254216)

[4.1 引言 7](#_Toc152254217)

[4.2 设计总体思路 8](#_Toc152254218)

[4.2.1 事件循环模块 10](#_Toc152254219)

[4.2.2 异步加载模块 10](#_Toc152254220)

[4.2.3 可视化模块 10](#_Toc152254221)

[4.3 实现细节 11](#_Toc152254222)

[4.3.1 CollapsibleBox类 11](#_Toc152254223)

[4.3.2 Ui\_Form类 12](#_Toc152254224)

[5 差分自回归移动平均 14](#_Toc152254225)

[5.1 原理介绍 14](#_Toc152254226)

[5.1.1 平稳性检验 15](#_Toc152254227)

[5.1.2 p、q确认 16](#_Toc152254228)

[5.2 算法设计 17](#_Toc152254229)

[5.2.1 ARIMA 17](#_Toc152254230)

[5.2.2 ARIMAResults 17](#_Toc152254231)

[6 指数平滑 19](#_Toc152254232)

[6.1 原理介绍 19](#_Toc152254233)

[6.2 算法设计 20](#_Toc152254234)

[6.2.1 ExponentialSmoothing 20](#_Toc152254235)

[6.2.2 Holt 21](#_Toc152254236)

[6.2.3 SimpleExpSmoothing 22](#_Toc152254237)

[6.2.4 HoltWintersResults 22](#_Toc152254238)

[7 支持向量机 24](#_Toc152254239)

[7.1 原理介绍 24](#_Toc152254240)

[7.1.1 基本定义 24](#_Toc152254241)

[7.2 算法设计 26](#_Toc152254242)

[7.2.1 BaseLibSVM 26](#_Toc152254243)

[7.2.2 SVR 27](#_Toc152254244)

[8 随机森林 29](#_Toc152254245)

[8.1 原理介绍部分 29](#_Toc152254246)

[8.1.1 集成学习 29](#_Toc152254247)

[8.1.2 决策树 29](#_Toc152254248)

[8.1.3 随机森林 32](#_Toc152254249)

[8.2 算法设计 32](#_Toc152254250)

[8.2.1 算法设计流程 32](#_Toc152254251)

[8.2.2 BaseEnsemble 33](#_Toc152254252)

[8.2.3 BaseDecisionTree 33](#_Toc152254253)

[8.2.4 DecisionTreeRegressor 34](#_Toc152254254)

[8.2.5 BaseForest 35](#_Toc152254255)

[8.2.6 ForestRegressor 35](#_Toc152254256)

[8.2.7 RandomForestRegressor 35](#_Toc152254257)

[9 卷积神经网络 37](#_Toc152254258)

[9.1 引言 37](#_Toc152254259)

[9.2 相关理论 38](#_Toc152254260)

[9.2.1 卷积 38](#_Toc152254261)

[9.2.2 池化 39](#_Toc152254262)

[9.3 算法设计 39](#_Toc152254263)

[9.4 算法实现 40](#_Toc152254264)

[9.4.1 nn.Module 40](#_Toc152254265)

[9.4.2 \_AvgPoolNd基类 41](#_Toc152254266)

[9.4.3 AvgPool1d类 41](#_Toc152254267)

[9.4.4 \_convNd基类 41](#_Toc152254268)

[9.4.5 conv2d类 43](#_Toc152254269)

[10 长短期记忆网络 45](#_Toc152254270)

[10.1 引言 45](#_Toc152254271)

[10.2 相关理论 45](#_Toc152254272)

[10.2.1 LSTM基本结构 45](#_Toc152254273)

[10.3 算法设计 47](#_Toc152254274)

[10.4 算法实现 48](#_Toc152254275)

[10.4.1 nn.Module 48](#_Toc152254276)

[10.4.2 RNNBase基类 48](#_Toc152254277)

[10.4.3 LSTM类 52](#_Toc152254278)

[11 Transformer 54](#_Toc152254279)

[11.1 引言 54](#_Toc152254280)

[11.2 相关理论 55](#_Toc152254281)

[11.2.1 自注意力机制 55](#_Toc152254282)

[11.3 算法设计 56](#_Toc152254283)

[11.3.1 Linear类 56](#_Toc152254284)

[11.3.2 Transformer\_RUL类 57](#_Toc152254285)

[12 图神经网络 59](#_Toc152254286)

[12.1 引言 59](#_Toc152254287)

[12.2 相关理论 59](#_Toc152254288)

[12.3 实现细节 60](#_Toc152254289)

[12.3.1 KOI\_model\_train\_test\_interface类 60](#_Toc152254290)

[12.3.2 Time\_Embeddings类 61](#_Toc152254291)

[12.3.3 Time\_Series\_Self\_Attention类 61](#_Toc152254292)

[12.4 Graph\_Representation\_Readout类 62](#_Toc152254293)

[12.4.2 Time\_Series\_Attention\_Transformer类 64](#_Toc152254294)

[13 其它工具类设计 66](#_Toc152254295)

[13.1 环境要求 66](#_Toc152254296)

[13.2 模型性能评估 66](#_Toc152254297)

[13.2.1 mse函数 67](#_Toc152254298)

[13.2.2 rmse函数 67](#_Toc152254299)

[13.2.3 score函数 67](#_Toc152254300)

[13.2.4 R2函数 68](#_Toc152254301)

[13.3 模型训练和保存 68](#_Toc152254302)

[13.3.1 pth文件 68](#_Toc152254303)

[13.3.2 onnx文件 68](#_Toc152254304)

[13.3.3 pt文件 68](#_Toc152254305)

[13.3.4 pkl文件 68](#_Toc152254306)

[13.3.5 h5文件 69](#_Toc152254307)

[13.3.6 model\_saved函数 69](#_Toc152254308)

[13.3.7 model\_loaded 69](#_Toc152254309)

[13.4 模型可视化 69](#_Toc152254310)

[13.4.1 plot\_tree函数 69](#_Toc152254311)

[13.4.2 torch.onnx.export函数 70](#_Toc152254312)

[13.4.3 Netron可视化工具介绍 70](#_Toc152254313)

# 概述

## 编写目的

本文旨在深入研究工业机械设备关键部件的趋势/寿命预测方法。我们将详细探讨设备故障趋势预测和寿命预测领域的算法设计细节，内容涵盖了统计学习、机器学习和深度学习等算法的设计过程。具体内容包括工具类的设计、数据预处理的设计、具体算法的设计等方面，旨在为后续工作的开展提供坚实的基础。

在本文中，我们将展开讨论关于趋势/寿命预测方法的设计，并将着重介绍设计阶段的关键方面。我们将深入研究工具类的设计，以确保其能够支持可靠的数据处理和算法实施。此外，我们将探讨数据预处理的设计策略，以确保数据质量和一致性。最重要的是，我们将详细介绍具体算法的设计过程，包括模型选择、参数调优和性能评估。

这一全面的设计探讨将为研究人员和开发人员提供坚实的依据，以在工业机械设备领域中开展更深入的工作。我们的目标是确保设计出可靠和高效的趋势/寿命预测方法，以满足工业机械设备的需求，并为未来的研究和应用奠定坚实基础。

## 参考资料和设计依据

### 参考资料

|  |  |
| --- | --- |
| GB/T 14394-2008 | 计算机软件可靠性和可维护性 |
| EJ/T 644-1992 | 核工业计算机软件开发规范 |
| EJ/T1006-1996 | [核工业计算机软件摘要](https://biaozhun.supfree.net/ifeng.asp?id=26983) |

### 设计依据

|  |  |
| --- | --- |
| \*\*\*\*\*040403 | XXX关键设备部件性能预测与运维决策支持技术研究项目合同 |

# 使用数据集介绍

## C-MAPSS寿命预测数据集

项目中使用到的寿命预测数据集源自于NASA的航空发动机仿真模型（Commercial Modular Aero-Propulsion System Simulation，C-MAPSS），该数据集为航空涡扇发动机剩余使用寿命（Remaining Useful Life，RUL）预测领域被广泛使用的基准数据，发动机模型如图2‑1所示。

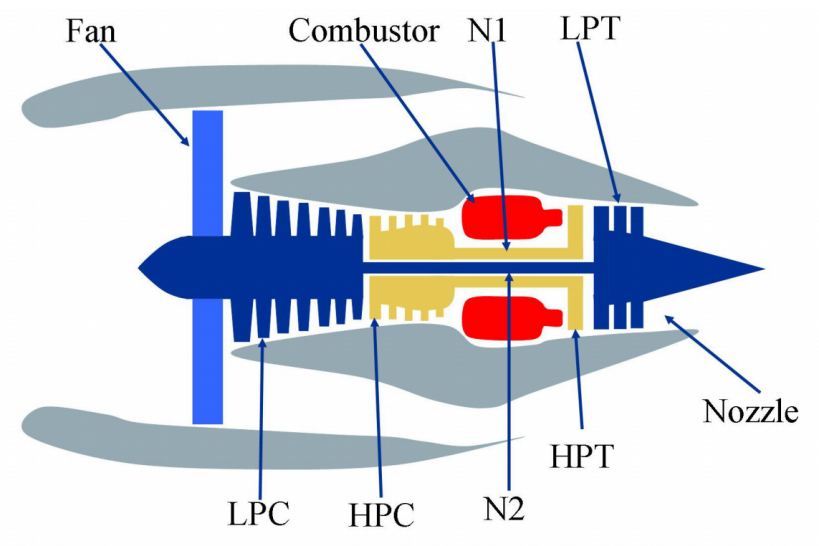


图2‑1 发动机模型图

CMAPSS数据集中有四个子集分别是FD001、FD002、FD003、FD004，数据是以文本形式给出，每个子集包含了不同的操作状态以及不同数量的模拟故障引擎数量，随着时间的推移，每台发动机都会出现不同类型的故障，当特定引擎发生故障时，每个引擎的数据将停止，因此实际的剩余寿命是根据数据的长度来确定的。CMAPSS数据集数据总览如表2‑1所示。

表2‑1 CMAPSS数据集总览

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | FD001 | FD002 | FD003 | FD004 |
| 训练样本数 | 100 | 260 | 100 | 249 |
| 测试样本数 | 100 | 259 | 100 | 248 |
| 训练样本长度 | 20361 | 53759 | 24720 | 61249 |
| 运行工况 | 1 | 6 | 1 | 6 |
| 故障种类 | 1 | 1 | 2 | 2 |

从表中可以看出，FD002和FD004数据集有着更多的运行工况条件以及故障类别，因此对于这两个数据集进行准确的寿命预测更加有难度。

数据集记录了每个航空发动机，在整个生命周期，每个采样时刻的24个维度性能参数，包括3个操作状态变量和21个传感器测量变量（具体变量见《设备故障趋势预测算法研究报告》3.1节表3-2）。因此，每组数据共有26列（第1列为发动机号，第2列为运行周期数，第3~5列为运行条件，后21列为机身传感器监测数据）。

对于训练集中的每一个发动机，都经历了完整的从正常状态到故障状态的转变，记录的是发动机全寿命周期数据，对于测试集中的每一个发动机，测量数据在发动机故障之前的某一时刻终止，需要预测出每个发动机的剩余寿命，图2‑2表示的是FD001子数据集中某台发动机在一个生命周期内的传感器数据样例。

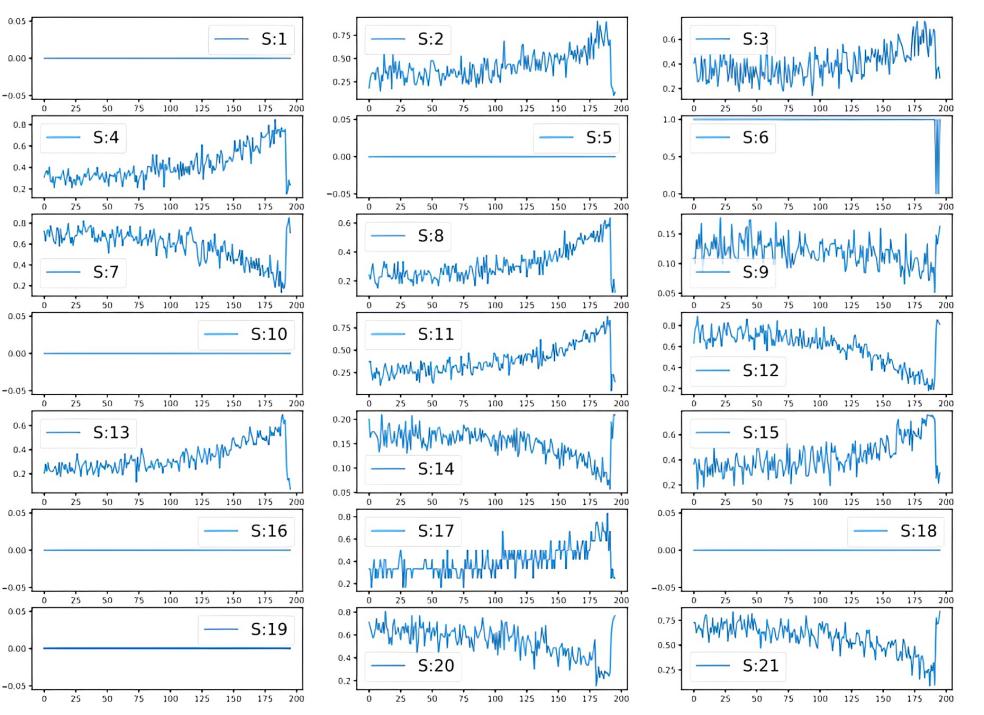


图2‑2 FD001 1号发动机传感器数据

### 数据预处理

由图2‑2可以看出，某些传感器监测数据在发动机整个寿命周期内是一个常数，因此对于故障趋势预测模型的构建没有帮助，需要在数据预处理环节删除这些数据列，具体来说，删除传感器1，5，6，10，16，18，19列监测数据，保留传感器2，3，4，7，8，9，11，12，13，14，15，17，20，21共14列监测数据

为了使不同传感器特征具有相同的度量尺度，帮助模型更好的适应数据，还需要对数据进行归一化操作，其中归一化方程为：



式中为某个发动机寿命周期数据中第个传感器监测数据的第个数据点，和分别表示发动机寿命周期数据中第个传感器监测数据的最大值和最小值，表示归一化后的值。

### 时间窗处理

相对于单个时间点的多变量时间数据，多变量时间序列数据往往能够蕴含更多的信息，例如趋势信息和相关性信息，因此采用一个时间窗口对原始时间序列数据进行滑动取值，以获取剩余寿命预测输入数据。展示了使用滑动窗口获取训练样本数据，其中滑动窗口Time Window的滑动步长（slide size）为1，窗口大小（window size）为30，沿着Time Cycles方向进行滑动，每次滑动得到一个训练样本，每个样本的形状为[30，14]，即代表这个样本包含了30个时间序列样本点，每个样本点的特征维度为14。如图2‑3所示：

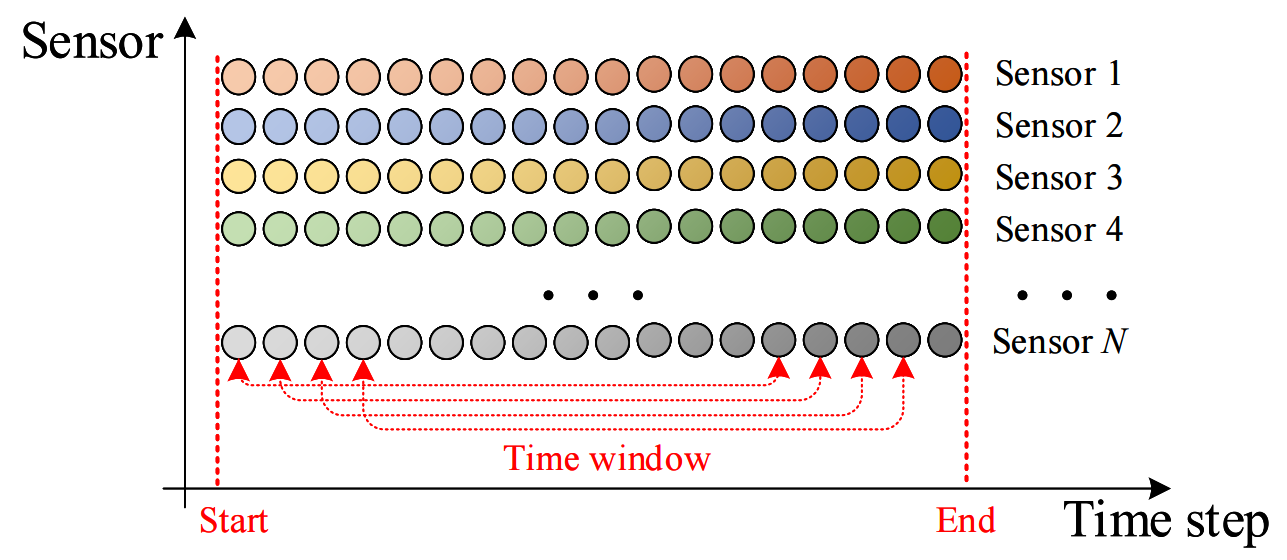


图2‑3 滑动窗口示意图

## FEMTO-ST轴承数据集

PHM2012数据集工况多变，包含不同工作条件/负载（转速和负载力）的数据。数据集每个文件夹都包含了名为acc\_xxxxx.csv的振动ASCII码文件，采样频率为25.6kHZ，每隔10秒采样0.1秒，即每10秒记录2560个样本，对于每个文件其数据列组成如表2‑2所示。

表2‑2 数据列组成

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column | 1 | 2 | 3 | 4 | 5 | 6 |
| Vibr signal | Hour | Minute | Second | μ-second | Hoiz sccel | vert accel |
| Temp signal | Hour | Minute | Second | 0.x second | Rtd sensor | vert accel |

对于该数据集，一般选择使用采样的振动信号来进行相关的性能趋势预测分析，对于振动信号文件，每一行包括了6列信息，分别为开始时刻时分秒、平均秒数、水平振动加速度、垂直振动加速度，取轴承1-1数据集中的csv文件，其水平加速度与垂直加速度大小随时间的关系如图2‑4所示。

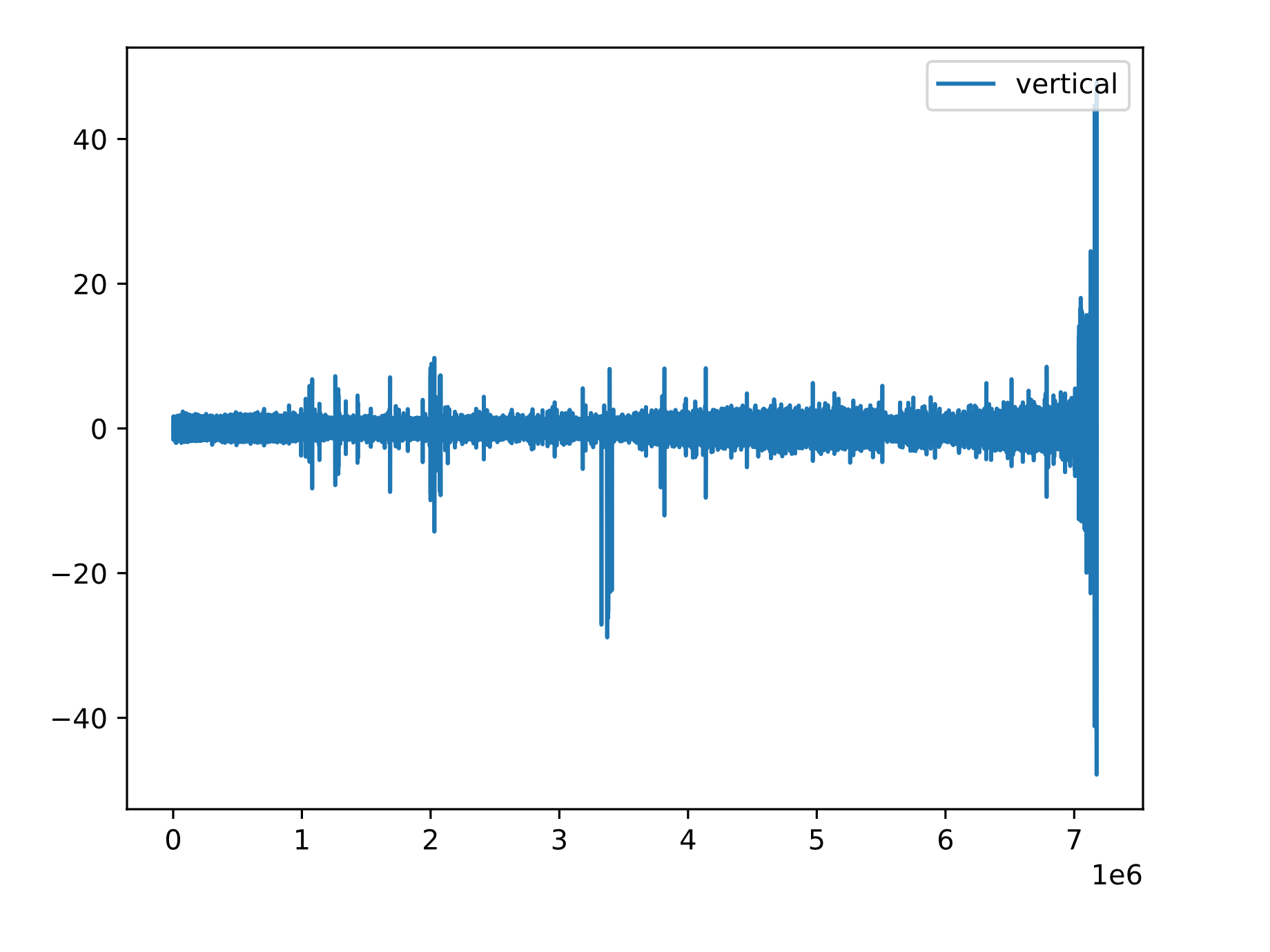
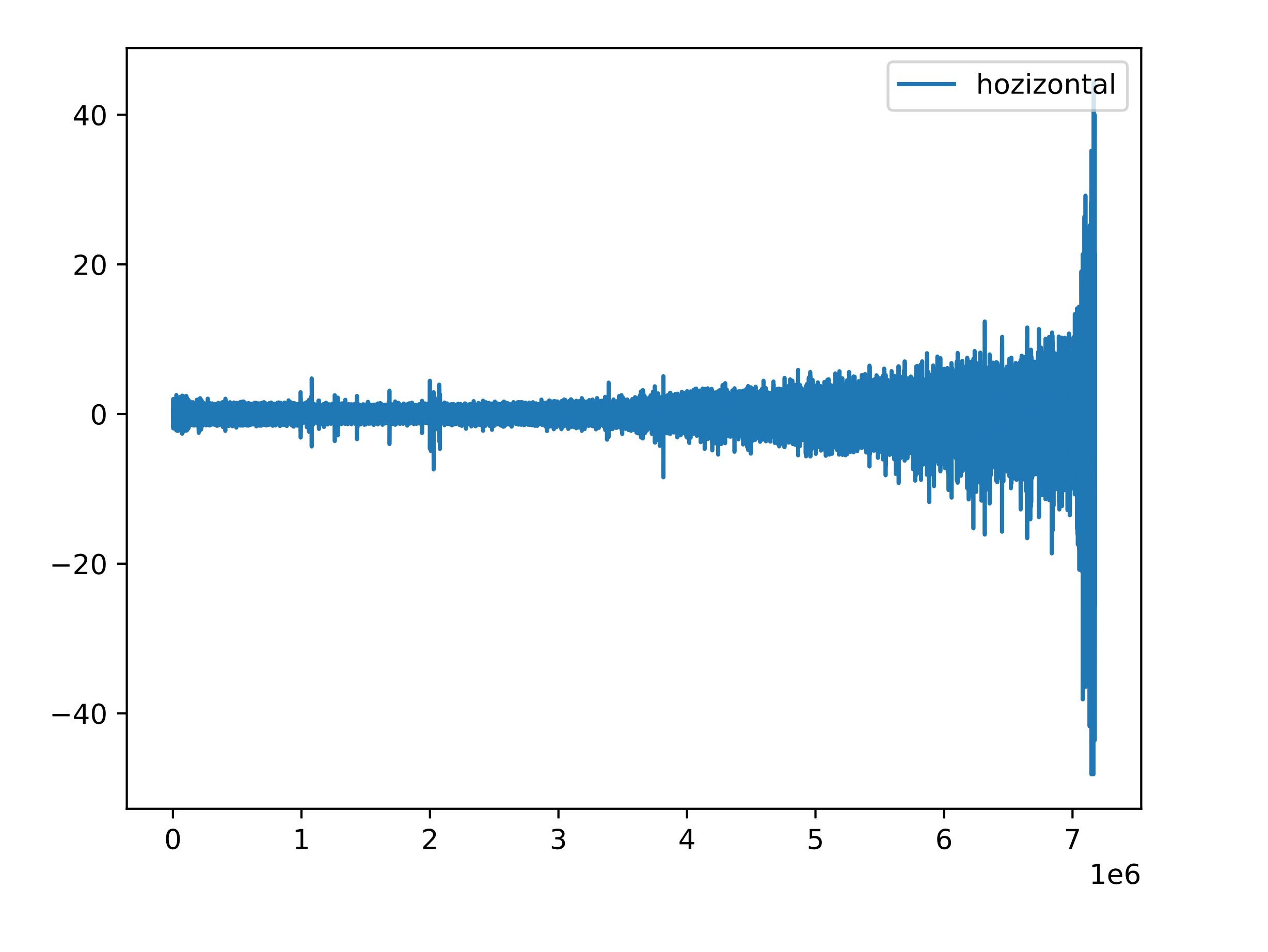


