|  |  |
| --- | --- |
|  |  |
|  |  |

**基于人工智能的智能运维方案**

**上海理想信息参产业（集团）有限公司**

**2022年9月**

目 录

[第一章 智能运维相关概念与技术 3](#_Toc115252223)

[1.1 背景介绍 3](#_Toc115252224)

[1.2 大数据及机器学习概述 3](#_Toc115252225)

[1.3 跨数据中心分布式系统 5](#_Toc115252226)

[1.4 系统静态数据及动态数据 6](#_Toc115252227)

[1.5 网络数据采集相关技术 7](#_Toc115252228)

[第二章 智能运维需求分析与总体方案 10](#_Toc115252229)

[2.1 需求分析 10](#_Toc115252230)

[2.1.1 系统功能性需求 10](#_Toc115252231)

[2.1.2 系统非功能性需求 16](#_Toc115252232)

[2.2 架构设计 18](#_Toc115252233)

[2.2.1 系统逻辑架构 18](#_Toc115252234)

[2.2.2 系统网络架构 21](#_Toc115252235)

[2.2.3 系统部署架构 21](#_Toc115252236)

[第三章 详细设计与实现 21](#_Toc115252237)

[3.1 基于Tensorflow的告警预测实现 32](#_Toc115252238)

[3.1.1 据特征选取 33](#_Toc115252239)

[3.1.2 构建样本数据集 34](#_Toc115252240)

[3.1.3 构建预测模型 36](#_Toc115252241)

[3.1.4 训练及测试 38](#_Toc115252242)

[3.2 应用效果 39](#_Toc115252243)

# 智能运维相关概念与技术

# 项目背景介绍

# 大数据及机器学习概述

在宽带互联网、移动互联网及各类私有云数据专网高速发展的背景下，数据信息爆炸已经成为人们所熟知的话题。当企业面对信息时代“海量”数据时，需要考虑如何利用数据来产生价值。人们讨论“大数据”时，不仅仅单纯是谈数据，还包括大数据平台、围绕大数据采集处理的工具以及对大数据进行分析和可视化的软件系统[9]。业界通常把大数据的特征概括成三个“V”，包括体量大（Volume），速度快（Velocity）和多模态（Variety）[10]。在大数据的处理技术中，基于Hadoop或者Spark的解决方案是企业应用比较广泛的解决方案。在Hadoop的软件体系中，其核心组件主要有：HDFS、HBase 、MapReduce、Hive、Mahout等。

在的机器学习技术中，以Python语言为基础的机器学习程序库应用较为广泛，例如Python语言中的sk-learn机器学习库非常优秀，sk-learn机器学习库囊括了几乎所有流行的机器学习算法的实现，用户只需要准备好数据，调用相应的算法库就可以完成训练及预测工作，其缺点是不支持GPU模式的分布式算法。

1. HBase数据库技术

HBase数据库建立在Hadoop大数据平台中的HDFS分布式文件系统之上，并且能提供高性能、高可靠性、列存储、弹性扩展、支持实时读写的数据库[11]。HBase的主要用键值对(Key-Value)方式的来存储松散型数据。通过横向扩展来增加计算和存储能力。HBase中的表一般具有这样的特点[12]：

1. 容量大：一个表可以具有上亿行、上百万列；
2. 面向列:面向列(族)的数据存储及控制，列(族)可以独立检索。
3. 稀疏:对于值为空(null)的列，并不占用存储空间，因此也非常适合稀疏表的数据存储。

[HBase](http://lib.csdn.net/base/hbase)的表数据是三维有序存储的，通过Row Key（行键），Column Key（Column Family和Qualifier）和TimeStamp（时间戳）这个三个维度，可以对[HBase](http://lib.csdn.net/base/hbase)中的数据进行快速定位。

表2-1 HBase数据库中表数据视图

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Column Family 1 | | … | Column Family N |
| 行键  (Row Key) | 列 1 | 列 2 | … | 列 N |
| Key 1 | value11 | value12 | … | value 1N |
| Key 2 | value21 | value22 | … | value 2N |
| Key 3 | value31 | value32 | … | value 3N |
| … | … | … | … | … |

HBase数据库中行键(Row Key)能够唯一定位一行记录，HBase查询的时候，通常采用以下几种方式[13]：

1. 根据Row Key获取唯一一条最新时间戳的记录，此时采用HBase的API 中的get方法；
2. 通过设置起始Row Key 及 终止Row Key进行条件扫描，此时采用HBase的 API中的scan方法；
3. 直接读取整张表中所有行记录，这种方式不常用，因为一次性读取表中所有的数据会导致内存空间不足。
4. 机器学习

机器学习在计算机科学技术中属于人工智能研究与应用的一个分支[14]。自1997年IBM公司的深蓝（Deep Blue）超级计算打败国际象棋大师卡斯帕罗夫后，人工智能引起了全世界的瞩目，之后经过多年的发展，随着大数据存储能力提升、训练算法的增强以及图形处理芯片GPU的广泛应用,在机器学习的基础上，深度学习得到了更大的发展。