图2‑4 轴承监测加速度曲线

### 数据预处理

与航空发动机数据集类型，为了使不同传感器特征具有相同的度量尺度，帮助模型更好的适应数据，对数据进行归一化操作，其中归一化方程为：



式中为某个轴承寿命周期数据中第个传感器监测数据的第个数据点，和分别表示发动机寿命周期数据中第个传感器监测数据的最大值和最小值，表示归一化后的值。

### 时间窗处理

相对于单个时间点的多变量时间数据，多变量时间序列数据往往能够蕴含更多的信息，例如趋势信息和相关性信息，因此采用一个时间窗口对原始时间序列数据进行滑动取值，以获取趋势预测输入数据。与1.1.2节中处理方式一样，采用滑动窗口的方式依次截取单个样本，滑动窗口的大小为120，其中滑动窗口中的前96个元素作为训练集样本，后24个元素作为训练集标签。

# 算法设计框架

## 概述

在项目的寿命预测/趋势预测部分，我们专注于开发和应用机器学习和深度学习方法，以解决关键设备部件性能的寿命预测问题。我们之所以选择这些方法，是因为它们能够从历史性能数据中提取关键信息，帮助我们预测设备寿命和性能趋势。

机器学习和深度学习：我们采用机器学习和深度学习技术，因为它们能够自动学习和捕捉设备性能数据中的复杂趋势和模式。机器学习模型具有能力识别性能下降的迹象，而深度学习模型则能够有效处理大规模、多维的性能数据，以提供更准确的寿命预测结果。

这一部分的目标是为关键设备部件提供可靠的寿命预测，帮助项目方更好地规划设备维护和更换计划，以降低潜在的故障风险，提高设备的可靠性，并实现更高的运营效率。我们相信机器学习和深度学习方法将为寿命预测提供强大的工具，使项目能够做出明智的决策，确保设备的长期可用性。

## 算法总体架构

算法开发总体架构如图3‑1所示，针对趋势预测算法，将其分为了3个模块进行开发，分别为统计学习、机器学习、深度学习，每个模块下有着不同的预测算法，这种模块化的开发方式有利于后期动态增加新的算法以及开发过程中的维护性工作。

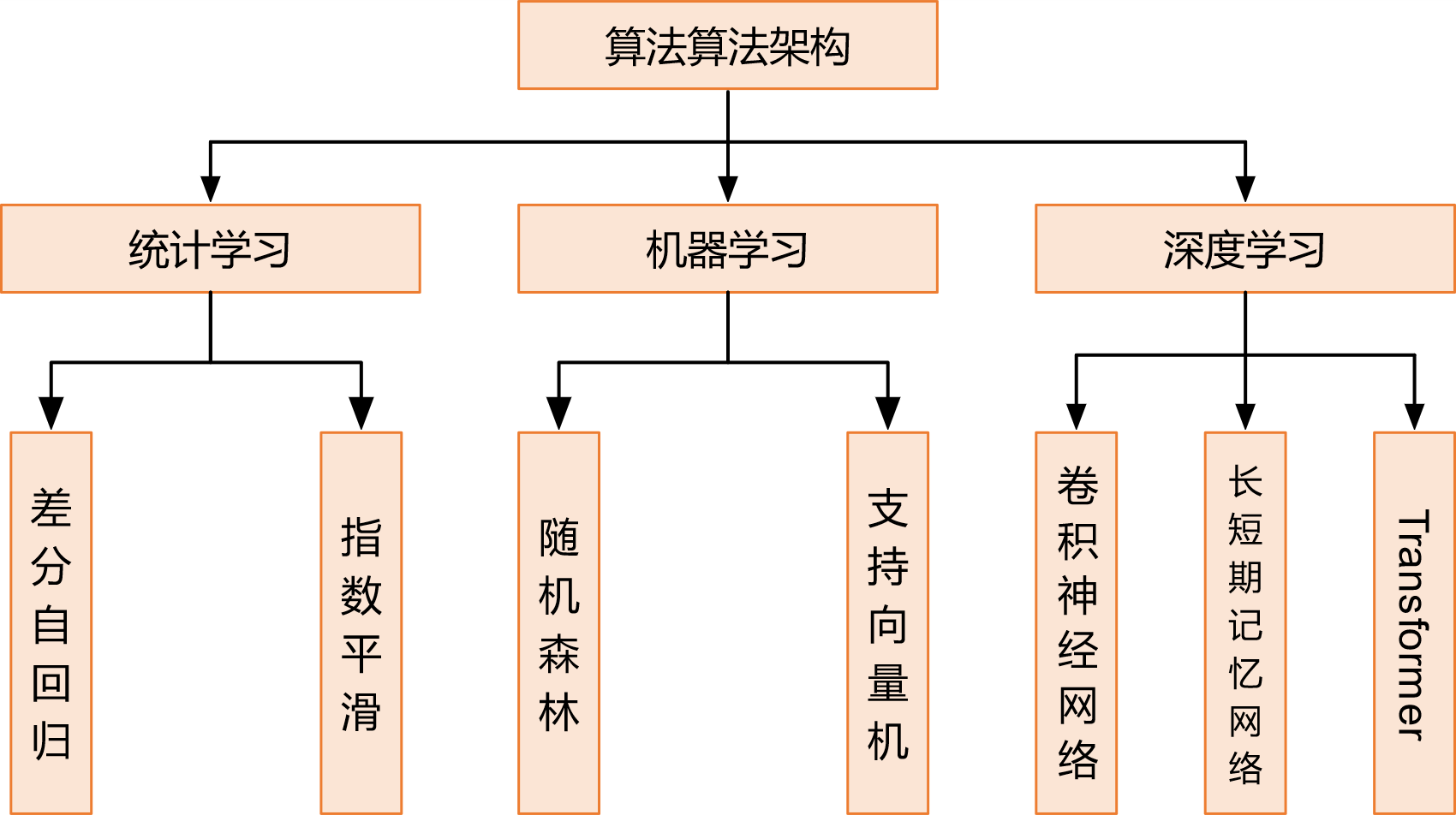


图3‑1 算法框架模型

# 软件设计框架

## 引言

在设备趋势预测和寿命预测的算法设计项目中，不仅需要开发出相应的趋势预测算法，还需要将这些算法集成到相应的算法开发软件工具箱中以便于调用和测试相应的趋势预测算法。为了开发相应的算法软件工具箱，使其能够将算法结果以直观、易于理解的方式展示给用户，提供良好的用户体验和决策支持。该软件平台的设计应充分遵循了高内聚低耦合的架构原则，以确保系统的模块化性和可维护性。以下是算法软件工具箱的开发准则：

1）高内聚低耦合的设计：该软件平台的总体设计采纳了高内聚低耦合的设计范式，这一原则确保各个模块能够独立运作并减少模块之间的相互依赖。这意味着不同模块可以在不干扰其他模块的情况下进行开发、测试和维护。这种设计方式大幅提高了系统的可维护性和可扩展性，使其适应未来需求的变化。

2）模块化的算法设计：在算法模块中，对模型训练和数据清洗等任务进行模块化设计，这使得每个算法模块专注于特定任务。这种模块化设计提供了更高的代码可读性和可测试性。算法的独立模块化构建还有助于更容易地开发和维护各种算法，从而使平台能够不断演进，应对新的数据挖掘和故障诊断需求。

3）解耦算法模块和客户端模块：为了实现高度的灵活性和可维护性，核心算法模块与客户端模块之间的耦合已被有效解耦。这意味着算法开发人员可以独立工作，而不需要深入修改客户端代码。新的算法模块可以轻松地添加到系统中，同时现有的模块之间的依赖关系被最小化，从而提高了系统的可扩展性。

4）多线程模型的客户端设计：为了提高客户端的性能和响应速度，采用多线程模型来允许并行处理多个任务。多线程设计使得软件平台能够在运行时同时执行多个任务，从而更快地处理数据和算法计算。同时，采用并发编程技术，以解决线程阻塞问题，确保客户端的稳定运行。

这些设计决策共同助力于软件平台的高度灵活性和性能优化，使其能够更好地满足用户的需求，提高算法预测的工作效率。这一综合的设计理念旨在确保软件平台在复杂任务的背景下以高度专业和可维护的方式运行，软件开发总体架构如图4‑1所示。

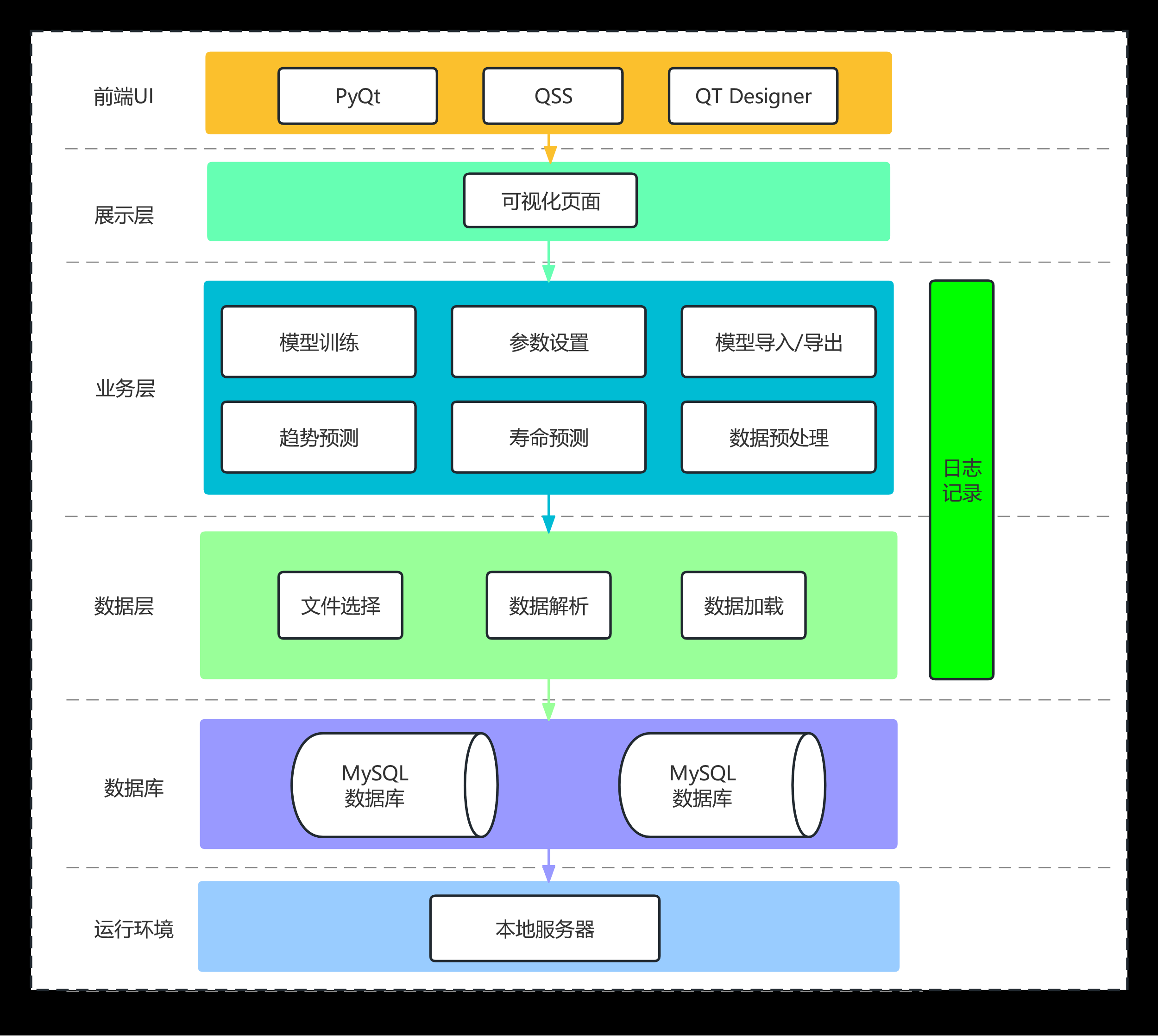


图4‑1 软件开发总体架构图

## 设计总体思路

经过前期调研，确认了Python为项目开发过程中的编程语言，在开发算法软件工具箱的过程中为了与算法模块快速集成以减少开发成本和降低开发时间，《XXX关键设备部件性能预测与运维决策支持技术研究》项目的可视化软件开发也采用Python开发相关规范，并遵循Python Enhancement Proposal 8编程语言标准。

为了实现这一目标，我们选择了PyQt作为我们的可视化软件开发工具。PyQt是一个功能强大、灵活且跨平台的Python库，它结合了Qt应用程序框架和Python语言的优势。借助PyQt，我们能够轻松创建交互性强、美观且易于使用的用户界面，使用户能够直观地理解和操作设备趋势预测和寿命预测的算法。通过使用PyQt，我们能够提供一种全面且可靠的解决方案，满足用户对于可视化软件界面的需求，并帮助他们更好地利用算法结果进行决策和规划。表4‑1列举了PyQt中一些常用的组件。

表4‑1 PyQt常用组件

|  |  |
| --- | --- |
| 组件 | 描述 |
| QWidget | 所有用户界面对象的基类，提供基本的事件处理和绘图功能。 |
| QLabel | 用于显示文本或图像。 |
| QPushButton | 按钮组件，用于触发事件或执行操作。 |
| QLineEdit | 单行文本输入框。 |
| QTextEdit | 多行文本输入框。 |
| QComboBox | 下拉框。 |
| QCheckBox | 复选框。 |
| QRadioButton | 单选框。 |
| 布局管理器 | |
| QVBoxLayout | 垂直布局。 |
| QHBoxLayout | 水平布局。 |
| QGridLayout | 网格布局，用于在表格中排列组件。 |
| QFormLayout | 用于表单的布局。 |
| QMainWindow | 主窗口类。 |
| Qt网络模块 | |
| QTcpSocket、QTcpServer | TCP套接字和服务器类。 |
| QUdpSocket | UDP套接字类。 |
| 多线程支持 | |
| QThread | 用于创建多线程的类。 |

客户端页面功能模块划分如图4‑2所示。

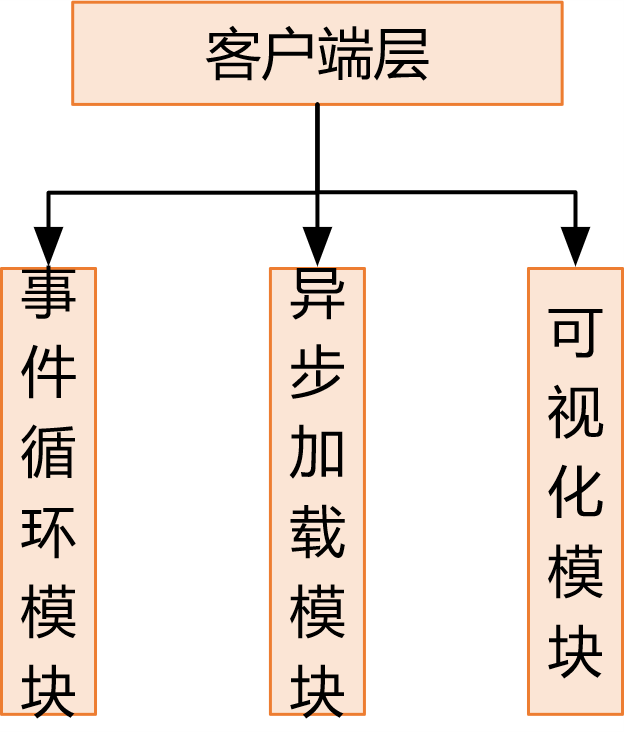


图4‑2 客户端功能模块架构

基于图4‑2的客户端功能模块架构，我们将软件的代码逻辑结构划分为三层：事件循环模块、异步加载模块和可视化模块。每个模块都承担特定的职责，以实现不同的业务功能和业务逻辑。以下是对每个模块的详细介绍。

### 事件循环模块

事件循环模块是软件系统的核心部分，通过QtWidgets.QApplication(sys.argv)创建一个QApplication实例app，该实例用于管理整个应用程序的事件循环和窗口系统的交互。并使用QtWidgets.QMainWindow()创建一个主窗口对象，该窗口将作为整个应用程序的主窗口，负责管理和协调不同的任务和事件。它包含了主要的应用程序逻辑，用于响应用户输入、处理系统事件和调度各个子任务。事件循环模块充当了软件的主引擎，确保不同的组件之间的协作和数据流畅通。

使用QtWidgets.QDockWidget创建一个可折叠的Dock窗口对象dock，并设置其标题为"故障趋势预测算法工具箱"。创建并添加自定义UI部件：创建Ui\_Form的实例ui，并将其与窗口左侧类建立联系，实现具体功能逻辑。进入应用程序的事件循环：使用sys.exit(app.exec\_())启动应用程序的事件循环，使应用程序保持运行状态，直到用户关闭窗口或终止应用程序。

职责：事件循环模块的主要职责包括处理用户输入、监控系统状态、执行任务调度、处理错误和异常，以及协调不同模块之间的通信。

技术特点：事件循环通常使用异步编程技术来实现，以确保高效的任务调度和响应性能。它通常包括事件处理器、回调函数和状态机，用于处理各种事件和任务。

### 异步加载模块

多线程异步模块负责处理并发任务和异步操作，以提高系统的性能和响应能力。它允许多个任务在并行执行，以减少任务处理时间，从而提供更好的用户体验。

职责：多线程异步模块的职责包括管理线程、执行并发任务、处理异步请求和数据处理，以充分利用多核处理器和减少任务的阻塞时间。

技术特点：多线程异步模块使用多线程编程技术，例如多线程池、异步编程库和并发数据结构，以实现任务并发和异步处理。

### 可视化模块

可视化模块是用户界面的重要部分，负责呈现数据和用户交互。它提供了用户友好的界面，以便用户可以与系统进行互动和获取信息。创建可折叠的Dock窗口：

该模块设计思路如下：

创建可折叠的Dock窗口：使用QtWidgets.QDockWidget创建一个可折叠的Dock窗口对象dock，并设置其标题为"故障趋势预测算法工具箱"。然后，使用w.addDockWidget(QtCore.Qt.LeftDockWidgetArea, dock)将该Dock窗口添加到主窗口的左侧。

创建并添加自定义UI部件：创建Ui\_Form的实例ui，并将其与窗口左侧类建立联系，实现具体功能逻辑。进入应用程序的事件循环：使用sys.exit(app.exec\_())启动应用程序的事件循环，使应用程序保持运行状态，直到用户关闭窗口或终止应用程序。

职责：可视化模块的主要职责是设计和管理用户界面，显示数据、图表和图形，处理用户输入和提供反馈，以确保用户能够轻松使用系统。

技术特点：可视化模块通常使用图形库、用户界面工具包和视图控制器模式来创建和管理用户界面。它涉及布局设计、图形渲染、事件处理和用户交互。

综上所述，该模块通过使用PyQt库和自定义的可折叠部件，创建了一个具有可折叠功能的主窗口，并将其他部件和自定义UI部件添加到不同的布局中，以实现设备趋势预测和寿命预测软件界面的设计。通过使用PyQt提供的丰富功能和灵活性，用户能够直观地理解和操作算法结果，并进行健康评估和运维决策

## 实现细节

### CollapsibleBox类

#### \_\_init\_\_函数

功能描述:该类用于创建一个可折叠的部件，其中包含一个标题和一个可以展开或折叠的内容区域。用户可以点击标题来切换内容区域的显示状态。。

参数介绍:

* title：部件的标题。
* parent：父级部件，默认为None。

#### on\_pressed函数

功能描述:该函数的主要功能是根据按钮的状态切换箭头的方向，用于显示内容的展开或收起状态。同时，它启动一个动画来实现平滑的展开和收起效果。

参数介绍:

* self: 表示当前对象或类的实例。
* toggle\_button: 表示切换按钮的对象，它应该是一个QtWidgets.QPushButton或其子类的实例。
* toggle\_animation: 表示切换动画的对象，它应该是一个QtCore.QAbstractAnimation或其子类的实例。

#### setContentLayout函数

功能描述：实现了一个方法setContentLayout，该方法用于设置可折叠部件的内容布局，并在布局中添加多个QPushButton按钮。在按钮点击事件中，会调用buttonClicked方法。

参数介绍：

* self: 指向当前对象的引用。
* layout: 待设置的内容布局对象。

### Ui\_Form类

#### \_\_init\_\_函数

功能描述:

该函数用于初始化Ui\_Form类的实例。创建了一个垂直布局(verticalLayout\_2)，并将其设置为类的主布局。然后在主布局中创建了一个垂直布局(verticalLayout)和一个水平布局(horizontalLayout)。

#### TrainFileSelect函数

功能描述:

该函数实现了一个训练文件选择函数，当调用该函数时，会弹出一个文件选择对话框，允许用户选择训练文件。选择文件后，将文件路径存储在self.trainFileName变量中，并将文件路径信息显示在传入的text组件中。

参数介绍:

* self: 对象自身的引用，表示调用该函数的对象。
* a: 传入的text组件，用于显示文件路径信息。

#### TestFileSelect函数

功能描述:

该函数用于实现文件选择对话框，并将选择的文件路径设置为指定的文本框的文本。

参数介绍:

* self: 表示当前类的实例对象。
* a: 表示一个文本框对象，用于显示选择的文件路径。

#### update\_textEdit\_4函数

功能描述:

该函数用于更新文本编辑框（self.textEdit\_4）的内容。根据传入的参数，包括按钮名称（button\_name）和文本数组（text\_array），函数执行以下操作：检查文本数组中的第一个元素，如果为'寿命预测'，则将self.mode设置为'RUL'，否则将self.mode设置为'Prediction'。然后，将按钮名称添加到self.textEdit\_4中。

参数介绍:

* self：指向类实例自身的引用。
* button\_name：字符串，表示按钮的名称。
* text\_array：包含文本数组的列表，根据数组的内容来更新self.mode和self.algorithm\_name。