机器学习以数据为基础，通过计算机算法进行自动训练及发现规律，而后对“未知”数据进行预测。机器学习所解决的问题是我们无法通过常规编程技术（固定的流程、固定的规则、固定的代码）来解决。所以机器学习程序的一个显著的特点是它能够从历史数据中“学习”规律，从而具备对“未知”数据进行预测的能力,通常把对“未知”数据的预测能力称为 “泛化力”(Generalization)[15]。机器学习是研究人工智能的一个强有力的工具，因此越来越多的企业通过机器学习技术使自己的信息系统具备一定的智能，更好地为业务创造价值。

1. Tensorflow深度学习框架

随着2016年初AlphaGo围棋程序击败韩国著名围棋大师李世石后，人们感受到人工智能的大潮已经来临了，人工智能的应用也好像在一夜之间遍地开花了。此后深度学习(甚至更进一步的强化学习)成为人工智能研究领域的热点，各种深度学习的开源框架也不断出现，其中Google公司在2015年11月将自己的分布式机器学习框架Tensorflow开源[16]，凭借Google公司在全球的巨大影响力，以及其在人工智能领域的顶尖水平，Tensorflow深度学习开源框架迅速引起各高校、研究机构、企业研发团队等高度关注。同时越来越多的企业在构建分布式机器学习系统时采用Tensorflow深度学习开源框架。基于Tensorflow开源框架构建机器学习系统具备代码简洁、优雅、部署方便等优点，从而深受广大程序员的喜爱。

另外基于Tensorflow实现的机器学习系统也可以运行在GPU上[17]，从而使模型在训练数据集上能够以更短的时间完成训练，相比Python的sk-learn机器学习库有着巨大的优势。

# 跨数据中心分布式系统

1. 分布式软件系统

分布式软件系统不是一个新的概念，早期的Oracle RAC (Real Application Cluster)[18]数据库系统就是一个双节点的分布式数据库系统。近年来随着人们越来越重视大数据,以及各种基于大数据的应用的迅速发展，分布式系统的作用也越来越显得非常重要，因为基于大数据的系统所面临的数据规模巨大，实时性要求也比较高，采用传统的应用软件架构以及硬件体系架构无法满足需求，或者实现的代价非常高。

随着大数据平台技术的发展及分布式框架技术的发展，越来越多的应用会基于分布式软件架构进行构建，从而使应用具备良好的高可用性及高性能等要求，满足大数据存储处理的需求。常见的分布式系统有Hadoop大数据平台中的HDFS分布式文件系统、HBase分布式数据库等，另外Zookeeper也是一个分布式的应用程序协调服务，Zookeeper管理和协调HDFS集群中的NameNode节点以及HBase集群中的HMaster节点[19]。

分布式系统的实现异常复杂，因为多节点分布式系统相比单节点系统而言，需要面临远程调用(Remote RPC)、进程间通信、分布式服务、分布式文件系统、数据同步、分布式事务等各种复杂问题。

分布式软件框架为分布式系统的构建提供基础平台和组件,从而使各种分布式系统的开发和建设能够更加便捷、高效、安全，并且满足高可用性及高性能等要求。常见的分布式软件框架有Apache Dubbo及OSGI[20]。其中Apache Dubbo主要是提供高性能的分布式远程调用RPC服务，而OSGI则侧重提供Java 动态化、模块化功能，同时提供分布式远程调用服务。

1. 跨数据中心分布式系统

在电信行业的数据业务中，架构师在构建软件系统时，需要考虑“灾难备份”（简称灾备）的需求，即当一个数据中心因为“灾难”事件而无法提供服务时，另外一个数据中心中的备份系统需要在短时间内启用，保证业务系统整体的高可用性及业务连续性，从而把因“灾难”而导致的业务影响降至最低。

跨数据中心的分布式系统构建来自于业务的灾备需求，政府或者企业客户考虑到自身业务的重要性，对系统的可用性提出了极高的要求，要求系统部署在地理位置上是跨数据中心的，这种软件部署架构具备一定程度的灾备能力，能够应对例突然断电、突发水灾、突发火灾、恐怖袭击等灾难性事件，另外还有更高要求的灾备是针对地震导致的数据中心瘫痪后，启动异地备用数据中心短时间内恢复业务。灾备的需求导致系统的架构必须考虑跨数据中心来进行设计，这对软件架构提出了更高的要求。各个行业中对跨数据中心分布式系统的建设没有统一的标准，基本上在不同的行业，不同的运营商根据自身业务和资源的特点进行个性化设计和实现。

# 系统静态数据及动态数据

在网络监控系统中被监测的目标对象统称为网元(Network Element)。网元是IT监控系统中可以监视和管理的最小单元。一切可以被监视和管理的单元都称为网元[21]。



图2-3 静态数据(D)和时间(T)的关系

在电信行业的数据业务中，描述网元特征的信息数据为静态数据，而描述网元随着时间变化而不断变化的信息数据为动态数据。

**静态数据：**在信息系统不间断运行状态下，数据量的增长并不会随着运行时间的增长而显著的增长，数据量（D）和时间（T）的关系可以用如图2-3所示。

该函数图形表示了一个分段函数。在实际应用中，例如系统刚上线是，签订了一个客户合同，上线了1个大客户和1000个待监控设备，此时系统中增加的静态数据就是1个客户的基本信息和1000个设备的基本信息，后来每隔几个月或者半年才签订一个客户合同，上线了部分客户和一些设备，系统的这部分数据量呈现阶梯式上升。随后1年或者3年后，随着客户服务合同时间到期，客户不再续签服务合同，系统则下线了一批客户和设备，系统静态数据阶梯式减少。因此总结下来，所谓静态数据是指数据的增加不会随着时间的变化呈现线性增长或者指数级增长，而且这部分数据是结构化的数据，这样的静态数据最合适使用关系型数据来处理，因为数据总量并不大，而且是高度结构化的数据。

**动态数据：**在信息系统不间断运行状态下，数据的增长会随着时间的变化而呈现抛物线式增长或者指数级的连续性增长。数据量（D）和时间（T）的关系可以用如下函数图形表示：



图2-4 动态数据(D)和时间(T)的关系

这个函数图形表示数据的增长随着时间呈现抛物线函数甚至是指数函数式增长。在实际应用场景中，这种数据是软件系统依据静态数据信息，通过定时调度任务的周期性执行，不断的采集、归并、计算而产生的新数据，随着时间的变化动态数据的数据量的增长非常显著。针对系统中动态数据随着时间的推移不断增长的情形，不能考虑单纯使用关系型数据库方案，还必须考虑分布式数据库(例如HBase)同关系型数据库的混合的应用方案。

# 智能运维需求分析与总体方案

本章阐述了系统的需求分析与总体架构。首先讨论了电信专线网监控系统的建设目标及功能性需求，并对系统用户及角色加以分析，然后讨论系统的非功能性需求，最后讨论了系统的逻辑架构、网络架构及部署架构。

# 需求分析

## 异常感知

1.监控指标的数据特征自动识别：基于机器学习算法对原始的性能数据和告警数据特征识别并进行提取

表2-5 网络故障告警预测特征数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 维度编号 | 维度名称 | 取值类型/范围 | 数值规范化 |
| 1 | Labels | 字符串 | One-Hot编码 |
| 2 | Mid | 字符串 | One-Hot编码 |
| 3 | Entity | 字符串 | One-Hot编码 |
| 4 | Timestamp | 字符串 | One-Hot编码 |
| 5 | Value | 字符串 | One-Hot编码 |
| 6 | Severity | 字符串 | One-Hot编码 |
| 7 | Idc | 字符串 | One-Hot编码 |
| 8 | StrValue | 字符串 | One-Hot编码 |
| 9 | Model | 字符串 | One-Hot编码 |
| 10 | Rule | 字符串 | One-Hot编码 |

定义：特征提取是将任意数据（如文本或图像）转换为可用于机器学习的数字特征

（注：特征值化是为了计算机更好的去理解数据）

特征提取有三类场景：

1）字典特征提取(特征离散化)

2）文本特征提取

3）图像特征提取（深度学习）

从目前的需求来分析，应该属于字典特征提取和文本特征提取的应用场景：

实现方案：Python 字典特征提取API：

sklearn.feature\_extraction

字典特征提取

|  |
| --- |
| 作用：对字典数据进行特征值化  sklearn.feature\_extraction.DictVectorizer(sparse=True,…)  DictVectorizer.fit\_transform(X) X：字典或者包含字典的迭代器，返回值：返回sparse矩阵  DictVectorizer.inverse\_transform(X) X：array数组或者sparse矩阵 返回值：转换之前数据格式  DictVectorizer.get\_feature\_names() 返回类别名称  实例化类DictVectorizer  调用fit\_transform方法输入数据并转换（注意返回格式） |

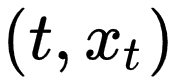
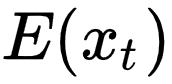
文本特征提取

|  |
| --- |
| 作用：对文本数据进行特征值化  sklearn.feature\_extraction.text.CountVectorizer(stop\_words=[])  返回词频矩阵  CountVectorizer.fit\_transform(X) X：文本或者包含文本字符串的可迭代对象 返回值：返回sparse矩阵  CountVectorizer.inverse\_transform(X) X：array数组或者sparse矩阵 返回值：转换之前数据格  CountVectorizer.get\_feature\_names() 返回值：单词列表  sklearn.feature\_extraction.text.TfidfVectorizer  实例化类CountVectorizer  调用fit\_transform方法输入数据并转换 （注意返回格式，利用toarray()进行sparse矩阵转换array数组） |

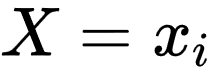
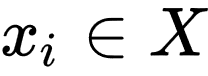
2.时序数据异常检测：基于时序算法对指标数据进行实时异常检测

通常时序数据的异常值主要分为三类：

1.异常值 Outlier 给定输入时间序列X，异常值是时间戳值对

，其中观测值 IMG_257，与该时间序列的期望值（即 ）不同。

2.[波动点](https://www.zhihu.com/search?q=%E6%B3%A2%E5%8A%A8%E7%82%B9&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b) Change Point 给定输入时间序列 x，波动点是指在某个时间 t，其状态（行为） 在这个[时间序列](https://www.zhihu.com/search?q=%E6%97%B6%E9%97%B4%E5%BA%8F%E5%88%97&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b)上表现出 与 t 前后的值不同 。