#### TrainFunc函数

功能描述:

该函数的主要功能是根据选择的模式（寿命预测或趋势预测）和算法名称，加载相应的训练数据集，并使用选择的算法模型对数据集进行训练。训练过程将进行多个epoch的循环，每个epoch都会计算并输出训练损失。

参数介绍:

* self.running: 一个布尔值，表示训练循环是否继续运行。
* self.mode: 字符串，表示模式（'RUL'或'Prediction'）。
* self.algorithm\_name: 字符串，表示选择的算法名称。
* self.pushButton\_3: PyQt界面中的按钮部件。
* self.trainFileName: 字符串，表示训练文件的路径。
* a: 列表，用于记录训练过程中的信息。
* data\_loader: PyTorch的数据加载器，用于批量加载训练数据。
* self.trainData: 训练数据集。
* self.model: 训练使用的算法模型。
* criterion: 损失函数。
* optimizer: 优化器。

#### TestFunc函数

功能描述:

该函数的功能是根据给定的模式（寿命预测或趋势预测）和算法名称，加载测试数据并使用相应的算法模型进行测试。

参数介绍:

* mode: 模式，表示是进行寿命预测还是趋势预测。
* algorithm\_name: 算法名称，表示要使用的算法模型的类型。
* self.TestFileName: 测试文件名，表示要加载的测试数据文件的路径。

# 差分自回归移动平均

## 原理介绍

差分自回归移动平均模型(Auto Regressive Integrate Moving Average Model，ARIMA)是一种流行且广泛使用的时间序列预测统计方法。ARIMA结合了自回归(Auto Regressive Model，AR)、移动平均（Moving Average Model，MA）、差分模型，是自回归移动平均模型(Auto Regressive Moving Average Model，ARMA)的进一步拓展，主要针对平稳时间序列或者差分化后平稳的时间序列数据。

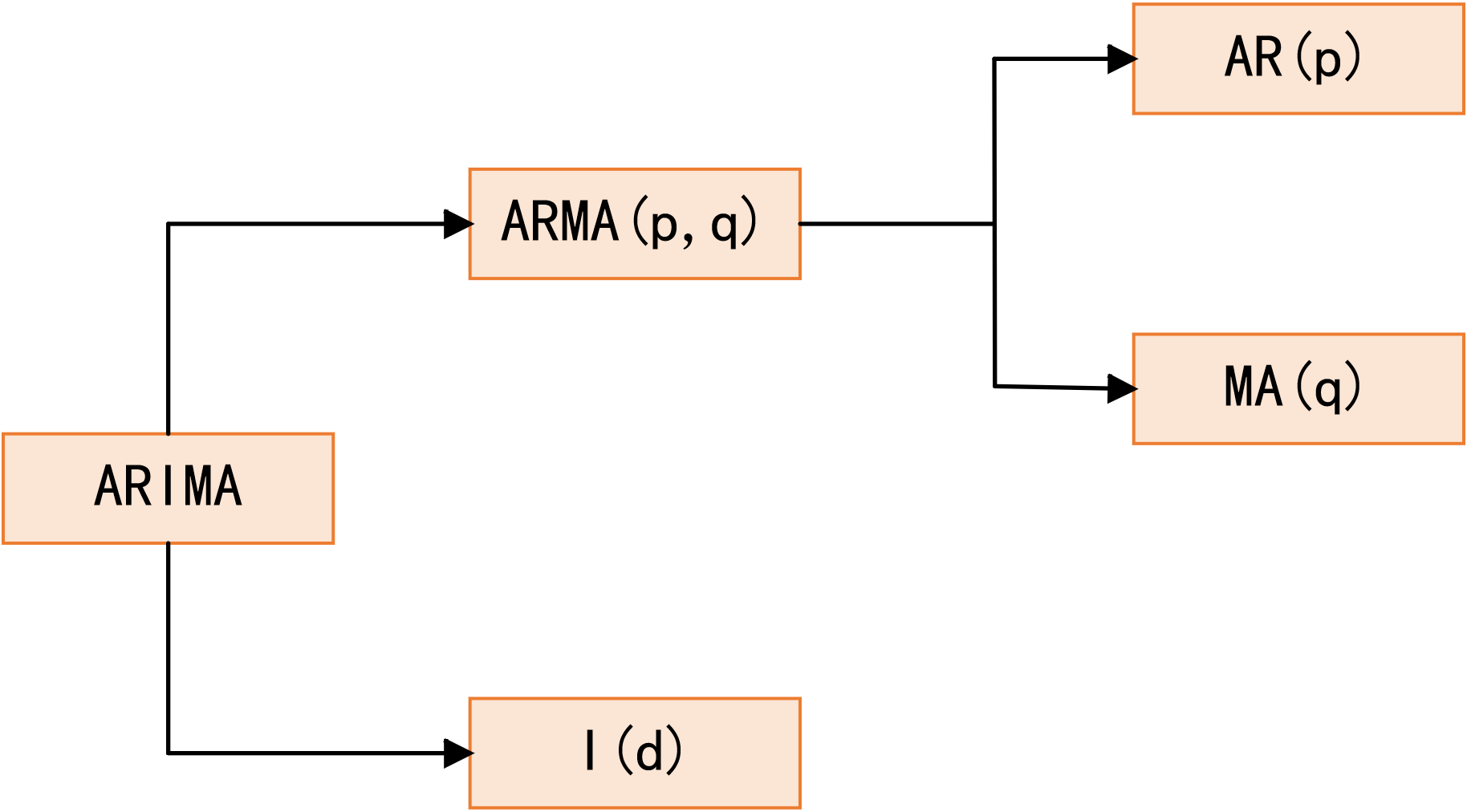


图5‑1 ARIMA模型

ARIMA将时间序列的每一个时刻值当做是过去时刻的序列值以及外界干扰量的线性组合。即通过对过去时刻的序列值和外界干扰量的确定，就能通过一定的函数关系将当前时刻的序列值表达出来。

ARIMA模型主要有三个参数：p、d、q。

p代表预测模型中采用的时序数据本身的滞后数，也叫做自回归项。

d代表时序数据需要进行几阶差分后变为平稳序列，也叫差分项。

q代表预测模型中采用的预测误差的滞后数，也叫做移动平均项。

ARIMA模型表示如下：



上式中是经过差分后的平稳时间序列，右侧为的延迟值和延迟误差值，该模型被定义为ARIMA（p，d，q），p代表自回归模型阶数，d代表差分阶数，q为移动平均阶数。

ARIMA建模流程如图5‑2所示：

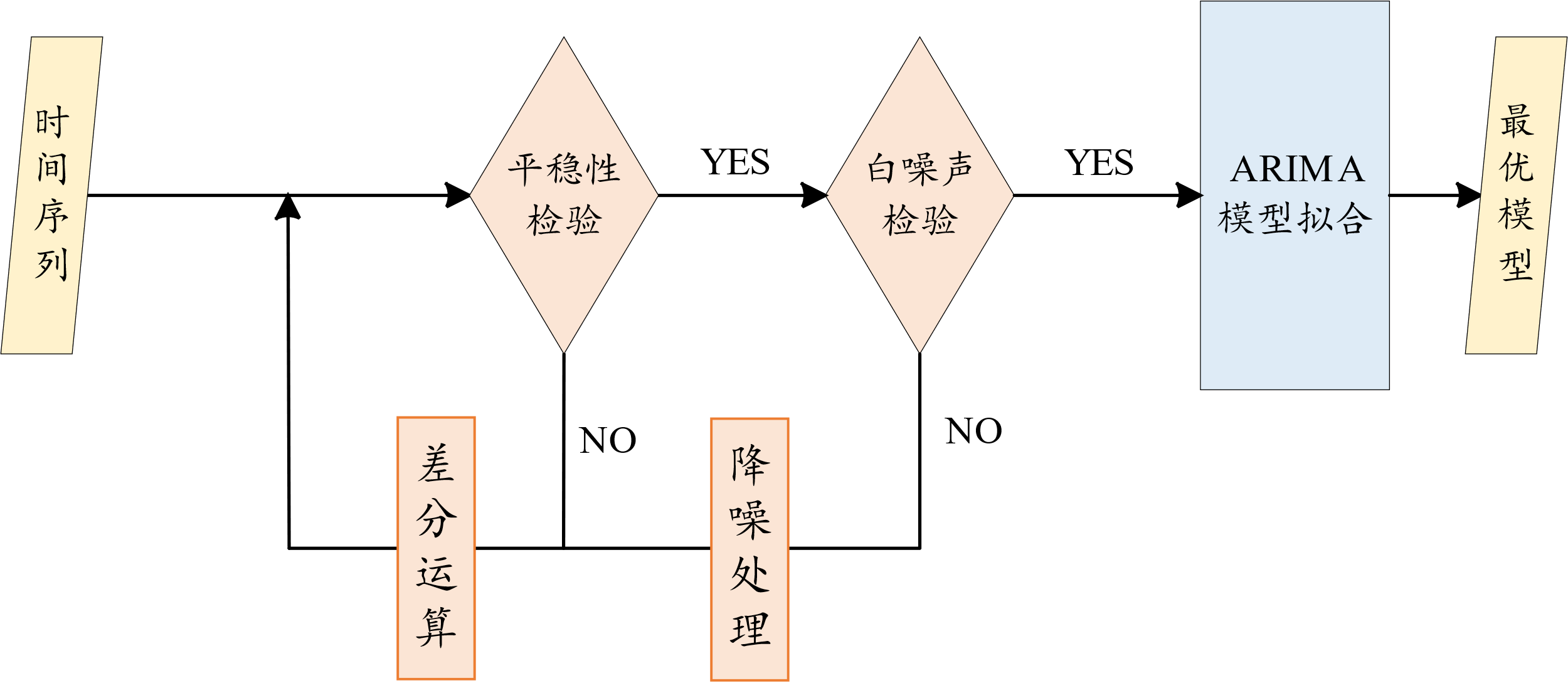


图5‑2 ARIMA建模流程

1、获取时间序列数据并判断其平稳性，不平稳则通过差分操作得到平稳时间序列

2、通过求相关系数或BIC信息准则等方法得到模型阶数p、q，得到拟合模型ARIMA（p，d，q）

3、模型检验、优化

4、模型预测

### 平稳性检验

因为ARIMA模型要求输入数据为平稳数据，在将输入数据送入模型之前需要判断原数据平稳性，若不平稳，需要对原始数据进行差分以获得平稳数据。

单位根检验用于检验时间序列的单位根存在与否。单位根存在意味着时间序列是非平稳的，需要进行差分处理。常用的单位根检验方法包括ADF检验（Augmented Dickey-Fuller Test）和KPSS检验（Kwiatkowski-Phillips-Schmidt-Shin Test）。

项目中采用了ADF单位根检验，它使用自回归模型并优化跨多个不同滞后值的信息标准，通过结果中的P值大小来确定原始时间序列是否平稳，若P小于等于值0.05，则拒绝原假设，原始时间序列数据没有单位根是平稳的，若P值大于0.5，则无法拒绝原假设，原始时间序列数据具有单位根且是非平稳的。

通过statsmodels模块的adfuller方法可快速检验输入数据是否平稳，该函数参数如下：

x：一维数据序列

maxlog：最大滞后数目

regression：回归中的包含项

autolag：自动选择滞后项准则，可选为AIC、BIC

返回结果为一元组，各个元素分别为：

Test statistic：T检验，假设检验值

p-value：假设检验结果

Lags Used：使用的滞后阶数

Number of Observation：用于ADF回归和计算临界值用到的观测值数目

Critical Value：1%、5%、10%临界值大小

如果经过统计检验后发现时间序列数据的p值大于0.05，则表明原始数据中存在趋势和季节性的影响，为了使时间序列更加稳定和可预测，则需要进行差分操作。差分操作是一种针对时间序列数据的处理方法，通过减法运算将当前观测值与前一个观测值之间的差异作为新的观测值。这样做的目的是消除时间序列中的趋势和季节性影响，使得数据更加平稳。通过差分操作，可以得到一系列的差分值，这些差分值代表了时间序列在不同时间点的变化情况。在差分后的序列中，如果趋势和季节性的影响被有效消除，序列将更加稳定，随机性更强，从而更容易进行模型建立和预测。

### p、q确认

经过前面的平稳性检测，现在需要确定ARIMA模型的p、q大小，通过对输入时间序列求自相关系数ACF和偏自相关系数PACF来确定p、q大小。

自相关函数（Autocorrelation Function，ACF）用于衡量一个时间序列与其自身在不同滞后阶数下的相关性，其计算公式如下：







偏自相关函数（Partial Autocorrelation Function，PACF）用于衡量一个时间序列与其自身在不同滞后阶数下消除其他阶数影响后的相关性，其计算公式如下：



通过对自相关图和偏自相关图的分析，以得到最佳的阶数p和阶数q，如果样本自相关系数和样本偏自相关系数在最初的阶明显大于2倍标准差，而后几乎95%的系数都落在2倍标准差的范围内，且非零系数衰减为小值波动的过程非常突然，通常视为k阶截尾，如果有超过5%的样本相关系数大于2倍标准差，或者非零系数衰减为小值波动的过程比较缓慢或连续，通常视为拖尾。

p、q阶数的确定与自相关图和偏自相关图的关系如表5‑1所示

表5‑1 p、q值对照表

|  |  |  |
| --- | --- | --- |
|  | ACF拖尾 | ACF截尾 |
| PACF拖尾 | ARMA（p、q） | MA（q） |
| PACF截尾 | AR（p） | ARMA模型不适用 |

单纯地依靠自相关系数和偏自相关系数来确认p、q值具有一定的主观性，为了避免模型过拟合使得在模型复杂度与模型对数据集描述能力之间寻求一个最佳平衡，可以进一步借助赤池信息准则（Akaike Information Criterion，AIC）和贝叶斯信息准则（Bayesian Information Criterion，BIC）来帮助我们确认最后的

阶数p和阶数q。AIC是衡量统计模型拟合优良性的一种标准，由日本统计学家赤池弘次在1974年提出，它建立在熵的概念上，提供了权衡估计模型复杂度和拟合数据优良性的标准。AIC定义为



其中是模型参数个数，是模型的似然函数。从一组可供选择的模型中选择最佳模型时，通常选择AIC最小的模型。当两个模型之间存在较大差异时，差异主要体现在似然函数项，当似然函数差异不显著时，上式第一项，即模型复杂度则起作用，从而参数个数少的模型是较好的选择。

贝叶斯信息准则与AIC相似，用于模型选择，1978年由Schwarz提出。训练模型时，增加参数数量，也就是增加模型复杂度，会增大似然函数，但是也会导致过拟合现象，针对该问题，AIC和BIC均引入了与模型参数个数相关的惩罚项，BIC的惩罚项比AIC的大，考虑了样本数量，样本数量过多时，可有效防止模型精度过高造成的模型复杂度过高。BIC定义为



其中，为模型参数个数，为样本数量，为似然函数。惩罚项在维数过大且训练样本数据相对较少的情况下，可以有效避免出现维度灾难现象。

当拟合出相应的ARIMA模型后，通过AIC和BIC等其它指标从而选择出泛化性能强，预测效果好的模型。

## 算法设计

### ARIMA

ARIMA算法实现类，定义了实现ARIMA的相关参数、函数，其参数如下：

endog：list类型，对应输入时间序列

order：tuple类型，自回归，差分和移动平均值分量的模型的（p，d，q）值

trend：string类型，控制确定性趋势的参数，“c”表示常数项，“t”表示时间的线性趋势，而“ct”包括两者

fit函数根据给定时间序列数据拟合出对应的ARIMA模型

### ARIMAResults

为了方便模型的保存与加载及其各种参数的展示，定义了ARIMAResults类以封装模型和各类统计指标。

\_\_init\_\_函数接受模型和各类统计指标：

model：拟合的指数平滑模型

forecast（step）通过指定step大小以进行未来时间预测

predict函数通过给定的预测步长进行时间序列预测，其形参为：

params：ndarray类型，模型拟合的参数

start：int类型，模型预测初始时刻，默认为None，即拟合原始时间序列

end：int类型，模型预测结束时刻，默认为None，即拟合原始时间序列

通过这些定义的变量和静态方法，使得可以轻松地访问和展示该类的各项属性和统计指标。用户可以通过直观的方式获取关键属性的值，同时还提供了一些统计指标以帮助用户更好地理解和分析模型拟合情况。

# 指数平滑

## 原理介绍

指数平滑（Exponential Smoothing）在20世纪50年代后期被提出，是除了ARIMA之外的另一种被广泛使用的时间序列预测方法。指数平滑即指数移动平均（Exponential Moving Average，EMA），由移动平均法进一步发展而来，是一种改良的加权平均法，简单的全期平均法是对过去的所有时间数列数据进行平均；移动平均法则不考虑较远期的数据，并在加权移动平均中给予近期数据更大的权重；而指数平滑法则兼容了全期平均和移动平均所长，在不舍弃历史数据的前提下，对离预测期较近的历史数据给予较大的权数，权重由近到远按指数规律递减，指数平滑法根据本期的实际值和预测值，并借助于平滑系数α进行加权平均计算，预测下一期的值。它是对时间序列数据给予加权平滑，从而获得其变化规律与趋势。常用的指数平滑方法有一次指数平滑、二次指数平滑和三次指数平滑。

一次指数平滑适用于没有趋势和季节性特征的时间序列，二次指数平滑适用于有趋势特征但无季节性特征的时间序列，三次指数平滑适用于有趋势和季节性特征的时间序列。

一次指数滑动平均的预测方程与平滑方程见下式：





进一步推导可得到：



可以看出t时刻的预测结果跟前面所有时刻的观测结果都有关，且观测时刻越远，则其对应的权重就越小。的取值范围为0~1，值越大，越关注近期的观测值，远期的观测值影响越小，值越小，则近期观测值的影响越小，曲线越平滑。当时间序列相对平稳时，取较小的；当时间序列波动较大时，取较大的。

一次指数平滑法虽然克服了移动平均法的缺点。但当时间序列的变动出现直线趋势时，用一次指数平滑法进行预测，仍存在明显的滞后偏差。因此，必须加以修正。修正的方法与趋势移动平均法相同，即再作二次指数平滑，利用滞后偏差的规律建立直线趋势模型。这就是二次指数平滑法。其计算公式为：







当时间序列的变动表现为二次曲线趋势时，则需要用三次指数平滑法，它与二次指数平滑的区别就是三次平滑还考虑到了季节效应。与趋势一样，季节性也有加性（线性）或乘性（指数）过程。当季节变化在时间序列中大致保持不变时，通常选择加法模型；而当季节变化与时间序列的水平成比例变化时，通常选择乘法模型。三次指数平滑是在二次指数平滑的基础上，再进行一次平滑，其计算公式为：









## 算法设计

### ExponentialSmoothing

ExponentialSmoothing算法实现类，定义了实现ExponentialSmoothing的相关参数、函数。

初始化函数\_\_init\_\_及其参数：

endog：list类型，对应时间序列y

trend：string类型，是否有趋势特征，可选“add”、“mul”、“additive”、“multiplicative”、None，代表对应的加法趋势、乘性趋势、

damped\_trend：bool类型，对应趋势项是否有衰减

seasonal：string类型，是否有季节性特征，可选“add”、“mul”、“additive”、“multiplicative”、None，代表对应的加法趋势、乘性趋势

seasonal\_periods：int类型，季节性特征周期大小

initialization\_method：string类型，初始化递归的方法，可选“estimated”、“heuristic”“legacy-heuristic”、“known”，如果选择“known”，则随后的三个参数项需要给定

initial\_level：float类型，初始水平成分，当initialization\_method选择Known时需要显式指定大小

initial\_trend：float类型，初始趋势成分，当initialization\_method选择Known时需要显式指定大小

initial\_seasonal：list类型，初始季节性成分，当initialization\_method选择Known时需要显式指定大小

模型拟合函数fit（）及其参数：

smoothing\_level：float类型，指数平均模型的水平因子，未指定则自动拟合

smoothing\_trend：float类型，指数平均模型的趋势因子，未指定则自动拟合

smoothing\_seasonal：float类型，指数平均模型的季节性因子，未指定则自动拟合

damping\_trend：float类型，指数平均模型的衰减因子，未指定则自动拟合

optimized：bool类型，是否通过最大化对数似然函数来估计模型参数，为True则\_optimize\_parameters方法进行参数优化

method：string类型，参数优化方法，可选least\_squares、basinhopping

use\_brute：bool类型，是否通过暴力搜索确定初始值

initial\_level：float类型，初始水平成分

initial\_trend：float类型，初始趋势成分

值初始化函数\_initialize（）用来执行模型的属性初始化操作

具体初始化方法实现函数\_initialize\_heuristic、\_initialize\_legacy、\_initialize\_known根据模型初始化时method参数选择的不同执行不同的值初始化。

\_construct\_bounds函数用于构建参数的边界并检查边界的合法性

initial\_values函数用于计算初始水平、初始趋势、初始季节性分量值

\_optimize\_parameters函数用于优化模型参数，其接收形参如下：

data：自定义\_OptConfig类型，包含待优化参数的配置信息

use\_brute：bool类型，初始值选择是否采用暴力搜索

method：string类型，模型优化方法

\_predict函数实现了趋势预测，接受预测步长和一系列参数进行预测计算并返回预测结果和相关统计量，其接收形参为：

h：预测的时间步数。默认值为`None`，表示不进行未来预测，仅拟合模型，当给出整数时进行预测

smoothing\_level：float类型，指数平均模型的水平因子

smoothing\_trend：float类型，指数平均模型的趋势因子

smoothing\_seasonal：float类型，指数平均模型的季节性因子

initial\_level：float类型，初始水平分量

initial\_trend：float类型，初始趋势分量

damping\_trend：float类型，趋势分量的阻尼因子。它用于调整趋势的变化速率

initial\_seasons：list类型，初始季节性分量

is\_optimized：bool类型，模型是否被优化

predict函数通过给定的预测步长进行时间序列预测，其形参为：

params：ndarray类型，模型拟合的参数

start：int类型，模型预测初始时刻，默认为None，即拟合原始时间序列

end：int类型，模型预测结束时刻，默认为None，即拟合原始时间序列

### Holt

Holt算法实现类，定义了二次指数平滑的相关参数、函数。它继承了ExponentialSmoothing类

初始化函数\_\_init\_\_及其参数如下：

endog：list类型，对应时间序列y

exponential：bool类型，是否为乘性趋势

damped\_trend：bool类型，对应趋势项是否有衰减

initialization\_method：string类型，初始化递归的方法，可选“estimated”、“heuristic”“legacy-heuristic”、“known”，如果选择“known”，则随后的三个参数项需要给定

initial\_level：float类型，初始水平成分，当initialization\_method选择Known时需要显式指定大小

initial\_trend：float类型，初始趋势成分，当initialization\_method选择Known时需要显式指定大小

模型拟合函数fit（）及其参数：

smoothing\_level：float类型，指数平均模型的水平因子，未指定则自动拟合

smoothing\_trend：float类型，指数平均模型的趋势因子，未指定则自动拟合

damping\_trend：float类型，指数平均模型的衰减因子，未指定则自动拟合

optimized：bool类型，是否通过最大化对数似然函数来估计模型参数，为True则\_optimize\_parameters方法进行参数优化

method：string类型，参数优化方法，可选least\_squares、basinhopping

use\_brute：bool类型，是否通过暴力搜索确定初始值

initial\_level：float类型，初始水平成分

initial\_trend：float类型，初始趋势成分

### SimpleExpSmoothing

SimpleExpSmoothing算法实现类，定义了一次指数平滑的相关参数、函数。它继承了ExponentialSmoothing类

初始化函数\_\_init\_\_及其参数如下：

endog：list类型，对应时间序列y

initialization\_method：string类型，初始化递归的方法，可选“estimated”、“heuristic”“legacy-heuristic”、“known”，如果选择“known”，则随后的三个参数项需要给定

initial\_level：float类型，初始水平成分，当initialization\_method选择Known时需要显式指定大小

模型拟合函数fit（）及其参数：

smoothing\_level：float类型，指数平均模型的水平因子，未指定则自动拟合

optimized：bool类型，是否通过最大化对数似然函数来估计模型参数，为True则\_optimize\_parameters方法进行参数优化

method：string类型，参数优化方法，可选least\_squares、basinhopping

use\_brute：bool类型，是否通过暴力搜索确定初始值

initial\_level：float类型，初始水平成分

### HoltWintersResults

为了方便模型的保存与加载及其各种参数的展示，定义了HoltWintersResults类以封装模型和各类统计指标。

\_\_init\_\_函数接受模型和各类统计指标：

model：拟合的指数平滑模型

params：dict类型，指数平滑模型的所有参数

sse：float类型，误差平方和

aic：float类型，AIC信息准则值

aicc：float类型，AICC信息准则值

bic：float类型，BIC信息准则值

optimized：bool类型，模型是否经过优化

level：ndarray类型，拟合水平分量

trend：ndarray类型，拟合趋势分量

season：ndarray类型，拟合季节性分量

params\_formatted：pd.DataFrame类型，参数的格式化信息

resid：拟合值和实际值之间的残差

k：用于在AIC、BIC求解过程中移除偏差的k值

fittedvalues：ndarray类型，拟合值

fittedfcast：ndarray类型，拟合值和预测值的组合

fcastvalues：ndarray类型，预测值

forecast（step）通过指定step大小以进行未来时间预测

summary函数将各拟合参数值、统计指标以字符串形式打印

该类通过在初始化函数中定义这些变量并通过相应的静态方式以友好、方便的展示各类属性、统计指标。

# 支持向量机

## 原理介绍

支持向量机（Support Vector Machine，SVM）是一种二分类模型．它的基本模型是定义在特征空间上的间隔最大的线性分类器。支持向量机的学习策略就是间隔最大化，可形式化为一个求解凸二次规划（convex quadratic programming）的问题，也等价于正则化的合页损失函数的最小化问题．支持向量机的学习算法是求解凸二次规划的最优化算法。