3.异常时间序列（ Anomalous Time-series ) 给定一组时间序列，异常时间序列 是在X 上与大多数时间序列值不一致的部分。

异常检测方法主要分为三大类：

1.基于统计模型 基于统计模型的异常点检测技术将所有数据构建成一个数据模型，其认为异常点是那些与模型不能完美拟合的对象。

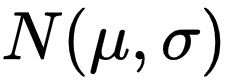
2.基于邻近度 通常可以在对象之间定义邻近性度量。异常对象是那些远离大部分其他对象的对象。

3.基于密度的技术 对象的密度估计可以相对直接计算，特别是当对象之间存在邻近性度量时。当一个点的局部密度显著低于它的大部分近邻时，可能会被看作是异常的。

### 基于统计模型的异常点检测

基于数据，构建一个概率分布模型，得出模 型的[概率密度函数](https://www.zhihu.com/search?q=%E6%A6%82%E7%8E%87%E5%AF%86%E5%BA%A6%E5%87%BD%E6%95%B0&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b)。通常，异常点的概率是很低的。

### 基于[正态分布](https://www.zhihu.com/search?q=%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":261993967}" \t "https://cloud.tencent.com/developer/article/_blank)的一元异常点检测

假设数据集由一个正态分布产生，该分布用 表示（如图 11 -1 所示），其中，IMG_257

和IMG_258分别表示均值和标准差 。 数据落在士3 个 IMG_259之外的区域的概率仅有 0.27%，落在±4 个也IMG_260之外的 区域的概率仅有 001%，几乎不可能发生，故将其判定为异常点 。 可以看出，异常点的判定与 我们所选定的标准（几个 IMG_261）有关，不是固定的。

### 多元正态分布的异常点检测

对于多元高斯分布检测，我们希望使用类似于一元高斯分布的方法。 例如，如果点关于估计的数据具有低概率，那么就把它们分类为异常点。

实际上[马氏距离](https://www.zhihu.com/search?q=%E9%A9%AC%E6%B0%8F%E8%B7%9D%E7%A6%BB&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b)也是统计算法，点到基础分布的Mahalanobis距离与点的概率直接相关

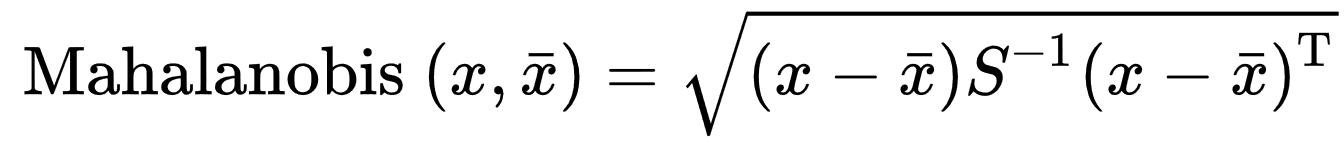
综上所述，两种基于统计模型的异常点检测方法，需要建立在标准的[统计学](https://www.zhihu.com/search?q=%E7%BB%9F%E8%AE%A1%E5%AD%A6&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra=%7b)技术（如分布 参数的估计）之上。这类方法对于低维数据效果可能较好，但是对于[高维数据](https://www.zhihu.com/search?q=%E9%AB%98%E7%BB%B4%E6%95%B0%E6%8D%AE&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":261993967}" \t "https://cloud.tencent.com/developer/article/_blank)，数据分布非常 复杂，基于统计模型的检测效果会比较差。

### 基于邻近度的异常点检测

### 马氏距离

对于一个多维数据集D，假设至是[均值向量](https://www.zhihu.com/search?q=%E5%9D%87%E5%80%BC%E5%90%91%E9%87%8F&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":261993967}" \t "https://cloud.tencent.com/developer/article/_blank)，那么对于数据集D 中的其他对象x ，从x

到数据均值（质心）的 Mahalanobis 距离（马氏距离）为：

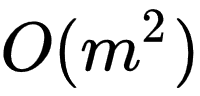


其中，x 为数据集D 中的元素，Z为数据均值，S为[协方差矩阵](https://www.zhihu.com/search?q=%E5%8D%8F%E6%96%B9%E5%B7%AE%E7%9F%A9%E9%98%B5&search_source=Entity&hybrid_search_source=Entity&hybrid_search_extra={"sourceType":"article","sourceId":261993967}" \t "https://cloud.tencent.com/developer/article/_blank)。

容易证明： 点到基础分布的Mahalanobis距离与点的概率直接相关，等于点的概率密度的对数加上一个常数。因此，可以对Mahalanobis距离进行排序，距离大的，就可以认为是异常点。