### 基本定义

假设给定一组线性可分的训练数据集



其中，为第个特征向量，为的类标记，当时，称为正例，当时，称为负例，称为样本点。

学习的目标是要在特征空间中找到一个分离超平面将两类数据正确分开。分离超平面对应于方程，其中是垂直于超平面的一个法向量，为截距。分离超平面将特征空间中的训练数据集划分为两部分，一部分是正类，一部分为负类，法向量指向的一侧为正类，另一侧为负类。

如下图所示，现有一个二维平面，二维平面中有着线性可分的两类数据，因此可用一条直线将这两类数据分开，这条直线就相当于一个超平面。一般来说，当训练数据线性可分时，存在无数个分离超平面可将两类数据正确分开，就下图来说，我们可以得到无数条直线成功地将这两类数据分开。支持向量机利用间隔最大化求最优分离超平面，这时，解是唯一的，相应地解为



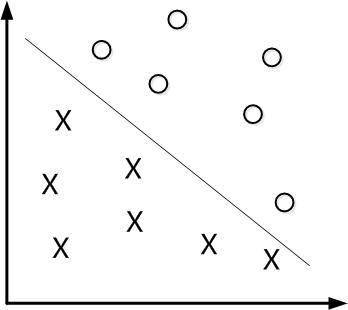


图7‑1 二维线性可分

超平面关于样本点的几何间隔为



定义超平面关于训练数据的几何间隔为超平面关于中所有样本点的几何间隔最小值



支持向量机学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面，对训练数据集找到几何间隔最大的超平面意味着以充分大的确信度对训练数据进行分类。也就是说，不仅将正负实例点分开，而且对最难分的实例点（离超平面最近的点）也有足够大的确信度将它们分开．这样的超平面应该对未知的新实例有很好的分类预测能力介绍原理。

上述求解几何间隔最大的分离超平面问题可定义为：





注意到



可将上式改写为





由于函数间隔的取值不影响最优化问题的解，因此令为1，此外，最大化与最小化是等价的，因此总结得到一下线性可分支持向量机学习的最优化问题





求得最优解，并由此得到分离超平面如下：



在线性可分情况下，训练数据集的样本点中与分离超平面距离最近的样本点的实例称为支持向量（support vector），在决定分离超平面时只有支持向量起作用，而其他实例点并不起作用。如果移动支持向量将改变所求的解，在间隔边界以外移动其他实例点，甚至去掉这些点，则解是不会改变的。由于支持向量在确定分离超平面中起着决定性作用，所以将这种分类模型称为支持向量机。支持向量的个数一般很少，所以支持向量机由很少的“重要的”训练样本确定。

需要注意的是，上述给定的是支持向量机的基本定义，现实世界中的数据很少能够直接线性可分，因此根据数据类型的不同，需要使用不同类型的支持向量机模型，常见的支持向量机分类模型如下：

* 当样本线性可分时，通过硬间隔最大化，学习一个线性可分支持向量机
* 近似线性可分时，通过软间隔最大化，学习一个线性可分支持向量机
* 线性不可分时，通过核技巧和软间隔最大化，学习非线性支持向量机

## 算法设计

### BaseLibSVM

svm算法实现类，包含了各种属性设置以及拟合svm模型方法，其主要参数和方法如下：

\_\_init\_\_初始化函数包含了要设置的参数

kernel：string类型，指定要在算法中使用的核函数类型。它必须是'linear'，'poly'，'rbf'，'sigmoid'，'precomputed'之一，其中'linear'为线性核函数，数学表示为；

'poly'为线性核函数，数学表示为，是gamma参数，是coef参数，是degree参数；

'poly'为多项式核函数，数学表示为，是gamma参数，是coef参数，是degree参数；

'rbf'代表径向基核函数，数学表示为，其中是gamma参数；

'sigmoid'为Sigmoid核函数，数学表示为，是gamma参数，是coef参数；

'precomputed'则代表预先计算好的核矩阵，如果您选择这个选项，那么您需要提供一个预先计算好的核矩阵作为输入数据

degree：int类型，当kernel为'poly'时，核函数的阶数，选择其它核函数时该参数无效

gamma：float或str类型，'rbf'，'poly'和'sigmoid'的核系数，对于多项式核函数和径向基函数核函数，该参数表示衰减速度的倒数。较小的gamma值会使得核函数的作用范围更广，较大的gamma值会使得核函数的作用范围更窄，如果类型为string，可取‘scale’、‘auto’，分别代表取值为，

coef0：float类型，核函数中的常数项。只在kernel为‘poly’或‘sigmoid’时有效

tol：float类型，定义模型停止拟合的误差值

C：float类型，正则化系数，C越大，越不能够容忍出现误差，越容易过拟合，C越小，越容易出现欠拟合

epsilon：float类型，定义模型对于错误分类的容忍度，值越大，模型允许错误分类的容忍度越高，反之，容忍度越小

max\_iter：int类型，设置默认迭代次数，取-1时代表不限制次数

fit函数使用给定的训练数据拟合出SVM模型

\_validate\_targets函数用于验证输入数据的形状或属性等参数是否合法

\_dense\_fit函数用于具体拟合SVR模型，被fit函数所调用

\_dense\_predict函数用于具体实现预测功能，被predict函数所调用

predict函数用于根据输入数据X进行预测

\_validate\_for\_predict函数用于验证输入的预测数据X。它会检查X的维度是否与训练数据匹配，并进行必要的数据转换

### SVR

支持向量回归机算法模型对象类，该类设置了使用支持向量机进行回归预测的各种参数默认值，包括：

kernel='rbf'

degree=3

gamma='scale'

coef0=0.0

tol=1e-3

C=1.0

epsilon=0.1

max\_iter=-1

# 随机森林

## 原理介绍部分

### 集成学习

集成学习（Ensemble Learning）是目前非常流行的机器学习策略，基本上所有问题都可以借用其思想来得到效果上的提升。基本出发点就是把算法和各种策略集中在一起。集成学习既可以用于分类问题，也可以用于回归问题。

集成算法通常分为两类：bagging和boosting。

bagging即bootstrap aggregating，其中bootstrap是一种有放回的抽样方法，每个预测器使用的算法相同，在不同的训练集子集上训练，采样时样本放回，最终得到多个预测器，最后的结果由多个预测共同决定。

boosting是一簇可将弱学习器提升为强学习器的算法。其工作机制为：先从初始训练集训练出一个基学习器，再根据基学习器的表现对样本分布进行调整，使得先前的基学习器做错的训练样本在后续收到更多的关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器数目达到实现指定的值，或整个集成结果达到退出条件，然后将这些学习器进行加权结合。

而随机森林则是一种经典的bagging集成学习算法，每个预测器都是独立的决策树，但是每个预测器使用的训练数据有所不同。

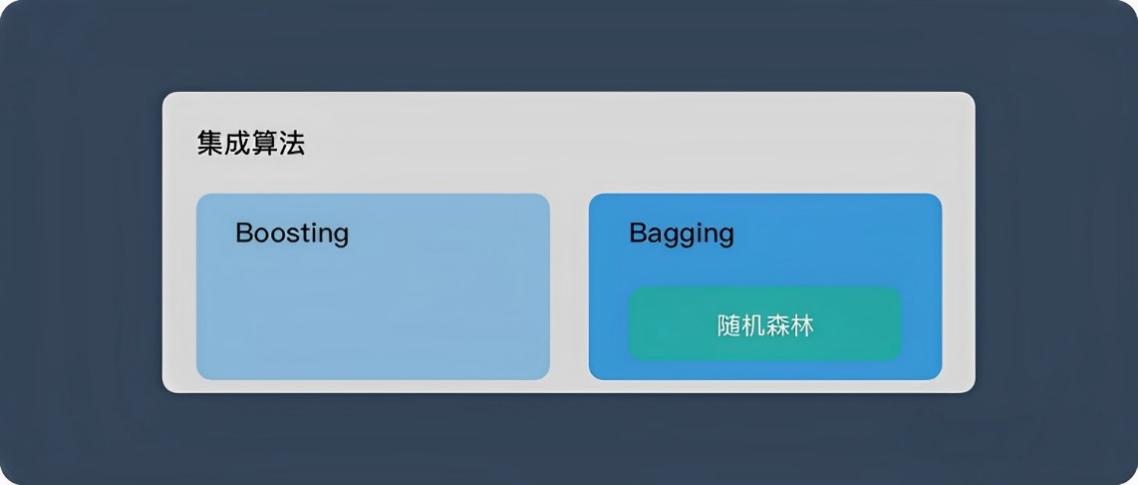


图8‑1 集成学习算法分类

### 决策树

决策树（decision tree）是一种基本的分类与回归方法，它可以认为是if-then-else规则的集合，也可以认为是定义在特征空间与类空间上的条件概率分布。其主要优点是模型具有可解释性较强，符合人类直观思维。学习时，利用训练数据，根据损失函数最小化的原则建立决策树模型。预测时，对新的输入数据，利用已经拟合好的决策树模型进行分类或回归。

决策树算法采用树形结构，使用层层推理来实现最终的分类。其结构示意图如图8‑2所示，决策树由下面几种元素构成：

根节点：包含样本的全集

内部节点：对应样本特征或属性

叶节点：代表决策的结果

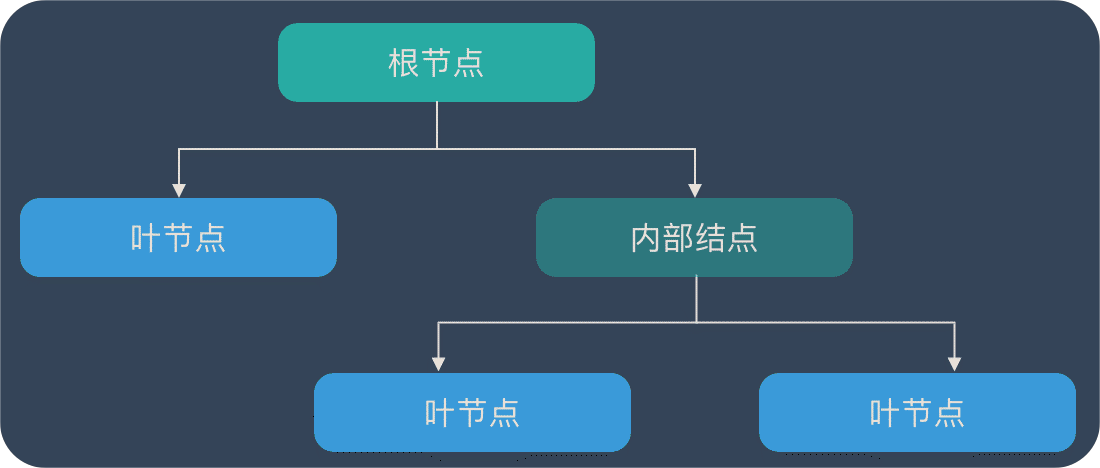


图8‑2 决策树结构示意图

构建好的决策树从根节点到叶节点有着不同的路径，路径上内部结点的特征对应着规则的条件，而叶结点的类对应着规则的结论。预测时，在树的内部节点处用某一属性值进行判断，根据判断结果决定进入哪个分支节点，直到到达叶节点处，得到分类或者是回归的结果。

决策树学习本质上是从训练数据集中归纳出一组分类规则。与训练数据集不相矛盾的决策树（即能对训练数据进行正确分类的决策树）可能有多个，也可能一个也没有。我们需要拟合出一个与训练数据矛盾较小的决策树，同时具有很好的泛化能力。

决策树学习通常包括3个步骤：特征选择、决策树的生成和决策树的修剪。

特征选择：特征选择决定了使用哪些特征来做判断。在训练数据集中，每个样本的属性可能有很多个，不同属性的作用有大有小。因而特征选择的作用就是筛选出跟分类或回归结果相关性较高的特征，也就是表征力较强的特征。在构建决策时前，需要进行特征选择，特征选择往往根据信息增益的大小来决定，而信息增益的大小计算与熵有着紧密的联系。在信息论与概率统计中，熵（entropy）是表示随机变量不确定性的度量。设是一个取有限个值的离散随机变量，其概率分布为



则随机变量的熵定义为



当随机变量只取两个值，例如1,0时，即的分布为



熵为



熵随着概率变化曲线如图8‑3所示

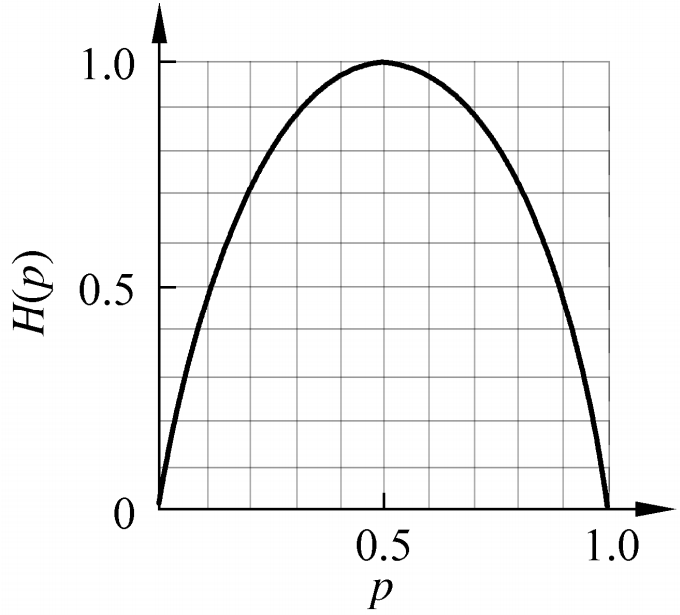


图8‑3 分布为贝努利分布时熵与概率的关系

当或时，随机变量完全没有不确定性。当时，，熵取值最大，随机变量不确定性最大。

信息增益（信息增益表示得知特征A的信息而使得类Z的信息的不确定性减少的程度），其计算公式由下式给出：







其中为训练数据集，为样本数，为样本类别，为某个样本类别数，根据特征将划分为个子集，为的样本数目，子集中属于类的样本集合为，为的样本数目。

决策树生成：选择好特征后，就从根节点出发，对节点计算所有特征的信息增益，选择信息增益最大的特征作为节点特征，根据该特征的不同取值建立子节点；对每个子节点使用相同的方式生成新的子节点，直到信息增益很小或者没有特征可以选择为止，生成决策时的常用算法有ID3、C4.5与CART。

ID3算法基本思路为选择信息增益最大的特征作为根结点，在子集中计算信息增益最大的特征，若增益小于ε，则返回实例数最大的类为叶结点，否则划分为一个内部结点，根据划分的训练集划分下一个结点。

C4.5算法则是根据信息增益比作为结点划分依据。

CART算法对于回归树来说采用平方误差最小准则来划分叶结点，分类树采用基尼指数最小化准则来划分叶结点

决策树的剪枝：剪枝的主要目的是对抗过拟合，通过上述方法生成的决策树只考虑了通过提高信息增益（或信息增益比）对训练数据进行更好的拟合，生成的决策树可能对训练数据有很好的分类或回归能力，但对未知的测试数据却未必有很好的分类或回归能力，即可能发生过拟合现象。我们需要对已生成的树自下而上进行剪枝，将树变得更简单以降低模型复杂度，从而使它具有更好的泛化能力。

具体地，就是去掉过于细分的叶结点，使其回退到父结点，甚至更高的结点，然后将父结点或更高的结点改为新的叶结点。

如果特征数量很多，也可以在决策树学习开始的时候，对特征进行选择，只留下对训练数据有足够分类能力的特征。通过主动去掉部分分支来降低过拟合的风险。

### 随机森林

随机森林是一种基于集成学习和Bagging思想的算法，用于解决分类和回归问题。它的核心组成部分是决策树，而整个算法则是通过集成多棵决策树来提高预测性能。

随机森林的名称中蕴含着两个关键词，一个是“随机”，一个是“森林”。“随机”表示在构建每棵决策树时引入了随机性，这样可以降低过拟合的风险，增加模型的泛化能力。具体来说，随机森林通过在每棵树的训练过程中随机选择特征和样本进行建模，使得每棵树都有一定的差异性，从而提升整个模型的稳健性和准确性。

而“森林”则表明随机森林由多棵决策树组成，形成了一个强大的集成模型。每棵树都是通过对不同的样本和特征进行训练而生成的，它们相互独立地进行预测，并通过投票或平均等方式综合得出最终的预测结果。这样的集成策略能够有效地减少个别决策树的偏差和方差，提高模型的整体性能。

当要将一个输入样本进行分类时，需要将它输入到每棵树中进行分类操作。将若干个弱分类器的分类结果进行投票选择，从而组成一个强分类器，这就是随机森林bagging的思想：

（1）如果训练集大小为N，对于每棵树而言，随机且有放回地从训练集中的抽取n个训练样本（（n＜N））作为该树的训练集；从这里我们可以知道：每棵树的训练集都是不同的，而且里面包含重复的训练样本。

（2）如果存在M个特征，则在每个节点分裂的时候，从M中随机选择m个特征维度（m＜M），使用这些m个特征维度中最佳特征(最大化信息增益)来分割节点。在森林生长期间，m的值保持不变。

## 算法设计

### 算法设计流程

由于项目中使用随机森林进行回归预测，因此特征选择函数可选为均方误差（MSE）或（MAE），默认选择MSE，即针对某个划分节点有



随机森林算法模型的构建如图8‑4所示

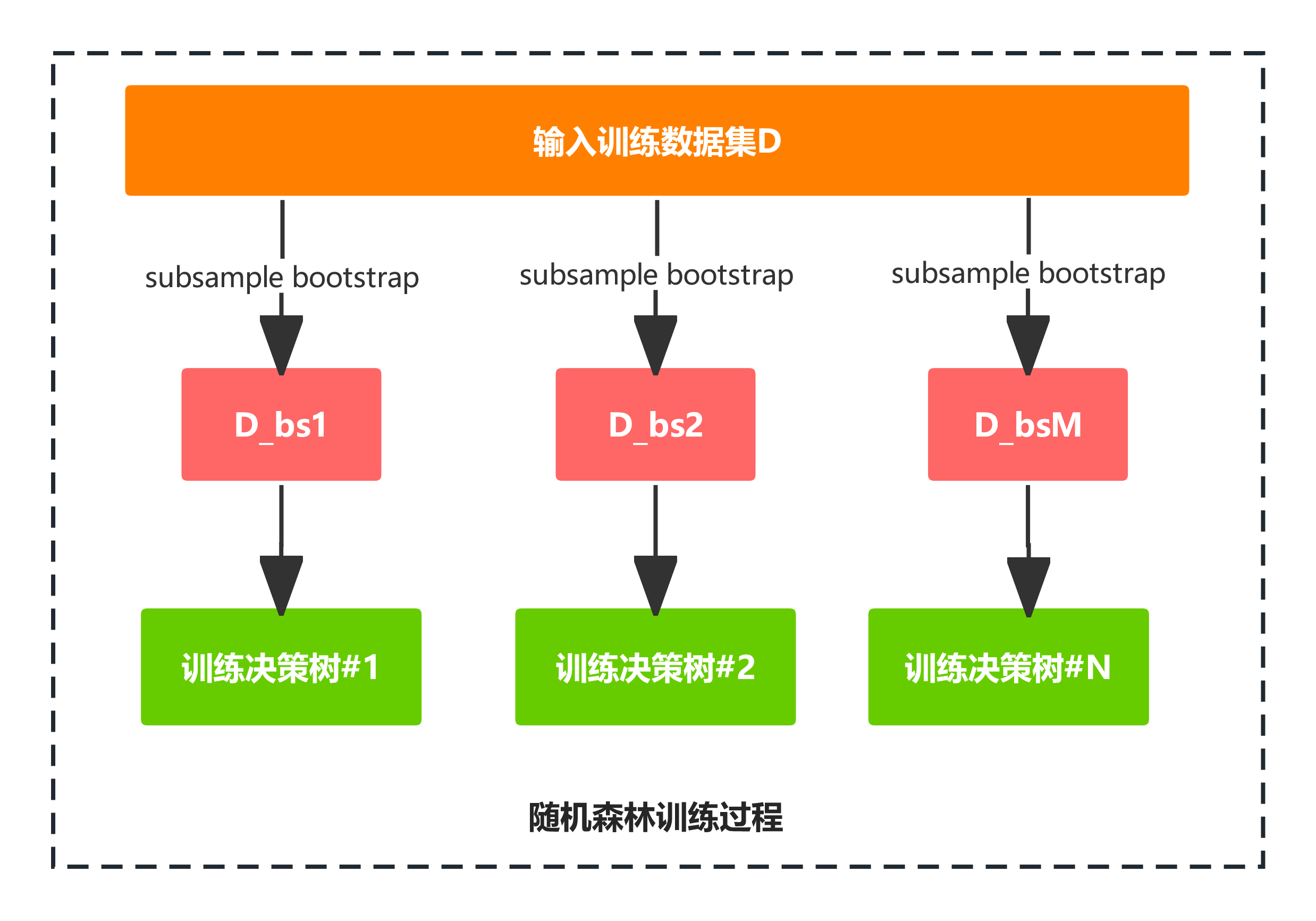


图8‑4 随机森林训练示意图

bootstrap对输入训练样本集合进行N次放回重复抽样得到样本集合。在对训练集进行bootstrap采样时，每个样本会以63.3%的概率被抽中。bootstrap使用多个不同的子模型来增加最终模型预测结果的鲁棒性和稳定性(即减小方差)，注意如果多个子模型都采用同样的数据集训练，那么训练得出的子模型都是相同的，集成学习将变得没有意义。