### KNN

KNN 算法是基于邻近度的算法， 不需要对数据集进行统计模型的拟合，可以直接用距离来识别异常点。但是，这种基于距离的算法也有其明显的缺点：①时间复杂度为

, 这意味着如果数据量比较大，会导致计算代价过高，效率低下：②对k 值的选取非常敏感， 并且k 值的选取是全局的，不能处理具有不同密度区域的数据集。

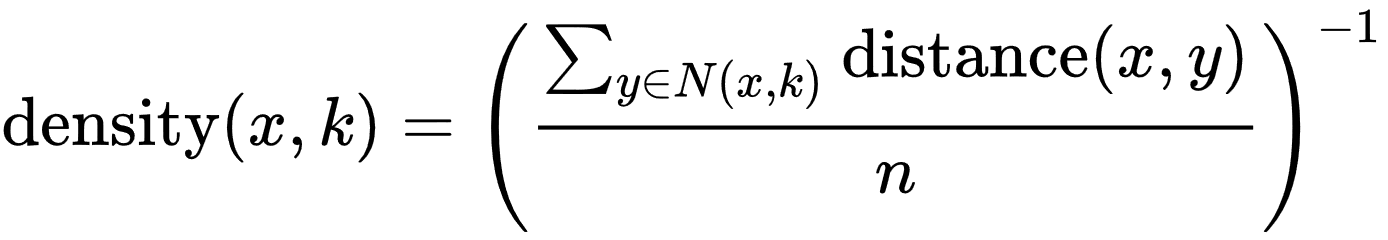
### 基于密度的异常点检测

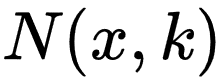
从基于密度的观点来说，异常点是低密度区域中的对象。

定义密度的方法有以下三种。

### 逆距离

一个对象的密度为该对象周围k个最近邻的平均距离的倒数。



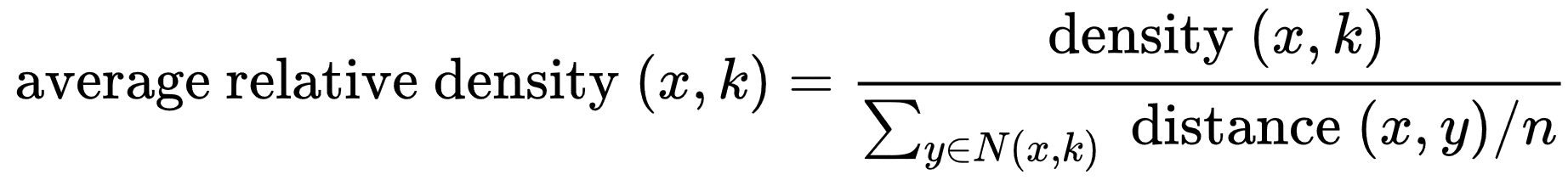
其中， 是x的k个最近邻的集合， IMG_266是该集合的大小 。

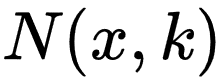
### 半径 d 内的个数

即一个对象周围的密度等于该对象指定半径 d 内对象的个数。d是人为选择的，那么这个d的选择就很重要了。

### 相对密度

即用点 x 的密度与它最近邻 y 的平均密度之比作为相对密度。



其中， 是x的k个最近邻的集合，n是该集合的大小 。

参见

（<https://cloud.tencent.com/developer/article/1918901>）

监督模型训练：从原始的性能数据和告警数据中训练检测模型

在机器学习中，无监督学习（Unsupervised learning）就是聚类，事先不知道样本的类别，通过某种办法，把相似的样本放在一起归位一类；而监督型学习（Supervised learning）就是有训练样本，带有属性标签，也可以理解成样本有输入有输出。

所有的回归算法和分类算法都属于监督学习。回归（Regression）和分类（Classification）的算法区别在于输出变量的类型，定量输出称为回归，或者说是连续变量预测；定性输出称为分类，或者说是离散变量预测。