### BaseEnsemble

随机森林是一种集成学习算法，该类定义了基类算法的基本属性和一些函数，便于后期拓展实现不同的集成学习算法，具体包括的函数与参数如下：

初始化\_\_init\_\_函数及其参数：

base\_estimator：对象，集成模型构建的基础算法类别

n\_estimators：int类型，构建基础算法模型数量

estimator\_params：tuple类型，用于实例化新基本估计器的属性列表

\_validate\_estimator函数用于用于检查基本估计器和n\_estimators属性的有效性，并设置base\_estimator\_属性

\_make\_estimator函数创建和配置基本估计器的副本

\_\_len\_\_函数返回基类估计器的数量

\_\_getitem\_\_函数根据索引返回某个估计器

### BaseDecisionTree

基础的决策树类实现，定义了决策树的各种属性与方法，具体如下：

\_\_init\_\_函数定义了生成一颗决策树需要的各种属性包括：

criterion：string，划分准则，用于选择最佳的特征来进行节点划分，可选gini、entropy、等，对于决策树回归设置为mse

splitter：str决策树节点划分策略，可选"best"（选择最佳划分特征）或"random"（随机选择划分特征）

max\_depth：int决策树的最大深度限制。如果设置为None，表示不限制最大深度

min\_samples\_split：int进行节点划分所需的最小样本数

min\_samples\_leaf：int叶节点所需的最小样本数

min\_weight\_fraction\_leaf：float叶节点所需的最小样本权重总和

max\_features：int、float或者"auto"、"sqrt"、"log2"在寻找最佳划分时考虑的特征数量

random\_state：随机种子，用于控制随机性

max\_leaf\_nodes：最大叶节点数量限制，如果为None，则不限制叶节点数量

min\_impurity\_decrease：float划分的最小不纯度减少量，浮点数

min\_impurity\_split：float划分的最小不纯度阈值，浮点数

ccp\_alpha：float用于剪枝的复杂度参数

get\_depth函数用于返回决策树深度

get\_n\_leaves函数用于返回叶节点数量

fit函数接收输入X和标签y拟合出单颗决策树

predict函数接收输入X进行预测

### DecisionTreeRegressor

回归决策树实现，该类继承了BaseDecisionTree，该类的一些属性和方法如下：

criterion：string，划分准则，用于选择最佳的特征来进行节点划分，可选gini、entropy、等，对于决策树回归设置为mse

splitter：str决策树节点划分策略

可选"best"（选择最佳划分特征）或"random"（随机选择划分特征）

max\_depth：int决策树的最大深度限制。如果设置为None，表示不限制最大深度

min\_samples\_split：int进行节点划分所需的最小样本数

min\_samples\_leaf：int叶节点所需的最小样本数

min\_weight\_fraction\_leaf：float叶节点所需的最小样本权重总和

max\_features：int、float或者"auto"、"sqrt"、"log2"在寻找最佳划分时考虑的特征数量

random\_state：随机种子，用于控制随机性

max\_leaf\_nodes：最大叶节点数量限制，如果为None，则不限制叶节点数量

min\_impurity\_decrease：float划分的最小不纯度减少量，浮点数

min\_impurity\_split：float划分的最小不纯度阈值，浮点数

ccp\_alpha：float用于剪枝的复杂度参数

fit函数接收输入X和标签y拟合出单颗回归决策树

### BaseForest

继承自BaseEnsemble，构建随机森林基础类，包括参数如下：

base\_estimator：对象，集成模型构建的基础算法类别

n\_estimators：int类型，构建基础算法模型数量

estimator\_params：列表，相应参数设置

bootstrap：bool类型，是否有放回的随机选取样本

oob\_score=bool类型，是否采用袋外样本来评估模型的好坏

n\_jobs=int类型，并行计算数。默认是None。一般选择-1,根据计算机核数自动选择

random\_state=int类型,控制bootstrap的随机性以及选择样本的随机性

max\_samples=None，如果bootstrap为True，则从X中抽取的样本数量，以训练每个基础估计器

### ForestRegressor

继承自BaseForest，基于决策树类算法进行回归预测的基础类，其主要函数如下：

\_\_init\_\_函数定义了生成一颗决策树需要的各种属性包括：

base\_estimator：指定每个基础估计器的类型，即用于构建随机森林的基本单元的算法。

n\_estimators：随机森林中基础估计器的数量，默认为100。

estimator\_params：用于传递给基础估计器的额外参数的元组，默认为空元组。

bootstrap：指定是否使用自助采样（bootstrap sampling）来构建每个基础估计器的训练集。默认为False，即不使用自助采样。

oob\_score：指定是否计算袋外（out-of-bag）得分。默认为False，即不计算袋外得分。

random\_state：控制随机性的种子值。默认为None。

warm\_start：指定是否重用上一次调用的解决方案作为初始拟合。默认为False，即不重用解决方案。

max\_samples：指定每个基础估计器的最大采样数量。默认为None，表示使用整个训练集作为采样。

predict函数接收输入X进行预测

### RandomForestRegressor

继承自ForestRegressor，最终的随机森林回归算法实现类。它使用数据集的不同子样本上拟合出一些分类决策树，并使用平均法来提高预测精度并控制过度拟合，其主要函数如下：

\_\_init\_\_函数定义了生成随机森林模型的一些默认值参数包括：

n\_estimators：int类型，构建基础算法模型数量。默认值为100

criterion：string类型，节点划分标准，由于是回归任务，因此可选为“mse”或“mae”，默认选择“mse”

max\_depth：int类型，决策树最大深度，默认可以不输入，如果不输入的话，决策树在建立子树的时候不会限制子树的深度。一般来说，数据少或者特征少的时候可以不管这个值。如果模型样本量多，特征也多的情况下，推荐限制这个最大深度，具体的取值取决于数据的分布。常用的可以取值10-100之间

min\_samples\_split：int类型，内部节点再划分所需最小样本数，这个值限制了子树继续划分的条件，如果某节点的样本数少于min\_samples\_split，则不会继续再划分。默认是2。如果样本量数量级非常大，则推荐增大这个值

min\_samples\_leaf：int类型，叶子节点最少样本数，这个值限制了叶子节点最少的样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝，只保留原来的父节点。默认是1。如果样本量数量级非常大，则推荐增大这个值

min\_weight\_fraction\_leaf：int类型，叶子节点最小的样本权重和，这个值限制了叶子节点所有样本权重和的最小值，如果小于这个值，则会和兄弟节点一起被剪枝，只保留原来的父节点。 默认是0，就是不考虑权重问题。如果我们有较多样本有缺失值，或者分类树样本的分布类别非常不平衡，就会引入样本权重

max\_features=None，决策树划分时考虑的最大特征数，可以指定为具体特征数量，也可选“auto”、“sqrt”、“log2”，其中“auto”代表最大特征数为n\_features，“sqrt”代表最大特征数为，“log2”代表最大特征数为

max\_leaf\_nodes：int类型，最大叶子节点数，通过限制最大叶子节点数，可以防止过拟合，默认是“None”，即不限制最大的叶子节点数。如果加了限制，算法会建立在最大叶子节点数内最优的决策树

min\_impurity\_decrease：float类型，节点划分最小不纯度，如果某节点的不纯度(基于基尼系数，均方差)小于这个阈值，则该节点不再生成子节点。即为叶子节点

min\_impurity\_split：float类型，用于分割节点的最小不纯度阈值。当节点的不纯度小于此值时，停止划分并将其视为叶节点。如果设置值过高，将会导致过度拟合；如果设置值过低，则会导致欠拟合

bootstrap：bool类型，是否有放回的随机选取样本

oob\_score：bool类型，默认值为False，指定是否使用袋外（Out-of-Bag）样本来评估回归器的泛化性能，当oob\_score参数设置为True时，随机森林回归器在训练完成后会使用袋外样本计算一个袋外得分（OOB Score）

n\_jobs：int类型，并行计算数。默认是None。一般选择-1,根据计算机核数自动选择

random\_state：int类型，控制bootstrap的随机性以及选择样本的随机性

verbose：int类型，用于控制输出的详细程度，默认值为0，代表模型拟合过程中不打印任何输出，当大于1时则会输出一些训练过程中的进展信息

warm\_start：bool类型，是否在前一阶段的训练结果上继续训练

ccp\_alpha：float类型，用于最小成本-复杂度修剪的复杂度参数，将选择成本复杂度最大且小于ccp\_alpha的子树。默认情况下，不进行修剪

max\_samples：int或者float类型，如果bootstrap为True，则从X中抽取的样本数量，以训练每个基础估计器，当指定int时则代表具体样本数量，当指定0~1之间的浮点数时则代表抽取样本数量百分比

# 卷积神经网络

## 引言

神经网络，一种受生物学启发的编程范式，可以让计算机从观测数据中进行学习，是一种仿生模型，模拟了生物神经网络。

在生物神经网络中，每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，从而改变神经元内的电位，如果某个神经元的电位超过了一个[阈值](https://so.csdn.net/so/search?q=%E9%98%88%E5%80%BC&spm=1001.2101.3001.7020)，那么它就会被激活，即“兴奋”起来，向其他神经元发送化学物质。神经网络借鉴了这样的结构，每一个神经元接受输入x，通过带权重w的连接进行传递，将总输入信号与神经元的阈值进行比较，最后通过激活函数处理确定是否激活，并将激活后的计算结果y输出。一个简单的三层神经网络如图9‑1所示。

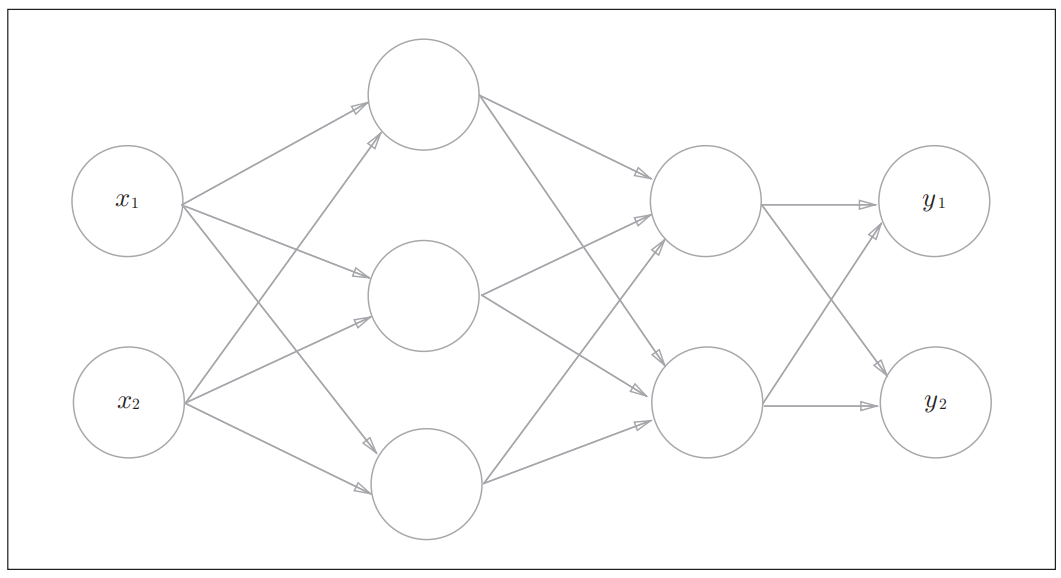


图9‑1 三层神经网络模型

神经网络是是实现机器学习的一种方法，它是深度学习的基础，这种网络依靠系统的复杂程度，通过调整内部大量节点之间相互连接的关系，从而达到处理信息的目的。前馈神经网络与反馈神经网络是神经网络的两种不同类别，对于前馈神经网络来说，当前层的输入只依赖于上一层的输出，与更早的网络输出状态无关；对于反馈神经网络来说，输出经过一步时移再进入到输入层，输入不仅仅取决于上一层的输出。图9‑2展示了神经网络的基本分类情况。其中的卷积神经网络依旧是层级网络，只是层的功能和形式做了变化，可以说是传统神经网络的一个改进。简而言之，卷积神经网络（Convolutional Neural Networks，CNN）是一种深度学习模型或类似于人工神经网络的多层感知器，常用来分析视觉图像。

相较于传统神经网络它主要有以下优点：

(1)局部连接：对于卷积神经网络来说，相邻两层神经元之间只有部分节点连接，在卷积层中利用卷积核与输入图像的局部区域进行运算，提取输入图形的部分特征，在最后一层将前面提取到的特征信息进行整合就得到了完整的特征信息。稀疏连接能够有效地减小了神经网络结构中神经元和权值的数量，从而提高卷积神经网络的运算效率。

(2)权值共享：在卷积神经网络中，不同的图像或者同一张图像可以共用一个卷积核，卷积核的参数是共享的。不同的卷积核负责提取不同的特征，不同的图像中可能具有相同的特征，因此可以使用同一个卷积核来提取这种特征。权值共享可以进一步减少卷积神经网络中的参数数量，从而提高网络运算效率。

(3)下采样：由于卷积神经网络中提取出的特征图数量较多，即使通过局部连接与权值共享降低了参数量，但是由于特征图的数量较多，因此参数量仍然较为庞大，参数量过多不仅会增加计算的难度，还会导致过拟合等问题导致网络的泛化性较差。为了进一步降低卷积神经网络中的参数量，同时在一定程度上抑制过拟合，卷积神经网络通过下采样操作来进一步降低参数量。下采样本质上是提取局部区域中感兴趣的参数，对于那些不重要的参数，则直接丢弃。

卷积神经网络可以直接输入原始图像进行一系列工作，提取出一系列特征，不仅包括线条、形状等常见的特征信息，还包括一些例如光泽、纹理等更深层次的特征信息，正是由于卷积神经网络在图像特征提取方面具有十分突出的效果，因此使用卷积神经网络来进行有关图片特征提取方面的工作是十分常见且有效的。

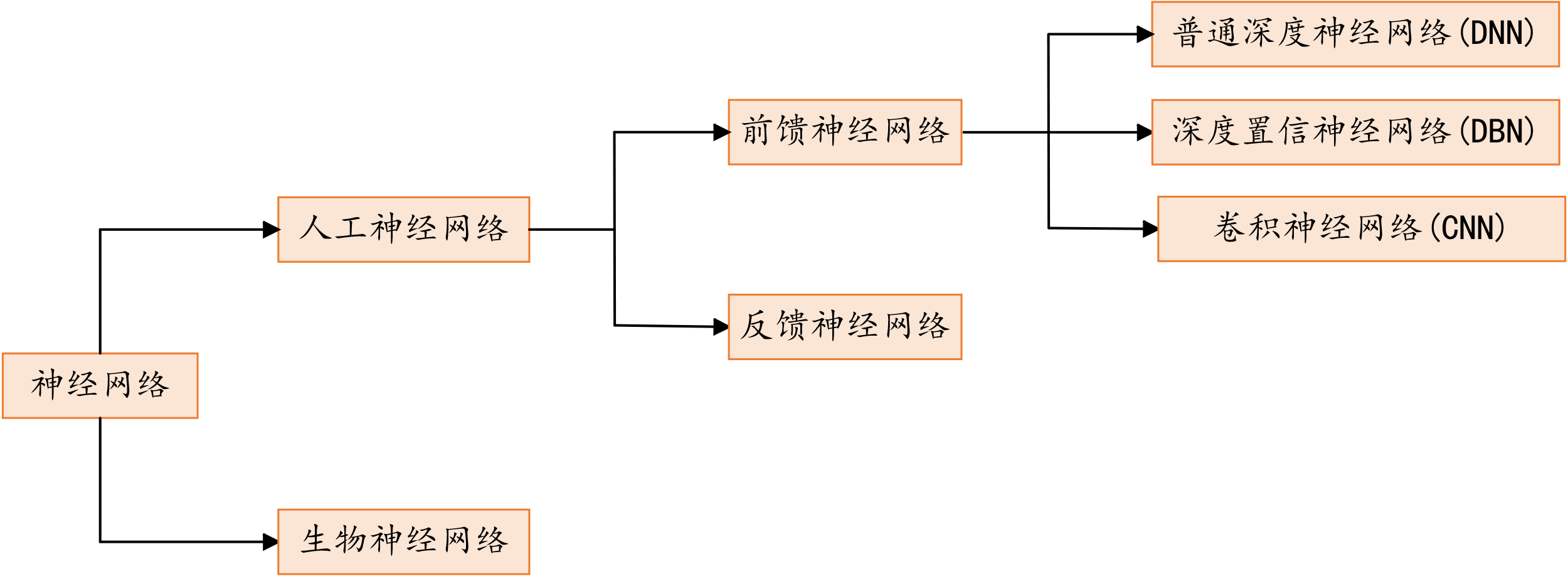


图9‑2 神经网络分类

## 相关理论

### 卷积

首先输入一张图片，然后对图片进行卷积操作，该过程中使用的是不同大小的三维卷积核（对应输入图片的3个通道），卷积核有着数值各异的权值，卷积核前两维一般是3×3或者5×5，反应了每次对原图片卷积区域的大小，第三维为对应输入的通道数，在此处输入的是RGB图片，对应通道数应为3，卷积核的数量则代表了提取图像的特征种类，例如，对于上面的VGG模型来说，第一次卷积，有64个3×3×3的卷积核对图片进行卷积操作，得到的是112×112×64的特征图。图9‑3展示了一个大小为3×3的卷积核进行卷积操作的过程操作，大小为5×5的图片经过卷积后得到3×3的特征图，左上角的4通过1×1+1×0+1×1+0×0+1×1+1×0+0×1+0×0+1×1计算得到。



图9‑3 卷积操作过程

### 池化

池化一般跟在卷积层的后面，其目的是为了降低卷积神经网络的维数，减少参数量，同时在一定程度上防止过拟合，池化操作一般可分为最大池化（max-pooling）与平均池化（mean-pooling），实际操作中采用最大池化操作较多，因为在特征提取的过程一般希望保留特征图中数字较大的部分，数字较大意味着可能探测到了某些特定的特征，如图9‑4所示，使用2×2的滤波器，在2×2的区域中，选择最大的数字作为输出，通过该操作，图9‑4中4×4的特征图变为了2×2的特征图。这里需要注意的是池化层和卷积层不同，没有要学习的参数。池化只是从目标区域中取最大值（或者平均值），所以不存在要学习的参数，而且经过池化运算，输入数据和输出数据的通道数不会发生变化。

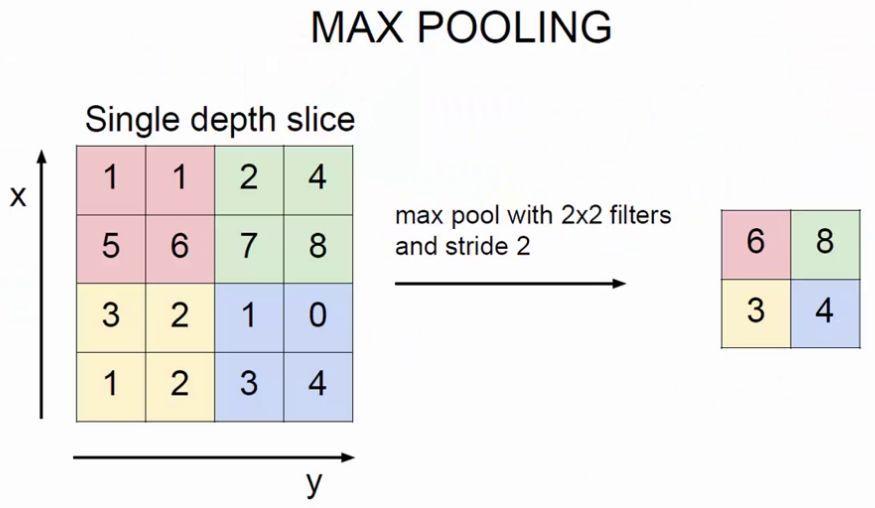


图9‑4 最大池化过程

## 算法设计

趋势预测算法设计目的，旨在利用传感器数据准确估计子系统或组件的未来运行趋势或者是剩余使用寿命。然而，现有算法大多基于线性模型，无法捕捉传感器数据与RUL之间的复杂关系，且由于其网络结构，无法自动学习显著特征。本章将趋势预测问题视为多元时间序列回归，设计了一种基于深度卷积神经网络（CNN）的趋势预测方法。

基于CNN在计算机视觉、自然语言处理、语音识别等任务应用的思想，CNN的关键属性是交替地进行不同的处理单元（例如卷积、池化、Sigmoid/双曲正切函数压缩、整流和归一化）。这种多样化的处理单元可以有效地表示信号的局部显著性。此外，深度架构允许多层这些处理单元堆叠在一起，使得这个深度学习模型可以表征不同尺度上的信号显著性。因此，CNN提取的特征是任务相关的，非手工设计的，具有更强的预测能力。

与用于计算机视觉的现有CNN结构不同，本章方法在时间维度上沿多通道传感器数据应用卷积和池化滤波器，以系统化地融入原始传感器信号的自动特征学习。通过深度架构，学到的特征是低级原始传感器信号的更高级抽象表示，以此实现特征学习和趋势预测的监督反馈增强。具体模型结构如下：

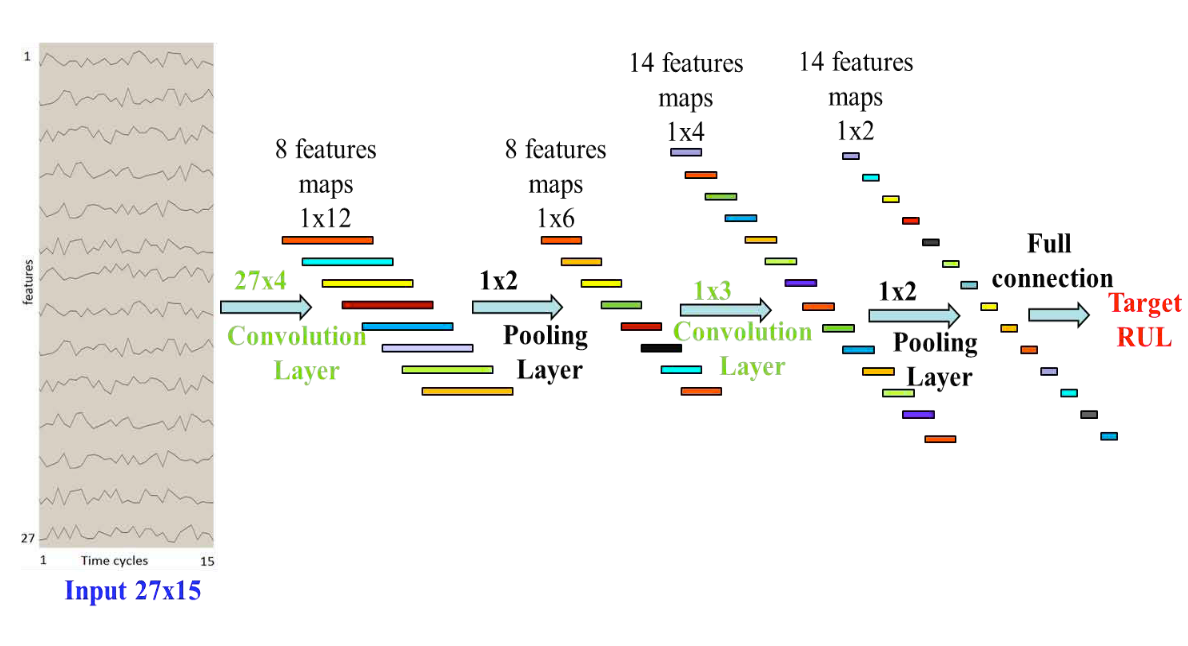


图9‑5 CNN趋势预测模型结构

## 算法实现

### nn.Module

所有的网络层都是继承于这个类。

nn.module中有八个重要的属性用于管理整个类，主要关注其中的parameters和modules两个属性。

parameters：管理存储属于nn.Parameter类的属性，例如权值或者偏置参数；

modules：用来存储管理nn.Module类，例如在LeNet中会构建子模块，modules就会存储创建的卷积层等；

buffers：存储管理缓冲的属性，如训练过程中BN的均值，或者是方差都会存储在buffers

\*\_hooks：存储管理钩子函数

nn.Module的属性构建会在module类中进行属性赋值的时候会被setattr()函数拦截，在这个函数当中会判断即将要赋值的数据类型是否是nn.parameters类，如果是的话就会存储到parameters字典中；如果是module类就会存储到module字典中