## 故障诊断

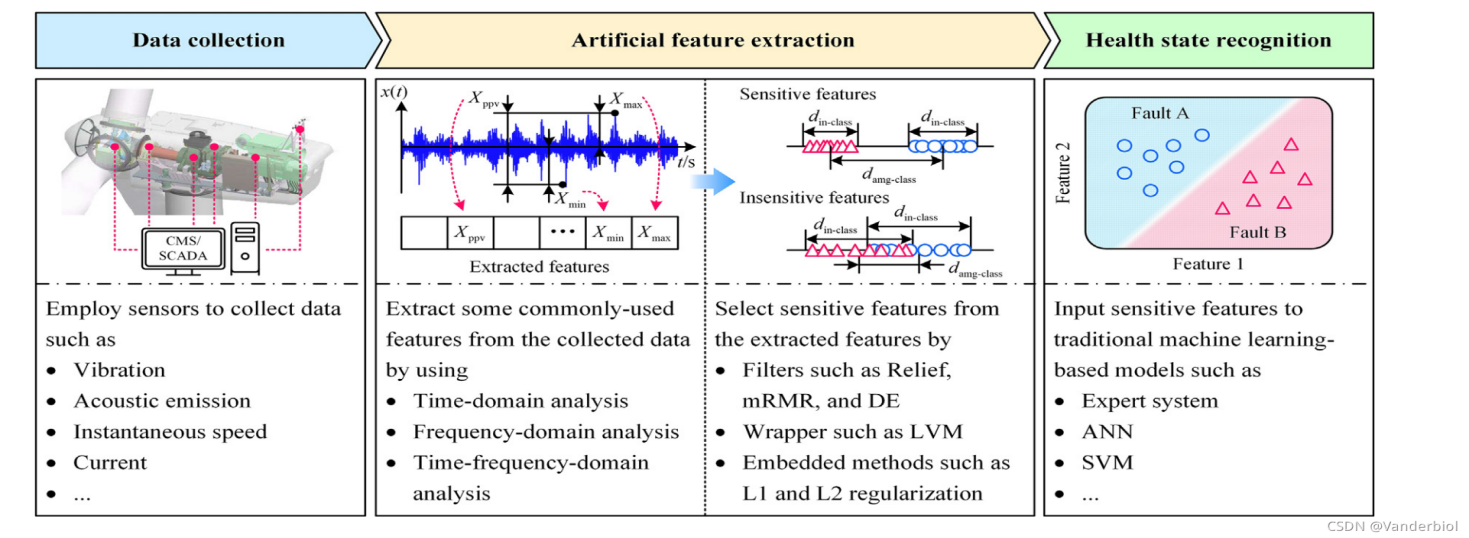
基于机器学习的故障根因诊断：对某种故障进行故障诊断

智能故障诊断（IFD）是指将机器学习理论，如人工神经网络（ANN）、支持向量机（SVM）和深度神经网络（DNN）应用于机器故障诊断。这种方法利用机器学习理论，从采集的数据中自适应地学习机器的诊断知识，而不是利用工程师的经验和知识。具体而言，IFD需要构建一个诊断模型，该模型能够自动将收集的数据与机器的健康状态之间的关系连接起来。

机器学习的早期研究可追溯到1950年，1980年左右成为了人工智能的一个重要方向，并于2010年开始得到了广泛的应用。在此期间发明了许多传统理论，如ANN、SVM、k-近邻算法（kNN）和概率图形模型（PGM）等。这些理论促进了IFD的出现，包括基于专家系统的方法、基于神经网络的方法、基于SVM的方法等。在这些方法中，故障特征是从收集的数据中人工提取，然后用故障特征来训练能够自动识别机器状态的诊断模型。在传统机器学习的帮助下，诊断模型开始建立所选特征与机器健康状态之间的关系，这削弱了人类劳动对机器故障诊断的贡献，将机器故障诊断推向了人工智能时代。

2.基于传统机器学习的之智能故障诊断

在过去的故障诊断体系中，一些传统的机器学习理论，如ANN和SVM，被应用于机器故障诊断。诊断程序包括三个步骤，即数据收集、故障特征提取和状态识别，如图2.1所示。

图2.1

2.1数据采集

通常数据采集的方法是将传感器安装在机器上以不断采集数据。不同的数据需要使用不同的传感器，如振动、声发射、温度和电流互感器。电流数据在电动机械的故障诊断中起着重要作用。仅使用电流互感器即可轻松收集此类数据。根据发现，来自多源传感器的数据具有互补信息，与仅使用来自单个传感器的数据相比，这些信息可以进行融合以实现更高的诊断准确率 。

2.2故障特征提取

传统机器学习理论中故障特征提取是由人手动完成的，包括两个步骤：首先，从收集的数据中提取一些常用的特征，如时域特征、频域特征和时频域特征。这些特征包含反映机器健康状态的健康信息。其次，使用特征选择方法，如过滤器、包装器和嵌入方法，从提取的特征中选择对机器健康状态敏感的特征。这有利于去除冗余信息，进一步提高诊断效果。

2.2.1冗余信息处理

冗余可能会增加计算成本，甚至导致维数灾难。为了削弱这一问题，一种方法是从收集的机器健康状态中选择敏感特征。它们可以分为三类，即基于过滤器、包装器和嵌入式方法。

基于过滤器方法直接预处理收集的特征，这些特征独立于分类器的训练。

与基于过滤器的方法不同，基于包装器的方法侧重于特征选择与训练分类器的交互作用。换句话说，分类器的性能用于评估所选特征集。如果所选特征子集不能产生最佳分类精度，则在下一次迭代中重新选择另一子集，直到所选特征以最有利的性能强制分类器。

基于嵌入式方法将特征选择与分类器训练相结合。通常，它们在分类器的优化对象上引入正则化项，并在分类器训练完成后自动选择特征。

2.3状态识别

健康状态识别使用基于机器学习的诊断模型来建立所选特征与机器健康状态之间的关系。为了达到这一目的，首先用标记样本对诊断模型进行训练。然后，当输入样本未标记时，模型能够识别机器的健康状态。

## 动态阈值

动态阈值模型训练：基于原始的性能数据和告警数据自动训练动态阈值模型

通常阈值设置方法会参考过去一段时间内的均值、最大值以及最小值，我们也同样应用此方法。取过去一段时间（比如T窗口）的平均值、最大值以及最小值，然后取max-avg和avg-min的最小值。之所以取最小值的原因是让筛选条件设置的宽松一些，让更多的值通过此条件，减少一些漏报的事件。

v2-17e6190d2a2837971a1f28be0e2c0220_720w

## 故障处理

* 1. **角色分析**

基于上述功能需求分析，电信专网监控系统涉及到两大类用户角色，一类是客户侧用户，另外一类是电信运营商侧用户，其中电信运营商侧用户的角色包括服务台(一线工程师)、二线工程师、事件经理、专家团队、维护工程师。