### \_AvgPoolNd基类

该类是nn.Module的子类，用于实现平均池化操作。

#### extra\_repr函数

功能描述：用于返回该类的初始化参数的字符串表示的方法函数。

输出描述：

kernel\_size：池化窗口的大小。

stride：池化窗口的步幅大小。

padding：输入的每一条边补充0的层数。

### AvgPool1d类

池化函数具体实现类，继承\_AvgPoolNd基类，通过\_\_init\_\_函数完成初始化，通过\_AvgPoolNd基类中的extra\_repr函数显示初始化参数，并调用F.conv2d()进行卷积运算

#### \_\_init\_\_函数

功能描述：该函数用于初始化AvgPool1d类的实例，并设置了池化窗口大小、步幅、填充等相关参数。

参数描述：

kernel\_size：\_size\_1\_t类型，池化窗口的大小，可以是一个整数。

stride：\_size\_1\_t类型，池化窗口的步幅大小，可以是一个整数或者None。如果为None，则默认与kernel\_size相同。

padding：\_size\_1\_t类型，输入的每一条边补充0的层数，可以是一个整数。

ceil\_mode：bool类型，是否使用天花板除法（向上取整）来计算输出形状，默认为False，表示使用地板除法（向下取整）。

count\_include\_pad：bool类型，在计算池化值时是否包括填充值，默认为True，表示包括填充值。

#### forward函数

功能描述：这个函数是AvgPool1d类的前向传播函数，通过F.avg\_pool1d()函数来执行平均池化操作。

参数描述：

input：输入的张量数据。

self.kernel\_size：池化窗口的大小。

self.stride：池化窗口的步幅大小。

self.padding：输入的每一条边补充0的层数。

self.ceil\_mode：是否使用天花板除法（向上取整）来计算输出形状。

self.count\_include\_pad：在计算池化值时是否包括填充值。

### \_convNd基类

该类是nn.Module的子类，用于实现卷积层的基本功能。

#### \_\_init\_\_函数

功能描述：\_ConvNd类的初始化函数。在函数内部，首先判断输入通道数是否能被分组数整除，以及输出通道数是否能被分组数整除。然后判断填充模式是否有效，如果是字符串类型的填充，则检查是否为same或valid，如果是same，则检查步长是否为1，如果不是，则抛出异常。最后，根据是否转置初始化卷积核权重和偏置项，并调用reset\_parameters()函数对它们进行初始化。

参数介绍：

in\_channels：int，输入张量的通道数

out\_channels：int，卷积核的数量，即输出通道数

kernel\_size：Tuple[int, ...]，卷积核的大小

stride：Tuple[int, ...]，卷积核在每个空间维度的步长

padding：Tuple[int, ...]，在输入张量的每个空间维度上增加的零填充数量

dilation：Tuple[int, ...]，卷积核元素之间的间隔

transposed：bool，是否为转置卷积，即反卷积

output\_padding：Tuple[int, ...]，输出张量每个空间维度上增加的大小，用于转置卷积

groups：int，输入和输出通道之间的分组数

bias：bool，是否添加偏置

padding\_mode：str，可选的填充模式。如果为'zeros'，则表示使用零填充。如果为'reflect'、'replicate'或'circular'，则表示使用不同类型的填充方式。

device：torch.device，表示所需设备的名称或索引。默认值为None。

dtype：torch.dtype，表示所需张量的数据类型。默认值为None。

#### reset\_parameters 的函数

功能描述：对卷积层的权重和偏置进行初始化。使用了Kaiming He提出的一种初始化方法kaiming\_uniform\_来对权重进行初始化，其中a=sqrt(5)时等价于使用均匀分布在[-1/sqrt(k), 1/sqrt(k)] 区间内的随机数来初始化权重，k是权重张量的第二维大小（即输入通道数in\_channels与卷积核大小的乘积），并使用 PyTorch 中的 init 模块来实现初始化。如果卷积层包含偏置项，则使用均匀分布在 [-bound, bound] 区间内的随机数对偏置项进行初始化，其中 bound 的计算方式与 kaiming\_uniform\_ 中的相同。注意，这个函数在调用之前需要确保卷积层的权重和偏置已经被正确地创建，并且这个函数只需要在模型初始化时被调用一次，而不需要在每次前向传播时都调用。

参数介绍：

self.weight: 模型的权重参数，形状为[out\_channels, in\_channels , groups, \*kernel\_size]，类型为torch.Tensor。

self.bias: 模型的偏置参数，形状为[out\_channels]，类型为torch.Tensor或None。

#### extra\_repr(self)函数

功能描述：这是一个用于打印出当前卷积层对象的可读表示的方法，它返回一个包含卷积层各种参数信息的字符串。其中，如果填充、膨胀、输出填充等参数不是默认值，它们也会被打印出来。该方法的实现利用了Python中字符串格式化的功能，通过替换字符串中的占位符来将各个参数信息组合成一个字符串。

参数介绍：

in\_channels (int): 输入数据的通道数。

out\_channels (int): 输出数据的通道数。

kernel\_size (Tuple[int, ...]): 卷积核的大小，以每个维度的尺寸表示。

stride (Tuple[int, ...]): 卷积核在每个维度上的步长大小。

padding (Tuple[int, ...]): 每个维度上卷积核的填充大小。

dilation (Tuple[int, ...]): 卷积核中每个元素之间的间距大小。

output\_padding (Tuple[int, ...]): 转置卷积输出的每个维度上的额外大小。

groups (int): 输入和输出通道之间的连接数量。

bias (bool): 是否包含偏置项。

padding\_mode (str): 如果使用填充，使用哪种填充方式。

输出描述：

in\_channels (int): 输入数据的通道数。

out\_channels (int): 输出数据的通道数。

kernel\_size (Tuple[int, ...]): 卷积核的大小，以每个维度的尺寸表示。

stride (Tuple[int, ...]): 卷积核在每个维度上的步长大小。

padding (Tuple[int, ...]): 每个维度上卷积核的填充大小，如果为0则省略。

dilation (Tuple[int, ...]): 卷积核中每个元素之间的间距大小，如果为1则省略。

output\_padding (Tuple[int, ...]): 转置卷积输出的每个维度上的额外大小，如果为0则省略。

groups (int): 输入和输出通道之间的连接数量，如果为1则省略。

bias (bool): 是否包含偏置项，如果为False则省略。

padding\_mode (str): 如果使用填充，使用哪种填充方式，如果为'zeros'则省略。

### conv2d类

卷积函数具体实现类，继承\_convNd基类，通过\_\_init\_\_函数完成初始化，重写nn.module中的forward函数，和\_convNd基类中的def \_conv\_forward（）函数，并调用F.conv2d()进行卷积运算

#### \_\_init\_\_函数：

功能描述：这段函数是定义了一个二维卷积层(Conv2d)的子类，用于构建神经网络模型。函数首先检查 padding\_mode 属性是否为 'zeros'，如果不是，则对输入进行相应的填充。然后，它使用 PyTorch 的 F.conv2d 函数执行卷积操作，返回卷积结果张量。

参数介绍：

input: 输入张量，形状为 (batch\_size, in\_channels, height, width)。

weight: 卷积核张量，形状为 (out\_channels, in\_channels/groups, kernel\_size[0], kernel\_size[1])。

bias: 可选的偏置张量，形状为 (out\_channels,)。

padding\_mode: 填充模式，字符串类型，默认为 'zeros'。

stride: 步幅，一个二元组，用于指定在每个空间维度上的步幅，默认为 1。

padding: 填充，一个二元组，用于指定在每个空间维度上要添加的填充数，默认为 0。

dilation: 膨胀系数，一个二元组，用于指定卷积核中各元素之间的间隔，默认为 1。

groups: 分组卷积的组数，默认为 1。

bias: 是否使用偏置，布尔类型，默认为 True。

#### \_conv\_forward函数：

功能描述：这段函数是Conv2d 模块的前向传递实现，用于计算 2D 卷积操作。函数首先判断 padding\_mode 是否为 'zeros'，如果不是，则对输入张量进行 padding 操作，然后使用 F.conv2d 函数计算卷积结果；否则，直接使用 F.conv2d 函数计算卷积结果。其中，F.pad 函数用于进行 padding 操作，\_reversed\_padding\_repeated\_twice 是 Conv2d 模块内部计算 padding 所需的一个变量，不需要用户传入。

参数介绍：

input: 输入张量

weight: 卷积核张量

bias: 可选的偏置张量

# 长短期记忆网络

## 引言

本节设计了基于长短期记忆网络（Long Short-Term Memory Network，LSTM）的趋势预测算法模型。趋势预测在许多领域具有重要的应用，如金融市场分析、天气预测、销售预测等。传感器数据作为一种具有长时间范围依赖性的时间序列数据，其准确的趋势预测对于决策和规划具有关键意义。

然而，传统的趋势预测方法往往忽视了时间序列数据中的长期依赖性，导致预测结果不准确或不稳定。为了克服这个问题提高趋势预测精度，本节尝试构建LSTM神经网络模型。LSTM是一种适用于时间序列数据的特殊类型的循环神经网络，通过引入门控单元来解决梯度消失或梯度爆炸的问题，从而更好地捕捉长期依赖性。本节将详细介绍LSTM网络的原理和结构，并给出相应的算法设计与算法实现流程。

## 相关理论

### LSTM基本结构

长短时记忆神经网络LSTM(Long Short Term Memory，LSTM)是一种时间递归神经网络，适合于处理和预测时间序列事件，它是在循环神经网络（Recurrent Neural Network，RNN）的基础上演变而来的，它的基本结构单元如图10‑1所示：

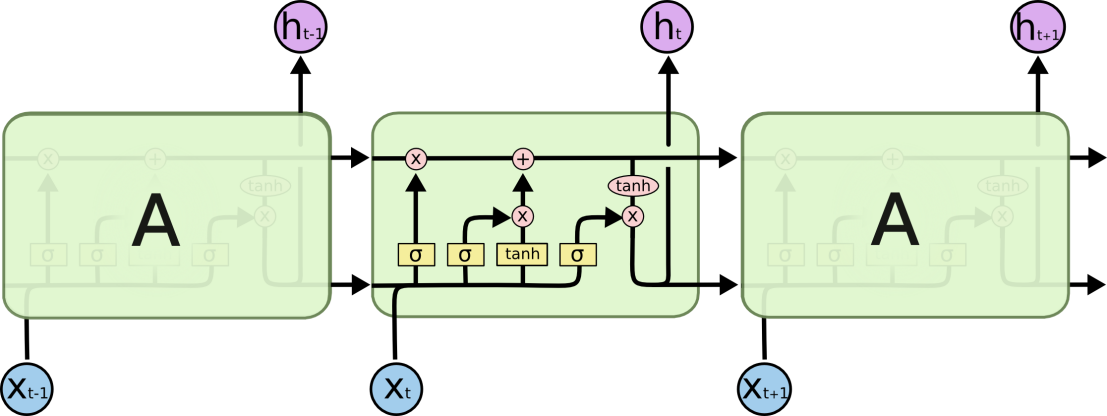


图10‑1 LSTM基本结构单元

LSTM主要由四部分组件和两种激活函数组成，分别是单元状态、遗忘门、输入门、输出门和sigmoid函数、tanh函数，如图10‑2所示。

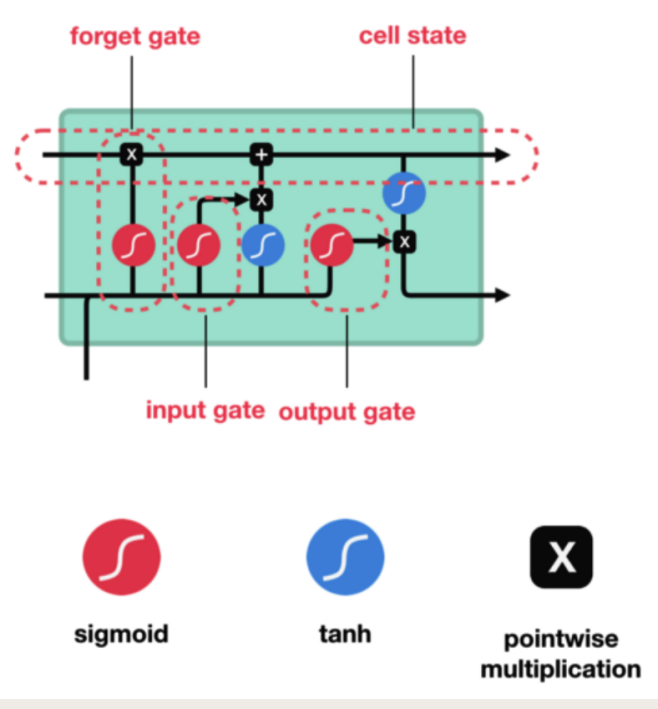


图10‑2 LSTM具体结构

其中tanh激活函数被用来帮助调节流经网络的数值，使其总是在-1和1之间，避免数据中较大值影响较小值的传播。而Sigmoid激活类似于tanh激活。用于隐层神经元输出，取值范围为(0,1)，它可以将一个实数映射到(0,1)的区间，可以用来做二分类，因此该值保持不变或被保留。

单元状态就像一条运输公路，将相对的信息一直传递到序列链中，且在整个序列的处理过程中携带相关信息。因此，即使是较早的时间步骤的信息也可以进入以后的时间步骤，减少短期记忆的影响。在单元状态的旅程中，信息通过门被添加或删除到单元状态中。

其中的遗忘门，决定哪些信息应该被丢弃或保留。具体表现为，将前一个隐藏状态的信息和来自当前输入的信息一起传递到sigmoid函数中，输出数值在0和1之间。越接近0意味着遗忘，越接近1意味着保留，如图10‑3所示。

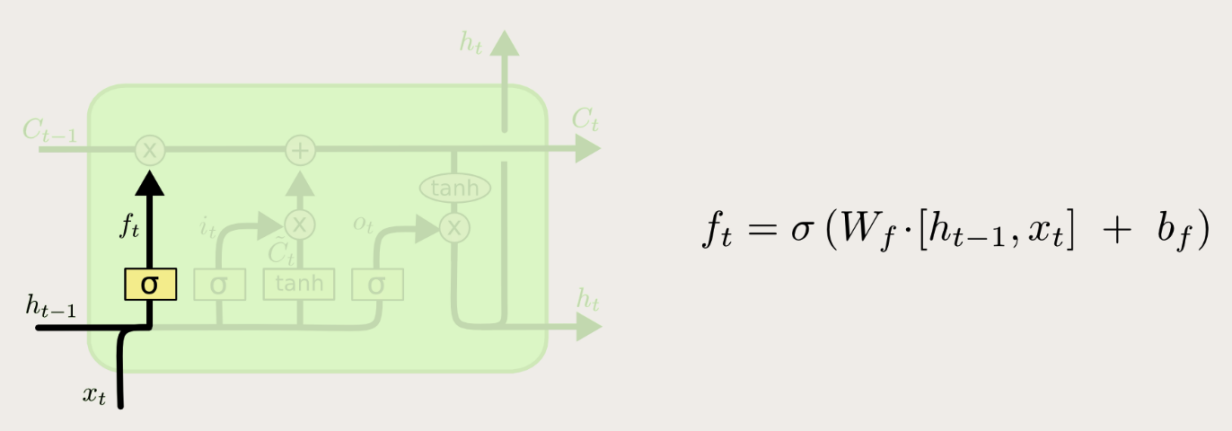


图10‑3 遗忘门结构图

输入门将之前的隐藏状态和当前的输入传给一个sigmoid函数，输出的值为0和1之间。此外，将隐藏状态和当前输入一起传入tanh函数，以转化为-1和1之间的值，帮助调节网络，最后然后将tanh输出与sigmoid输出相乘。sigmoid输出决定哪些信息是重要的，可以从tanh输出中保留，如图10‑4所示。

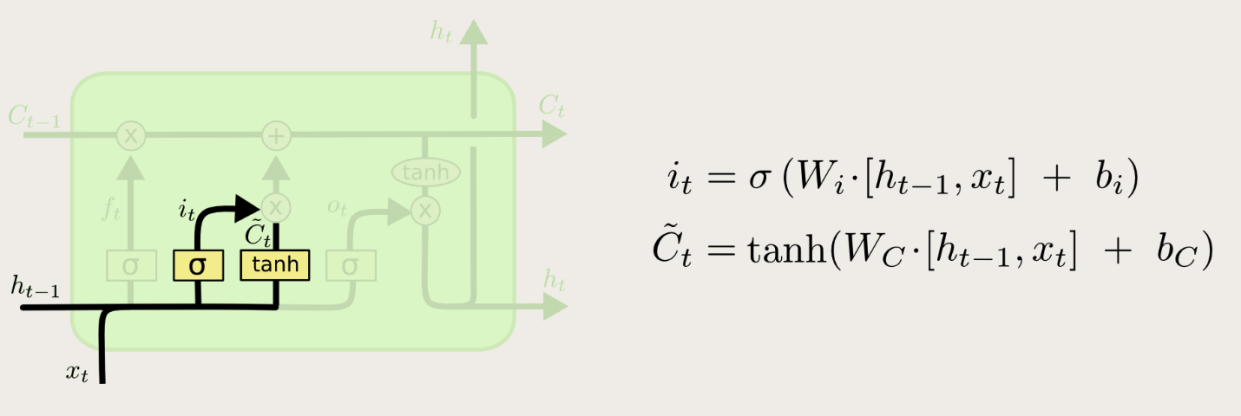


图10‑4 输入门结构图

输出门决定下一个隐藏状态应该是什么。首先，我们将之前的隐藏状态和当前的输入传给一个sigmoid函数。然后我们把新修改的单元格状态传给tanh函数。我们将tanh输出与sigmoid输出相乘，以决定隐藏状态应该携带什么信息，输出就是隐藏状态。最后将新的单元格状态和新的隐藏状态带到下一个时间步骤中，如图10‑5所示。

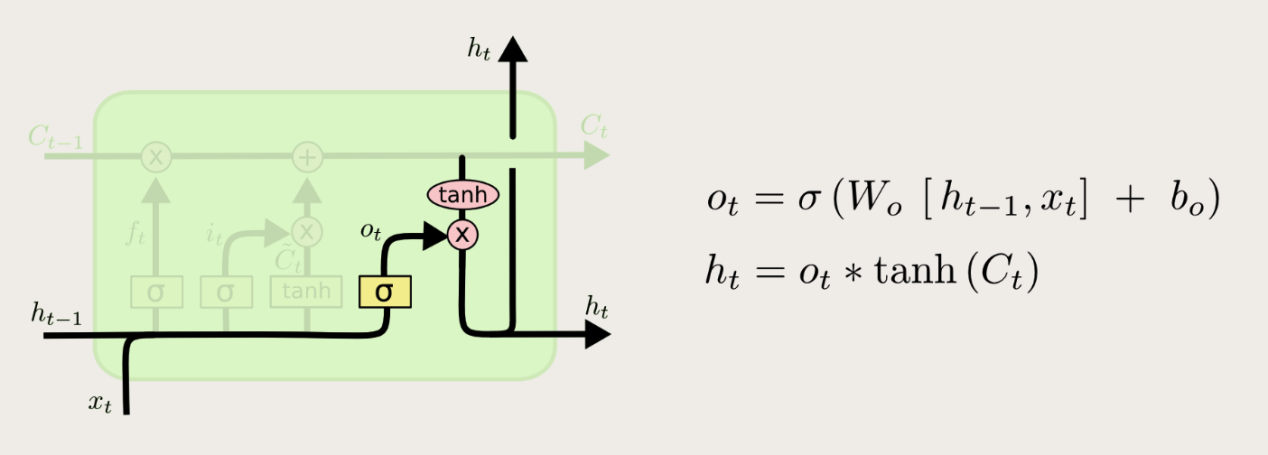


图10‑5 输出门结构图

## 算法设计

传感器数据是具有长时间范围依赖性的时间序列数据。当前趋势预测研究中，大多没有考虑长期依赖性。而RNN是一种适用于时间序列的神经网络，其输出可以是连续值，理论上适用于时间序列相关的趋势预测任务。然而，由于梯度消失或梯度爆炸问题，RNN在处理长期时间依赖的数据时并不可行。长短期记忆网络（Long Short-Term Memory Network，LSTM）是一种改进的RNN网络，通过一组门控单元避免了梯度消失或梯度爆炸的问题。具体LSTM趋势预测网络结构如图10‑6所示。

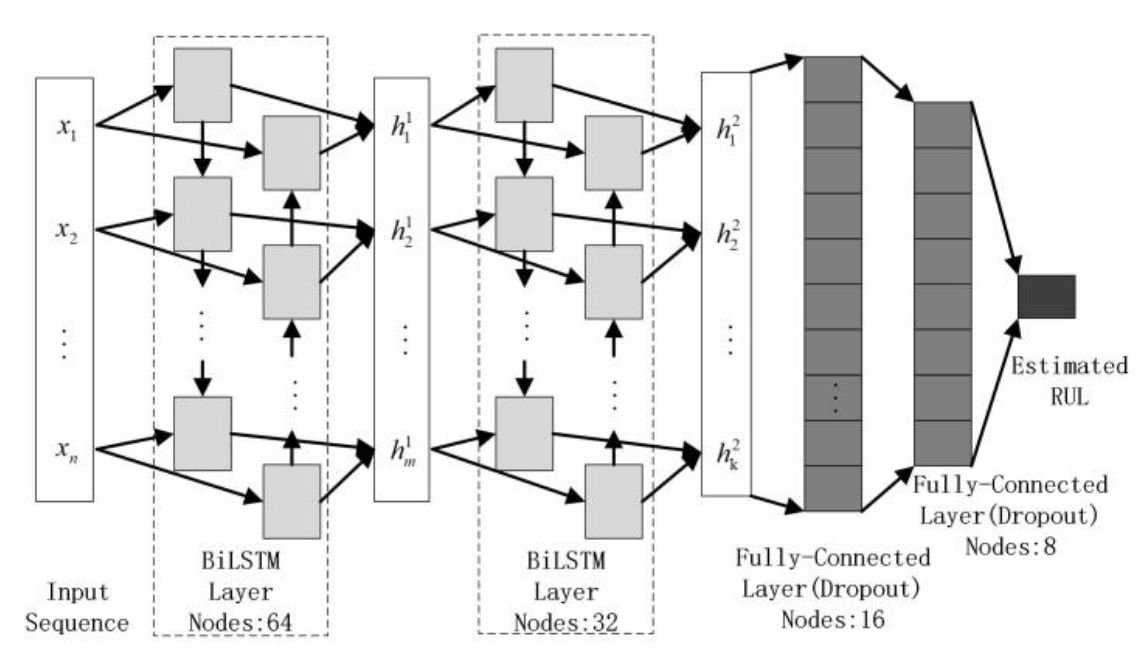


图10‑6 LSTM趋势预测模型结构

## 算法实现

### nn.Module

所有的网络层都是继承于这个类。

nn.module中有八个重要的属性用于管理整个类，主要关注其中的parameters和modules两个属性。

parameters：管理存储属于nn.Parameter类的属性，例如权值或者偏置参数；

modules：用来存储管理nn.Module类，例如在LeNet中会构建子模块，modules就会存储创建的卷积层等；

buffers：存储管理缓冲的属性，如训练过程中BN的均值，或者是方差都会存储在buffers

\*\_hooks：存储管理钩子函数

nn.Module的属性构建会在module类中进行属性赋值的时候会被setattr()函数拦截，在这个函数当中会判断即将要赋值的数据类型是否是nn.parameters类，如果是的话就会存储到parameters字典中；如果是module类就会存储到module字典中

### RNNBase基类

#### \_\_init\_\_函数

功能描述：这段代码定义了一个RNNBase类的初始化函数\_\_init\_\_()，该类是用于实现循环神经网络（RNN）的基础类。

参数介绍：

mode: str：RNN的模式，可以是"LSTM"、"GRU"、"RNN\_TANH"或"RNN\_RELU"。

input\_size: int：输入特征的维度。

hidden\_size: int：隐藏层的维度。

num\_layers: int = 1：RNN的层数，默认为1。

bias: bool = True：是否使用偏置项，默认为True。

batch\_first: bool = False：输入数据的维度中是否批次维度在第一维度，默认为False。

dropout: float = 0.：dropout的比例，默认为0，表示不使用dropout。

bidirectional: bool = False：是否使用双向RNN，默认为False。

proj\_size: int = 0：投影层的维度，默认为0，表示不使用投影层。

device=None, dtype=None：设备和数据类型的参数，默认为None。

参数验证：

针对dropout参数，检查其值是否在[0, 1]的范围内。

如果dropout大于0且num\_layers等于1，则发出警告，因为dropout只对除最后一层以外的层起作用。

针对proj\_size参数，检查其值是否为正整数或零，负数会引发ValueError。

针对proj\_size参数，检查其值是否小于hidden\_size，否则会引发ValueError。

#### \_\_setattr\_\_函数

功能描述：该函数重写了属性设置的行为，通过调用父类的\_\_setattr\_\_()函数，将属性设置为新的值。

参数介绍：

self：指向当前实例的引用。

attr：属性的名称。

value：属性的值。

#### flatten\_parameters函数

功能描述：该函数定义了一个名为flatten\_parameters()的函数，用于在启用cuDNN并且模块在GPU上运行时重置参数数据指针，以便使用更快的代码路径。

参数描述：

self：指向当前实例的引用。

#### \_apply函数

功能描述：调用父类的\_apply()方法，在CUDA设备上对参数进行扁平化操作，以便在后续计算中可以使用更高效的代码路径，并将返回值存储在ret变量中。

参数介绍：

self：指向当前实例的引用。

fn：要应用于模块的函数。

#### reset\_parameters函数

功能描述：重置模块的参数值，对于每个参数张量，使用均匀分布初始化其值。

参数介绍：

self：指向当前实例的引用。

实现细节：首先，计算标准差stdv，其值为1.0除以self.hidden\_size的平方根。

对于模块的每个参数张量，使用均匀分布初始化其值，范围为从-stdv到stdv。

这里使用了init.uniform\_()函数来实现参数的均匀分布初始化。

#### check\_input函数

功能描述：该函数用于检查输入张量的维度和大小是否符合预期。如果输入张量的维度不正确或输入大小与预期的输入大小不匹配，则引发RuntimeError异常。

参数介绍：

self：指向当前实例的引用。

input：输入张量。

batch\_sizes：可选的批次大小张量。

实现细节：首先根据是否存在batch\_sizes确定预期的输入维度。如果存在batch\_sizes，预期的维度为2；否则，预期的维度为3。检查输入张量的维度是否与预期维度相匹配，如果不匹配，则引发RuntimeError异常，指示输入必须具有正确的维度。检查输入张量的最后一个维度（-1维）是否与期望的输入大小（self.input\_size）相等，如果不相等，则引发RuntimeError异常，指示输入的最后一个维度大小必须与预期的输入大小相同。

#### get\_expected\_hidden\_size函数

功能描述：根据输入张量和批次大小张量（如果有），计算预期的隐藏状态大小。返回一个包含三个整数的元组，分别表示层数、批次大小和隐藏状态维度。

参数介绍：

self：指向当前实例的引用。

input：输入张量。

batch\_sizes：可选的批次大小张量。

实现细节：如果存在batch\_sizes，则将第一个元素转换为整数，并赋值给mini\_batch变量；否则，根据batch\_first属性判断采用input的第一维度还是第二维度作为批次大小，并赋值给mini\_batch变量。根据是否有投影层，确定预期的隐藏状态大小：如果有投影层（self.proj\_size > 0），则预期的隐藏状态大小为 (self.num\_layers \* num\_directions, mini\_batch, self.proj\_size)。如果没有投影层，预期的隐藏状态大小为 (self.num\_layers \* num\_directions, mini\_batch, self.hidden\_size)。返回包含预期隐藏状态大小的元组。

#### check\_hidden\_size函数

功能描述：检查隐藏状态的大小是否与预期大小相匹配。如果隐藏状态的大小与预期大小不匹配，则引发RuntimeError异常。

参数介绍：

self：指向当前实例的引用。

hx：隐藏状态张量。

expected\_hidden\_size：预期的隐藏状态大小，表示为包含三个整数的元组。

msg：异常消息的格式字符串，用于指示预期大小和实际大小。默认值为 'Expected hidden size {}, got {}'。

实现细节：检查隐藏状态张量的大小是否与预期大小相匹配，如果不匹配，则引发RuntimeError异常，异常消息包含预期大小和实际大小。

#### check\_forward\_args函数

功能描述：检查前向传播的输入参数是否符合要求、检查输入张量的维度和大小是否正确、检查隐藏状态张量的大小是否与预期大小相匹配。

参数介绍：

self：指向当前实例的引用。

input：输入张量。

hidden：隐藏状态张量。

batch\_sizes：可选的批次大小张量。

实现细节：调用check\_input方法，检查输入张量的维度和大小是否正确。调用get\_expected\_hidden\_size方法，计算预期的隐藏状态大小。调用check\_hidden\_size方法，检查隐藏状态张量的大小是否与预期大小相匹配。

#### permute\_hidden函数

功能描述：对隐藏状态进行排列。如果排列索引未提供，则返回原始隐藏状态。

参数介绍：

self：指向当前实例的引用。

hx：隐藏状态张量。

permutation：排列索引张量，用于指定排列顺序。可选参数。

输出介绍：

返回排列后的隐藏状态张量。

实现细节：如果排列索引未提供（permutation is None），则直接返回原始隐藏状态（hx）。

否则，调用apply\_permutation方法，将排列索引应用到隐藏状态张量上，返回排列后的结果。

#### forward函数

功能描述：执行RNN模型的前向传播操作。接收输入数据和初始隐藏状态作为输入。返回前向传播的结果，包括输出和最终隐藏状态。

参数介绍：

self：指向当前实例的引用。

input：输入数据张量或打包序列（PackedSequence）对象。

hx：初始隐藏状态张量。可选参数。

实现细节：首先检查输入数据是否为打包序列对象，并提取相关信息（如批次大小、排序索引等）。如果初始隐藏状态未提供，则根据输入数据的形状创建一个零填充的隐藏状态张量。

调用permute\_hidden方法，根据排序索引对初始隐藏状态进行排列。进行前向传播操作，根据RNN模型的不同模式（如LSTM、GRU等）选择相应的实现方法（\_impl）。将前向传播的结果（输出和最终隐藏状态）保存在result中。根据是否为打包序列对象，将输出进行适当的封装。返回封装后的输出和经过逆排序的最终隐藏状态。

### LSTM类

长短期神经网络具体实现类，继承RNNBase基类，通过\_\_init\_\_函数完成初始化，并通过RNNBase基类中的extra\_repr函数显示初始化参数。

#### \_\_init\_\_函数

功能描述：初始化LSTM类的实例。调用父类（RNNBase）的\_\_init\_\_方法进行初始化。

参数介绍：

self：指向当前实例的引用。

\*args：接收任意数量的位置参数。

\*\*kwargs：接收任意数量的关键字参数。

实现细节：调用父类的\_\_init\_\_方法，将字符串'LSTM'和传入的位置参数和关键字参数传递给父类的\_\_init\_\_方法。

#### get\_expected\_cell\_size函数

功能描述：根据输入和批次大小计算LSTM单元的期望大小。返回一个包含三个整数的元组，表示期望的隐藏状态大小。

参数介绍：

self：指向当前实例的引用。

input：输入张量，形状为[batch\_size, seq\_length, input\_size]或[seq\_length, batch\_size, input\_size]，取决于batch\_first参数。

batch\_sizes：可选的批次大小张量，形状为[batch\_size]。如果提供了批次大小，则可以根据它计算期望大小。

#### check\_forward\_args函数

功能描述：检查LSTM前向传播的参数是否符合要求。检查输入、隐藏状态的大小是否正确。

参数介绍：

self：指向当前实例的引用。

input：输入张量，形状为[seq\_length, batch\_size, input\_size]或[batch\_size, seq\_length, input\_size]，取决于batch\_first参数。

hidden：包含两个张量的元组，表示隐藏状态和细胞状态。形状为(num\_layers \* num\_directions, batch\_size, hidden\_size)。

batch\_sizes：可选的批次大小张量，形状为[batch\_size]。如果提供了批次大小，则可以根据它检查隐藏状态的大小。

#### permute\_hidden函数

功能描述：对LSTM的隐藏状态进行置换，根据给定的置换索引。返回置换后的隐藏状态。

参数介绍：

self：指向当前实例的引用。

hx：包含两个张量的元组，表示隐藏状态和细胞状态。形状为(num\_layers \* num\_directions, batch\_size, hidden\_size)。

permutation：可选的置换索引张量，用于指定新的顺序。

实现细节：首先，检查permutation是否为None，如果是，则直接返回原始的隐藏状态hx。

如果permutation不为None，则使用apply\_permutation函数对隐藏状态的两个张量分别进行置换，根据给定的permutation索引。最后，返回置换后的隐藏状态的元组。

#### forward函数

功能描述：对输入序列进行LSTM的前向计算。返回计算结果，包括输出序列和更新后的隐藏状态。

参数介绍：

self：指向当前实例的引用。

input：输入序列，可以是张量或PackedSequence对象。

hx：可选的初始隐藏状态，默认为None。

实现细节：首先，保存原始输入序列input的引用为orig\_input。根据输入序列的类型进行处理：如果input是PackedSequence对象，则提取出其中的数据和相关信息，如batch\_sizes、sorted\_indices和unsorted\_indices，并确定最大批次大小。如果input是张量，则根据batch\_first属性确定最大批次大小，并设置相关信息为None。如果初始隐藏状态hx为None：

根据是否双向LSTM确定隐藏状态的方向数（1或2）。计算真实的隐藏状态大小，考虑是否有投影层。创建全零张量作为初始隐藏状态，包括隐藏状态h和细胞状态c。将h和c组成元组作为初始隐藏状态hx。否则，根据输入序列的顺序索引对隐藏状态进行置换。调用check\_forward\_args函数检查输入的合法性。根据是否有batch\_sizes参数，调用底层的LSTM前向计算函数进行计算。提取计算结果中的输出序列和隐藏状态。如果输入是PackedSequence对象：将输出序列封装为PackedSequence对象，使用原始的batch\_sizessorted\_indices和unsorted\_indices。返回输出序列和置换后的隐藏状态。否则，返回输出序列和置换后的隐藏状态。

# Transformer

## 引言

Transformer是一种革命性的神经网络架构，最初由Vaswani等人在2017年提出，用于解决自然语言处理领域的序列建模问题。传统的序列模型，如循环神经网络（RNN）和卷积神经网络（CNN），在处理长期依赖关系和并行计算方面存在一定的限制。为了克服这些限制，Transformer引入了自注意力机制和位置编码，彻底改变了序列建模的方式。

Transformer的核心思想是将输入序列的不同位置之间的关系进行建模，而不依赖于顺序处理。它由编码器和解码器两部分组成。编码器负责将输入序列映射为一系列高维向量表示，而解码器则根据编码器的输出生成相应的输出序列。不同于传统的逐步处理方法，Transformer通过自注意力机制，能够在一次计算中同时考虑输入序列中的所有位置，从而捕捉到更长距离的依赖关系。

在自然语言处理任务中，Transformer取得了显著的突破。它在机器翻译、文本摘要、语言生成等任务上取得了优秀的效果，并且成为了当前最先进的序列建模方法之一。Transformer的成功得益于其能够并行计算的特性和强大的建模能力，同时还具备较好的泛化能力和可解释性。

在本节中，我们将引入Transformer架构来解决一种特定的问题，即趋势预测问题。通过借鉴Transformer的注意力机制和位置编码，我们希望能够更好地捕捉时间序列数据中的长期依赖关系，并提高对输入数据的建模能力。

Transformer和LSTM算法的区别在于，LSTM的训练过程是迭代的，在进行自然语言处理训练的过程中，需要将上一个序列的信息通过LSTM单元提取出隐藏状态，才能辅助进行下一个序列的信息的训练。而Transformer的训练是通过位置编码和自注意力机制，将所有信息一起并行训练，具有更高的效率，Transformer结构如图11‑1所示。

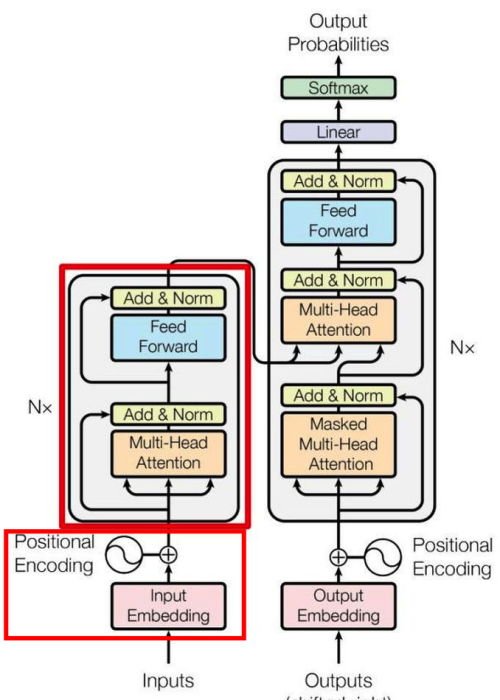


图11‑1 基本结构图

## 相关理论

### 自注意力机制

自注意力机制旨在估计给定序列中一个值与其他值的相关性，对序列中所有值之间的交互关系建模。自注意力机制的表示如下，设一个序列为，其中是输入序列的长度，表示向量的嵌入维度。输入向量先通过3个独立的权重矩阵映射成如式(3-1)所示的查询向量，键向量，以及值向量。这3个矩阵都是可学习的，其映射过程如下式所示：



其中三个可学习的权重矩阵分别为：，和*。*

然后计算查询与键的点积，并且将计算出的点积通过softmax算子归一化为注意力值。最后将每个值向量乘以注意力值的总和，如图11‑1（a）所示。



图11‑2 点积注意力与多有注意力机制

通过这种方式，具有较大注意力值的向量在后续的层中可以获得额外的关注。自注意力的计算定义公式如下：



通过自注意力的计算可以看出，序列中的值所处的位置并不会影响该值与其他值的相关性，也即自注意力层的输出与输入顺序是无关的。为了解决输入向量位置信息缺失的问题，需要对输入向量进行额外的位置编码。为了增强输入的表示，可以采用多个独立并行的权重矩阵将输入映射到多种表示中执行点积，最后将输出聚合在一起结果，如图11‑1（b）所示，也就是多头注意力机制。多头注意使得模型能够联合注意不同位置上不同表示子空间的信息。多头也即是指将自注意力的过程并行且独立地进行次，为了减少计算量，其中每一次的输出的隐藏维度为，然后将次的输出聚合。总体的计算开销与用一个单独的不减少维度的自注意力机制相同，具体如下所示：





其中，，以及都是用于映射的权重矩阵，表示并行的自注意力层数。通常。

## 算法设计

Transformer是一个序列到序列的架构，包括一个编码器和一个解码器。编码器接收输入序列并将其映射到一个高维向量，然后将该向量输入解码器生成输出序列。与RNN的顺序性不同，Transformer通过点积注意力机制来建立长期依赖关系。在本节中，我们提出了一种基于Transformer编码器的架构，用于解决基于RNN和CNN方法的局限性，网络结构如图11‑3所示：

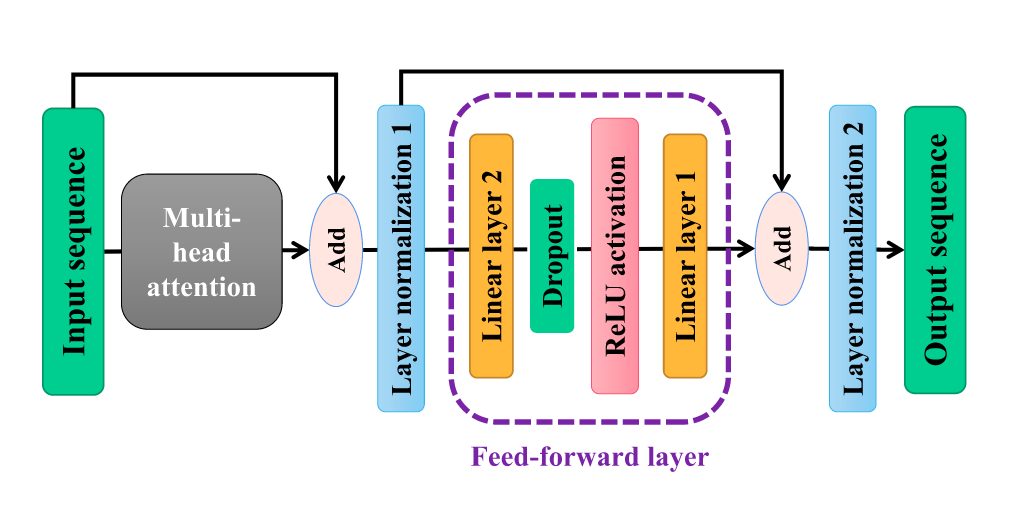


图11‑3 Transformer趋势预测模型结构

### Linear类

#### \_\_init\_\_函数

功能描述：定义了一个神经网络中的线性层（linear layer）的初始化函数。它是用于构建一个具有可学习权重和可选偏置项的线性层。

参数介绍：

in\_features（int）：输入特征的数量。

out\_features（int）：输出特征的数量。

bias（bool）：是否使用偏置项。默认为True，即使用偏置项。

device：可选参数，用于指定在哪个设备上创建张量。默认为None，表示使用当前设备。

dtype：可选参数，用于指定张量的数据类型。默认为None，表示使用默认数据类型。

#### reset\_parameters函数

功能描述：线性层（linear layer）中的参数重置函数reset\_parameters()。它用于初始化权重和偏置项的值。

参数介绍：

self.weight：模型权重。

#### forward函数

功能描述：前向传播函数，它使用线性变换将输入张量 input 映射到输出张量。

参数介绍：

input: 输入张量，形状为 (batch\_size, input\_features)。

self.weight: 权重张量，形状为 (output\_features, input\_features)。它用于将输入特征线性映射到输出特征。

self.bias: 偏置张量，形状为 (output\_features)。它用于添加偏置项到线性变换的结果。

#### extra\_repr函数

功能描述：用于返回额外的字符串表示的方法。它通常被用于定制类的字符串表示形式，以提供更多的信息

参数介绍：

self.in\_features: 输入特征的数量。

self.out\_features: 输出特征的数量。

self.bias: 是否存在偏置项。

实现细节：函数通过返回一个格式化的字符串来提供类的额外表示。字符串的格式为 'in\_features={}, out\_features={}, bias={}'，其中 {} 部分会被具体的参数值替换。如果存在偏置项（self.bias 不为 None），则将 'bias=True' 包含在返回的字符串中，否则为 'bias=False'。

### Transformer\_RUL类

#### PositionalEncoder函数

功能描述：这段函数实现了位置编码（Positional Encoding）。位置编码是在Transformer模型中用于为输入序列的每个位置添加位置信息的一种技术。

参数介绍：

maximum\_position\_encoding: 最大位置编码的长度，表示输入序列的最大长度。

d\_model:模型的维度，表示隐藏单元的数量或特征的维度。

实现细节：get\_angles 函数根据位置、索引和模型维度计算角度。它使用了一个角度因子，其中 i为偶数时应用正弦函数，i为奇数时应用余弦函数。使用 np.arange 创建位置编码的索引矩阵，并使用 get\_angles 计算角度。通过切片操作，将正弦函数应用于偶数索引的角度，将余弦函数应用于奇数索引的角度。将计算得到的角度数组添加一个额外的维度，并返回作为位置编码的张量。最后，函数将位置编码的张量转换为 torch.tensor 格式，并指定数据类型为 torch.float32，然后返回该张量。

#### MultiHeadAttention函数

功能描述：这段代码定义了一个多头注意力（Multi-Head Attention）的类 MultiHeadAttention。多头注意力是Transformer模型中的一个重要组成部分，用于对输入序列进行注意力加权的操作。

参数介绍：

heads: 多头注意力的头数。

d\_model: 模型的维度（或隐藏单元的数量）。

dropout: Dropout的概率，用于在注意力计算中进行随机丢弃

实现细节：在 \_\_init\_\_ 方法中，定义了多头注意力的各个线性层（q\_linear、k\_linear、v\_linear 和 out）和注意力机制（attention）。在 forward 方法中，进行了多头注意力的计算过程：

首先，将输入 q、k 和 v 分别通过线性层进行变换，然后按头数进行切分，得到对应的维度。对 k、q 和 v 进行转置操作，使得维度变为 (batch\_size, num\_heads, seq\_len, d\_k)。

调用注意力机制（attention）计算注意力权重 scores。将 scores 进行转置和重塑操作，得到形状为 (batch\_size, seq\_len, d\_model) 的张量 concat。将 concat 通过最后一个线性层 out 进行线性变换，得到最终的输出output。

# 图神经网络

## 引言

多变量时间序列的可靠和有效表示对于精确预测未来时间序列至关重要。在多变量时间序列预测中，每个变量的取值不仅取决于其自身的历史值，还受到其他变量的影响，因此变量之间存在着相互依赖的关系。传统的机器学习和深度学习方法往往难以直接地建模和表示多变量时间序列之间的复杂依赖关系。

为了克服这个问题，可采用图结构来构建原始输入数据，利用图神经网络的优势来有效地捕捉和表示多变量时间序列之间的关联性和依赖关系。通过将时间序列数据表示为图，将每个时间点或时间段视为图中的一个节点，并利用边来表示节点之间的关系。这种表示方式允许我们以图的形式呈现和分析时间序列数据，从而更好地捕捉非线性的依赖关系和复杂的交互作用，从而更好地表示多变量时间序列中的隐含模式和规律，提高模型的预测能力。

## 相关理论

1、将输入信息转化为图结构：节点特征、边特征，临接矩阵

首先通过SMD对原时间序列进行分解，再根据SMD提取出的特征信息进行构图，每个变量作为一个节点，节点的特征是该变量的时间序列。每两个节点之间都有边特征，边特征是根据两个序列各个IMF分解结果分别计算的相似度，公式如下，计算第i个和第k个节点中第k个IMF的相似度：



在邻接矩阵的构造上，文中计算两两节点序列趋势项对应序列的相似度，当相似度大于一定阈值时，这两个序列建立边，通过这种方式将具有相似趋势的变量连接起来。

2、基于Transformer架构引入图结构，修改后的自注意力机制计算公式如下：



上式中第二项第三项分别引入了每个IMF对应边特征的信息、邻接矩阵的信息，从而更好地表示多变量时间序列中的隐含模式和规律，提高模型的预测能力。网络架构图如图12‑1所示