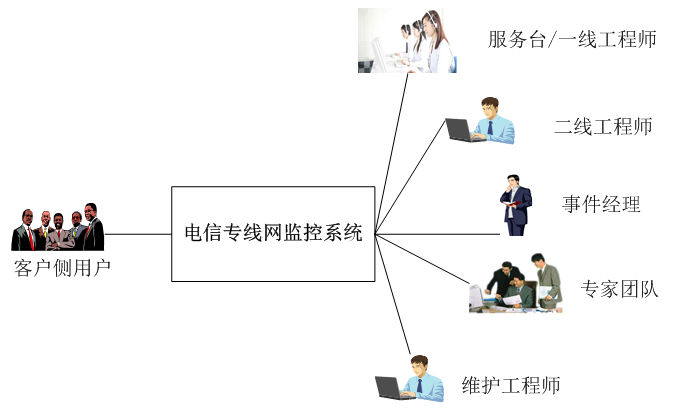


图3-1 电信专网监控系统用户视图

客户侧用户，是指政企客户方所有使用电信专网监控系统的用户，也是电信专线网数据业务所服务的对象。

服务台(一线工程师)，分为三班进行监控大厅值班，7\*24小时无间断值守监控告警板，并且负责对告警信息进行确认、远程诊断，通过告警信息提交事件单，进行故障跟踪处理。

二线工程师，接收事件升级的指派，根据升级后的事件，远程诊断，故障恢复。

事件经理，负责对升级上来的事件进行分派，同时负责核实已经处理的故障事件单，负责跟踪故障处理的进度，并且对服务台/一线工程师、二线工程师进行考核，对客户发布公告、通知信息等。

专家团队，对事件经理分派过来的比较困难的故障进行诊断，提出专家建议，协助排除故障。

维护工程师，负责电信专网监控系统日常巡检、网络、主机、应用维护，系统升级部署，系统变更维护、发布管理、问题管理、知识库管理等工作。

* 1. **用例分析**

在讨论了系统功能性需求和系统角色之后，便可以依据系统角色设计系统用例。

客户侧用户，是系统所服务的对象，客户侧用户在系统使用权限方面更多的是查看性功能，另外客户侧可以使用在线客服功能，还可以提交服务请求，询问或咨询具体的问题。其具体用例图如下：



图3-2 客户侧用户用例图

服务台(一线工程师)，是在电信专网监控系统中离客户最近的用户角色，接收客户的热线电话，提供在线客户服务，并且提供7\*24小时无间断服务。其具体用例图如下：



图3-3 服务台(一线工程师)用例图

二线工程师，负责接收来自于一线工程师的升级事件，并且执行远程诊断，完成故障恢复工作，同时负责监控服务上线的相关工作，其具体用例图如下：



图3-4 二线工程师用例图

事件经理，是电信专网监控系统面向客户提供服务的流程枢纽中心，负责事件升级及关闭审核，并且能够向专家团队进行事件升级，同时负责对一、二线工程师进行考核管理。其具体用例图如下：



图3-5 事件经理用例图

专家团队，负责解决各种“疑难杂症”类型的故障诊断及排除，接受事件经理升级上来的事件单（包含故障信息）。其具体用例图如下：



图3-6 专家团队用例图

维护工程师，负责整个系统的维护，同时对研发团队提交的上线模块进行部署，记录系统软件、硬件变更信息。其具体用例图如下：



图3-7 维护工程师用例图

* 1. **核心业务流程**

图3-8 监控服务上线流程

故障处理流程，这个流程的目的是明确告警产生之后，如何进行故障排除服务。故障有系统监控之后，产生告警，监控大厅一线工程师对告警进行确认，然后产生事件单，进行故障处理跟踪。具体流程如下：



图3-9 故障处理流程

# 架构设计

## 系统逻辑架构

本文在讨论逻辑架构设计时采用分层方式来实现，将系统不同功能划分在不同的逻辑层次中，便于对系统整体功能的理解。电信专线网监控系统的逻辑架构分为四个层次：数据采集层、数据处理层、数据存储层及业务管理层。

图3-11电信专线网监控系统逻辑结构图



其中有三个逻辑层负责对设备监控数据进行采集、处理、存储，一个逻辑层完成对电信专线网监控业务的管理和数据展现。把数据采集和数据处理分离，可以实现多个数据采集子系统和多个数据处理子系统的分布式集群化部署，并且通过消息队列来连接数据采集和数据处理，能够有效地提高数据采集和处理的吞吐量。

1. **数据存储层架构**

通过上述逻辑架构设计，明确在电信专线网监控系统的数据存储层包括MySQL数据库集群以及构建在HDFS分布式文件系统之上的HBase分布式数据库。关系型数据库MySQL用来存储系统的静态型数据，这类数据的体量不大，但是对查询和存储的实时性和灵活性要求比较高。HBase分布式数据库用来存放系统动态增长型数据，这类数据随着时间的推移，增长速度比较快，对数据的存储能力要求比较高，但是对数据查询的灵活性要求不高，因此非常适合采用HBase数据库。

1. **数据采集层子系统DCS（Data Collecting System）**

数据采集层子系统DCS实现的核心功能包括任务调度、采集驱动管理、内存缓存、采集指令数据发送、远程诊断命令、SNMP Trap接收器、NetFlow接收器。数据采集层的子系统DCS负责监控数据采集，由于电信专线网监控系统是面向多客户的集中式监控系统，通过专线的方式接入到不同的客户网络中，因此客户的监控设备分别由不同的DCS子系统进行数据采集，在网络配置层面可以通过路由器路由策略配置，把不同客户的网络映射到不同的DCS子系统，确保不同客户网络之间的隔离。