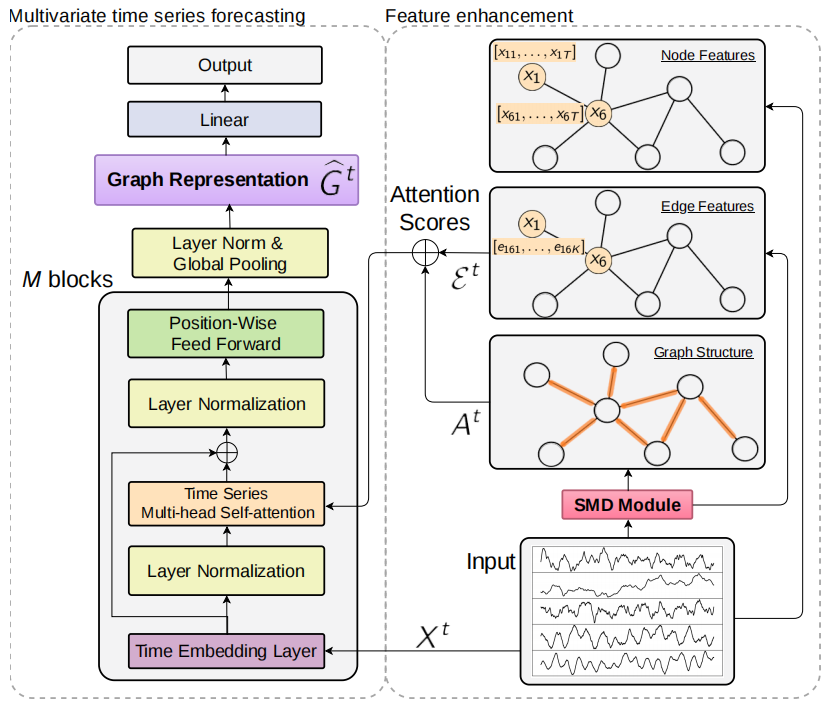


图12‑1 图神经网络趋势预测结构

## 实现细节

### KOI\_model\_train\_test\_interface类

#### \_\_init\_\_函数

功能描述:该代码定义了一个名为KOI\_model\_train\_test\_interface的类，该类用于训练和测试模型。它接受一个TSAT\_model作为输入，并根据提供的模型参数和训练参数进行模型训练和评估。

参数介绍:

TSAT\_model: 要训练和测试的模型。

model\_params: 模型的参数配置，以字典形式存储。

train\_params: 训练的参数配置，以字典形式存储。

criterion: 损失函数，根据训练参数指定的损失函数类型进行实例化。

optimizer: 优化器，这一行代码被注释掉了，未使用该优化器。

#### process\_data函数

功能描述:该代码定义了一个名为process\_data的函数，用于生成图数据。它接受一个DataFrame对象和一些参数，根据给定的参数生成图数据。

参数介绍：

df: 输入的DataFrame对象，包含时间序列数据。

n\_lookback\_days: 向前观察的天数。

n\_lookforward\_days: 向后观察的天数。

adj\_mat\_method: 邻接矩阵生成方法，默认为'fully\_connected'。可选值为'fully\_connected'、'correlation'、'zero\_mat'和'random'。

use\_tqdm: 是否使用tqdm库显示进度信息，默认为True。

kwargs： 其他可选参数。

### Time\_Embeddings类

#### \_\_init\_\_函数

功能描述：

该代码定义了一个名为Time\_Embeddings的类，继承自nn.Module类。它用于对时间序列数据进行时间嵌入（Time Embeddings）的操作。

实现步骤:

类的构造函数\_\_init\_\_接受三个参数：d\_model表示嵌入后的维度，l\_backcast表示输入序列的长度，dropout表示嵌入后的元素进行dropout的概率。

在构造函数中，调用父类的构造函数super(Time\_Embeddings, self).\_\_init\_\_()进行初始化。

创建一个线性层linear，输入维度为l\_backcast，输出维度为d\_model。该线性层将输入的时间序列进行线性变换，用于生成时间嵌入。

创建一个dropout层dropout，并指定dropout的概率。

该类没有定义forward方法，因此需要在具体使用时通过调用该类的对象来执行嵌入操作。

参数介绍:

d\_model: 嵌入后的维度，即输出维度。

l\_backcast: 输入序列的长度，即输入维度。

dropout: 嵌入后的元素进行dropout的概率。

#### forward函数

功能描述:

该代码定义了Time\_Embeddings类的前向传播方法forward，用于对输入进行时间嵌入操作。

参数介绍:

x：输入的时间序列数据，形状为(batch\_size, l\_backcast)，其中batch\_size表示批量大小，l\_backcast表示输入序列的长度。

### Time\_Series\_Self\_Attention类

#### \_\_init\_\_函数

功能描述:该代码定义了Time\_Series\_Self\_Attention类的初始化方法，用于创建一个时间序列的自注意力模型。

参数介绍:

layer: 自注意力层的实例对象，用于进行时间序列的自注意力计算。

N: 自注意力层的重复次数，表示在模型中重复使用多少个自注意力层。

scale\_norm: 是否使用ScaleNorm进行归一化操作的标志，如果为True，则使用ScaleNorm进行归一化；如果为False，则使用LayerNorm进行归一化。

#### forward函数

功能描述:该代码定义了Time\_Series\_Self\_Attention类的前向传播方法（forward），用于将输入数据通过每个自注意力层，并进行归一化操作后返回结果。

计算得到的x传递给self.Norm进行归一化操作，并将归一化后的结果返回。

参数介绍:

x: 输入数据，即时间序列数据。

mask: 掩码数据，用于指示输入数据中的有效数据位置。

adj\_matrix: 邻接矩阵，表示时间序列数据的节点之间的关联关系。

imf\_1\_matrix至imf\_5\_matrix: IMF特征矩阵，表示时间序列数据的不同IMF分量的特征。

## Graph\_Representation\_Readout类

#### \_\_init\_\_函数

功能描述:

该代码定义了Graph\_Representation\_Readout类的初始化方法（init），用于构建图表示读出模型的网络结构。

参数介绍:

n\_nodes (int): 节点数量，表示图中的节点数目。

d\_model (int): 输入特征的维度。

aggregation\_type (str): 聚合类型，用于指定节点特征的聚合方式，默认为'mean'。

n\_output (int): 输出维度，默认为1。

n\_layers (int): 神经网络层数，默认为1。

leaky\_relu\_slope (float): LeakyReLU激活函数的负斜率，默认为0.01。

dropout (float): Dropout的概率值，默认为0.0。

scale\_norm (bool): 是否使用ScaleNorm进行归一化，默认为False。

#### \_mean\_aggregation函数

功能描述:该代码定义了\_mean\_aggregation方法，用于进行均值聚合操作。

参数介绍:

output\_masked (Tensor): 经过掩码处理的输出张量，表示需要进行聚合的特征。

mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。

dim (int): 聚合操作的维度，默认为1。

#### \_splitted\_mean\_aggregation函数

功能描述:该代码定义了 \_splitted\_mean\_aggregation 方法，用于进行分割均值聚合操作。

参数介绍:

output\_masked (Tensor): 经过掩码处理的输出张量，表示需要进行聚合的特征。

mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。

dim (int): 聚合操作的维度，默认为 1。

#### \_sum\_aggregation函数

功能描述：该代码定义了 \_sum\_aggregation 方法，用于进行求和聚合操作。\_sum\_aggregation 方法接受多个参数，包括:output\_masked (Tensor): 经过掩码处理的输出张量，表示需要进行聚合的特征。mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。dim (int): 聚合操作的维度，默认为 1。使用断言 (assert) 验证 dim 参数的类型和值，确保 dim 为整数且不为 0。对 output\_masked 进行求和操作，沿着指定的维度 (dim) 进行求和，得到 out\_sum。返回求和结果 out\_sum。

参数介绍:

output\_masked (Tensor): 经过掩码处理的输出张量，表示需要进行聚合的特征。

mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。

dim (int): 聚合操作的维度，默认为 1。

#### \_dummy\_node\_aggregation函数

功能描述：

该代码定义了 \_dummy\_node\_aggregation 方法，用于进行虚拟节点聚合操作。\_dummy\_node\_aggregation 方法接受多个参数，返回 output\_masked 中的第一列，即 output\_masked[:, 0]。

参数介绍：

output\_masked (Tensor): 经过掩码处理的输出张量，表示需要进行聚合的特征。

mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。

dim (int): 聚合操作的维度，默认为 1。

#### \_aggregation函数

功能描述：该代码定义了 \_aggregation 方法，用于根据指定的聚合类型对输出进行聚合操作。\_aggregation 方法接受多个参数，返回聚合后的结果 output\_avg\_pooling。

参数介绍：

aggregation\_type (str): 聚合类型，表示要执行的聚合操作。

output\_masked (Tensor): 经过掩码处理的输出张量，表示需要进行聚合的特征。

mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。

#### forward函数

功能描述:该代码定义了 forward 方法，用于对输入进行前向传播，并返回经过聚合和投影处理后的结果。forward 方法接受两个参数:x (Tensor): 输入张量，表示待处理的特征。mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。将聚合后的特征 output\_avg\_pooling 输入到 self.projected\_layer 进行投影处理，得到经过投影处理后的结果 projected。返回投影处理后的结果 projected。

参数介绍:

x (Tensor): 输入张量，表示待处理的特征。

mask (Tensor): 掩码张量，用于指示哪些元素参与聚合操作。

### Time\_Series\_Attention\_Transformer类

#### \_\_init\_\_函数

功能描述:该代码定义了一个名为 Time\_Series\_Attention\_Transformer 的类，用于实现时间序列的注意力转换模型。它包含了时间嵌入、时间序列自注意力和图形表示读出器等组件。

参数介绍:

time\_embeddings (对象): 时间嵌入组件，用于将时间信息转换为嵌入向量。

time\_series\_self\_attention (对象): 时间序列自注意力组件，用于对时间序列进行自注意力计算。

graph\_representation\_readout (对象): 图形表示读出器组件，用于从图形表示中提取有意义的信息。

#### \_time\_embeddings函数

功能描述:该代码定义了一个名为 \_time\_embeddings 的方法，用于对多维时间序列进行时间嵌入操作。它调用了 time\_embeddings 组件来执行时间嵌入的转换。定义了一个名为 \_time\_embeddings 的方法。调用 self.time\_embeddings 方法，将 multi\_time\_series 传递给时间嵌入组件进行处理。返回时间嵌入后的结果。

参数介绍:

multi\_time\_series (张量): 多维时间序列数据，用于进行时间嵌入操作。

#### \_time\_series\_self\_attention函数

功能描述:该代码定义了一个名为 \_time\_series\_self\_attention 的方法，用于对多维时间序列数据应用时间序列自注意力机制。它调用了 time\_series\_self\_attention 组件，并通过调用 \_time\_embeddings 方法获取时间嵌入后的数据作为输入。将时间嵌入后的结果以及其他参数传递给 time\_series\_self\_attention 组件进行时间序列自注意力计算，返回时间序列自注意力的结果。

参数介绍:

multi\_time\_series (张量): 多维时间序列数据，用于进行时间嵌入和时间序列自注意力计算。

ts\_mask (张量): 时间序列数据的掩码，用于指示有效数据的位置。

adj\_matrix (张量): 时间序列数据之间的邻接矩阵。

imf\_1\_matrix (张量): IMF1 组分之间的邻接矩阵。

imf\_2\_matrix (张量): IMF2 组分之间的邻接矩阵。

imf\_3\_matrix (张量): IMF3 组分之间的邻接矩阵。

imf\_4\_matrix (张量): IMF4 组分之间的邻接矩阵。

imf\_5\_matrix (张量): IMF5 组分之间的邻接矩阵。

#### \_graph\_representation\_readout函数

功能描述:该代码定义了一个名为 \_graph\_representation\_readout 的方法，用于对图表示进行读出操作。它调用了 graph\_representation\_readout 组件，并将 out 和 out\_mask 作为输入。

参数介绍:

out (张量): 表示图节点的输出。

out\_mask (张量): 节点输出的掩码，用于指示有效节点的位置。

#### forward函数

功能描述:该代码定义了一个名为 forward 的方法，用于前向传播操作。它接收多时间序列数据、时间序列的掩码、邻接矩阵和多个 IMF 矩阵作为输入。在方法中，它首先通过调用 \_time\_series\_self\_attention 方法对时间序列进行自注意力计算，然后将计算结果传递给 \_graph\_representation\_readout 方法进行图表示的读出操作。最后，返回图表示读出的结果。

参数介绍:

multi\_time\_series (张量): 多时间序列数据作为输入。

ts\_mask (张量): 时间序列的掩码，用于指示有效时间步的位置。

adj\_matrix (张量): 邻接矩阵，表示图的连接关系。

imf\_1\_matrix、imf\_2\_matrix、imf\_3\_matrix、imf\_4\_matrix、imf\_5\_matrix (张量): 多个 IMF 矩阵作为输入。

# 其它工具类设计

## 环境要求

|  |  |
| --- | --- |
| 开发环境 | |
| 操作系统 | Windows10 |
| 开发工具 | Pycharm Anaconda |
| 开发语言 | Python 3.6 |
| CPU | Intel Core i5-9300H 2.4GHz |
| 显卡 | Nvidia GTX 1660ti |
| 硬盘空间 | 1T |
| 内存 | 16G |

## 模型性能评估

针对上述算法拟合的趋势预测模型，模型的性能评估是至关重要的一步。性能评估不仅可以帮助我们了解模型的好坏，还可以帮助我们选择最佳的模型、调整和优化模型的参数。选择正确的评价指标对于优化模型和理解模型性能具有重要影响。在本章节中，介绍了四种常用的评价指标：均方误差（Mean Squared Error，MSE）、均方根误差（Root Mean Squared Error，RMSE）、自定义得分函数（score）和决定系数（R-Square，R2）。

均方误差（MSE）是预测值与实际值之间差值的平方的平均值。MSE是最常用的评价指标之一，能直观地反映出模型预测的误差大小。然而，由于MSE采用了平方误差，对于较大的误差，其值会显得特别大。这意味着，如果模型对一部分样本的预测特别差，那么MSE会变得非常大。

均方根误差（RMSE）是MSE的平方根。相较于MSE，RMSE对误差的惩罚没有那么严重。因此，RMSE更能反映出模型预测的实际误差。

自定义得分函数（score）是针对机械设备剩余寿命预测领域常用的评价指标。score函数可以对超前和滞后的预测结果产生不同的惩罚，对滞后预测的惩罚高于超前预测。这是由于当模型预测的剩余寿命值高于实际剩余寿命值时，可能会导致机械设备出现严重损坏，从而造成无法挽回的损失。因此，该评价指标更符合实际应用需求。

决定系数（R2）是一种常用的统计指标，用于评估回归模型的拟合优度。它表示模型能够解释因变量方差的比例。R2的取值范围在0到1之间，其中0表示模型无法解释因变量的方差，1表示模型能够完全解释因变量的方差。通常情况下，R2的值越接近1，说明模型对观测数据的拟合程度越好；而值接近0则表示模型的拟合效果较差。

为了计算上述四种模型预测效果评价指标，设计了不同的函数来得到相应的结果。这些函数将根据实际需要，计算预测值与真实值之间的差异，并根据不同的指标公式计算相应的评价指标值。

### mse函数

功能描述：计算均方误差（MSE）。均方误差是预测值与目标值差的平方的平均值。

计算公式：，式中代表预测样本总数，代表第个样本的真实值，代表第个样本的预测值。

参数介绍：

pred (torch.Tensor): 预测值，应为PyTorch张量。

target (torch.Tensor): 目标值，应为PyTorch张量。

返回值：返回标量值，表示预测值与目标值之间的均方误差。

### rmse函数

功能描述：计算均方根误差（RMSE）。均方根误差是均方误差的平方根。

计算公式：





式中代表预测样本总数，代表第个样本的真实值，代表第个样本的预测值。

参数介绍：

pred (torch.Tensor): 预测值，应为PyTorch张量。

target (torch.Tensor): 目标值，应为PyTorch张量。

返回值：返回标量结果。

### score函数

功能描述：根据超前和滞后的预测结果产生不同的预测损失。

计算公式：，式中代表预测样本总数，代表第个样本的真实值，代表第个样本的预测值。

参数介绍：

pred (torch.Tensor): 预测值，应为PyTorch张量。

target (torch.Tensor): 目标值，应为PyTorch张量。

返回值：返回标量值，表示预测值与目标值之间的均方误差。

### R2函数

功能描述：计算决定系数。

计算公式：，式中代表预测样本总数，代表第个样本的真实值，代表第个样本的预测值。

参数介绍：

pred (torch.Tensor)：预测值，应为PyTorch张量。

target (torch.Tensor)：目标值，应为PyTorch张量。

返回值：返回标量结果。

## 模型训练和保存

在深度学习或者机器学习中，模型保存是非常重要的一步。训练模型的过程可能需要几个小时甚至几天，因此我们需要及时保存模型便于后续使用或分析。模型保存主要包括模型的参数以及模型的结构，这些信息可以帮助我们恢复模型的状态并在新的数据上应用模型。以下我们将介绍五种常见的模型保存格式。

### pth文件

PyTorch框架中用于保存模型的文件格式。能够保存模型的权重和参数，可用于模型的保存和加载。这种格式适用于PyTorch模型的存储和共享，可确保模型在不同的环境中保持一致性。

### onnx文件

ONNX是一种开放的模型表示格式，可跨多个深度学习框架进行模型交换。它提供了一个中间格式，使得将模型从一个框架导出到另一个框架变得容易。通过将模型保存为ONNX格式，可以在不同的深度学习框架之间共享和使用模型，以实现更好的互操作性。

### pt文件

PyTorch框架中用于保存模型的文件格式。它们保存了模型的权重和参数，可用于模型的保存和加载。这种格式适用于PyTorch模型的存储和共享，可确保模型在不同的环境中保持一致性。

### pkl文件

pkl文件是Python中常用的序列化文件格式。它可以用于保存任何Python对象，包括模型、数据集、配置等。通过使用Python的pickle模块，可以将对象以二进制形式保存到文件中，并在需要时加载回来。这种格式对于在Python环境中方便地保存和加载对象非常有用。

### h5文件

HDF5（Hierarchical Data Format）是一种数据存储格式，广泛用于科学数据和大型数据集的存储。.h5文件支持多种数据类型和层次结构，并且具有高效的存储能力，可以存储不同类型的图像和数码数据，并且可以在不同类型的机器上传输，同时还有统一处理这种文件格式的函数库。

### model\_saved函数

函数model\_saved的功能是将模型保存到文件中。它接受三个参数：

model\_name：模型名称，用于构建保存文件的文件名。

model：要保存的模型对象。

input\_data（可选）：输入数据，仅在保存ONNX类型模型时使用。

根据模型名称的不同，函数会选择不同的保存方式，如果model\_name是['ARIMA', 'ExponentialSmoothing', 'SVR', 'RandomForestRegressor']中的一种，则将模型以pkl类型保存为文件，否则，将模型以pth类型保存为文件。

### model\_loaded

函数model\_loaded的功能是从文件中加载模型。它接受两个参数：

file\_path：要加载的模型文件的路径。

model\_name：模型名称，用于确定加载方式。

根据模型名称的不同，函数会选择不同的加载方式：

如果model\_name是['ARIMA', 'ExponentialSmoothing','SVR', 'RandomForestRegressor']中的一种，则从pkl类型文件中加载模型。否则，从pth类型文件中加载模型，加载后的模型将作为函数的返回值。

## 模型可视化

模型可视化是算法开发过程中的重要步骤，它能够提供对模型结构和工作原理的直观理解，帮助我们更好地调试和优化模型。在本节中，我们将介绍一些常用的模型可视化方法，以便更深入地了解模型的内部运行情况。

### plot\_tree函数

plot\_tree函数用于可视化随机森林模型。提供一种直观的方式来展示随机森林中某一颗决策树模型的结构和决策规则，帮助我们理解模型的决策过程。

参数说明：

decision\_tree：决策树模型对象，即已经训练好的DecisionTreeRegressor对象。

feature\_names（可选）：特征名称的列表或数组，用于标识每个特征。

调用plot\_tree函数后，需要生成一个图形化界面展示决策树模型的图像。图像中的每个节点表示一个决策规则，包括特征和阈值。通过从根节点开始，沿着每个节点的边缘根据特征值的比较结果，最终到达叶节点，决策树模型将根据这些决策规则对新样本进行预测。

### torch.onnx.export函数

torch.onnx.export()是PyTorch库中用于将PyTorch模型导出为ONNX格式的函数。它提供了一种方便的方式将PyTorch模型转换为跨平台的ONNX模型，从而可以在不同的深度学习框架中进行模型部署和推理。

参数说明：

model：要导出的PyTorch模型对象。

x：输入模型的示例输入。

f：导出的ONNX模型文件的路径和名称。

operator\_export\_type：指定导出运算符的类型

export\_params（可选）：指定是否导出模型的参数（权重）。

opset\_version：指定导出模型的ONNX运算符集版本。

调用torch.onnx.export()函数后，将会生成一个包含模型结构和权重的ONNX文件。导出的ONNX模型可以用于跨平台部署和推理，可以在不同的深度学习框架中加载和使用。

通过torch.onnx.export()函数，我们可以方便地将训练好的PyTorch模型转换为ONNX格式，随后可以在Netron中加载和查看模型的结构、层次关系、参数等信息，从而更好地理解和分析模型。

### Netron可视化工具介绍

Netron是一种用于神经网络、深度学习和机器学习模型的可视化工具（logo见图13‑1），它可以为模型的架构生成具有描述性的可视化。Netron主要由JavaScript语言实现。Netron的主要特点包括：

1）多框架和多格式支持：Netron支持ONNX、TensorFlow Lite、Caffe、Keras、Darknet、PaddlePaddle、ncnn、MNN、Core ML、RKNN、MXNet、MindSpore Lite、TNN、Barracuda、Tengine、CNTK、TensorFlow.js、Caffe2和UFF。它还实验性支持PyTorch、TensorFlow、TorchScript、OpenVINO、Torch、Vitis AI、kmodel、Arm NN、BigDL、Chainer、Deeplearning4j、MediaPipe、ML.NET和scikit-learn，Netron支持相应的框架信息总结如表13‑1、表13‑2所示；

2）跨平台支持：Netron可以运行在多种操作系统上，包括 Windows、macOS 和Linux，方便用户在不同环境中使用；

3）直观的可视化界面：Netron提供一个直观的图形界面，以树状结构展示模型的网络层次和参数。用户可以浏览模型的结构、查看各层的输入输出尺寸、参数数量等，帮助理解模型的组成和特性。

Netron可视化工具安装起来也比较简单，下载对应的.exe安装包后根据指示安装即可，除了在本地安装软件之外，也可以在Netron在线网站中上传相应的模型文件进行可视化，其在线地址为<https://netron.app/>。



图13‑1 Netron logo

表13‑1 Netron支持的框架

|  |  |
| --- | --- |
| 框架名称 | 对应文件名称 |
| ONNX | .onnx、.pb、.pbtxt |
| Keras | .h5、.keras |
| Core ML | .mlmodel |
| Caffe | .caffemodel、.prototxt |
| Caffe2 | Predict\_net.pb |
| Draknet | .cft |
| MxNet | .model |
| TensorFlow Lite | .tflite |

表13‑2 Netron实验性支持的框架

|  |  |
| --- | --- |
| 框架名称 | 对应文件名称 |
| TorchScript | .pt、.pth |
| Pytorch | .pt、.pth |
| Torch | .t7 |
| PaddlePaddle | .zip |
| Scikit-learn | .pkl |
| TensorFlow.js | .pb |
| TensorFlow | .pb、.ckpt、.index |