图3-12 电信专线网监控系统子系统之间数据交互图

1. **数据处理层子系统DPS(Data Processing System)**

数据处理层的子系统为DPS主要负责从消息队列中读取监控数据，并且进行数据归并、汇总计算，然后把数据送到存储层，分别存入MySQL数据库集群及HBase分布式数据库集群。同时数据处理层子系统DPS还有很多定时任务，包括临时数据清除，数据定时计算等功能。

1. **业务管理层各子系统**

DKS (Data Kernel System)子系统，负责静态数据的存储访问，同时还周期性把业务配置数据通过消息队列下发到各个DCS子系统中，DKS子系统管理了系统全部配置数据，DCS子系统管理的配置数据为DKS子系统的数据子集，DKS子系统可以同时连接数据存储层的MySQL数据库集群及HBase数据库， DWeb子系统通过调用DKS的API接口进行数据展现查询。

DES(Data Event System)子系统，负责集中处理故障告警，包括告警等级确定、告警关联、告警确认后的短信、邮件通知等动作，同时还依据告警信息产生事件单，进入DEngine子系统进行事件流转处理。

DEngine(Data Engine)子系统，通过流程模板的预先制定，触发流程运转，实现客户故障告警后期的服务处理工作，包括事件管理流程、变更管理流程、配置管理流程、问题管理流程等。

DWeb (Data Web)子系统，负责人机交互界面，实现客户侧用户和运营侧用户的操作访问，所有人工数据录入及查询工作都是通过DWeb子系统来实现，同时DWeb子系统还提供各种Restful接口[24]，为移动终端的应用程序访问提供接口服务。

DReport(Data Reporting )子系统，实现报表、报告服务功能，通过配置各种报表模块实现周报、月报、季度报、年报等功能，同时定时读取数据存储层的数据，根据报告、报表模板的数据映射，产生各种业务需求的报表、报告。

## 系统网络架构

## 系统部署架构

# 详细设计与实现

1. **设备故障告警数据采集**

本文在实现上述各类数据采集的基础上，通过预置规则进行阈值匹配，系统判断采集的数据指标值超过阈值时，则自动产生告警信息，告警信息被封装为Java对象进行传递。告警信息被统一送往DES子系统进行告警级别判定，而后分别送往MySQL数据库及HBase集群进行存储。

表4-5系统故障告警数据字段信息表

|  |  |
| --- | --- |
| 字段名称 | 字段说明 |
| Alarm\_ID | 告警ID |
| Root\_NE\_ID | 根网元ID |
| Root\_NE\_Name | 根网元名称 |
| NE\_ID | 网元ID |
| NE\_Name | 网元名称 |
| NE\_Addr | 网元地址 |
| Customer\_ID | 客户编号 |
| Customer\_Name | 客户名称 |
| Alarm\_Name | 告警名称 |
| Alarm\_Desc | 告警描述 |
| Alarm\_Type | 告警类型 |
| Alarm\_Status | 告警状态 |
| Alarm\_Severity | 告警严重等级 |
| Alarm\_Time | 告警发生时间 |
| Alarm\_ACK\_Status | 告警确认状态 |
| USER\_ID | 处理人ID |

在上述故障告警信息表格中，告警的时间非常重要，它是客户SLA(Service Level Agreement)协议执行的时间依据，同时本文的第四章将讨论利用设备状态数据、在线时长数据、性能数据、历史告警数据等，使用Tensorflow机器学习框架，建立模型并对特定时间的设备故障进行预测。

# 基于Tensorflow的告警预测实现

随着人工智能浪潮来临，机器学习已经渗透到各个行业的应用场景中。在基于大数据的系统实现后，进一步利用长期积累的监控数据进行机器学习，实现智能化应用场景，这些应用场景包括故障预测、监控数据挖掘、网络监控值班机器人等。本文着重讨论实现故障预测。



图4-22 网络故障预测实现的路径图

本文在着手实现机器学习时，选取了广泛使用的Python语言，最终选取了Tensorflow机器学习框架主要是基于以下考虑：

1. Python语言中sk-learn机器学习库目前不支持GPU运算，在处理大规模网络监控数据时，训练所消耗的时间可能比较长,效率比较低。
2. Tensorflow机器学习框架在业界处于领先地位，开源贡献者和软件开发者众多，遇到疑难问题能够迅速的找到解决方法，提高开发效率。
3. Tensorflow机器学习框架能够自由的搭建训练模型或者深度学习神经网络，能够适应未来灵活多变的需求。

## 据特征选取

本文在特征数据选取阶段，需要结合网络监控领域经验知识，对电信专线网监控业务中的监控数据进行选择抽取， 根据经验选择可能对设备发生故障产生影响的数据特征，这里选择的特征数据如下：

表4-10 网络故障告警预测特征数据表

|  |  |  |  |
| --- | --- | --- | --- |
| 维度编号 | 维度名称 | 取值类型/范围 | 数值规范化 |
| 1 | 设备类型 | 字符串 | One-Hot编码 |
| 2 | 生产商 | 字符串 | One-Hot编码 |
| 3 | 设备出厂月数 | (0,+∞) |  |
| 4 | 网络接口数(激活状态) | (0,+∞) |  |
| 5 | 设备所在地点 | 字符串 | One-Hot编码 |
| 6 | 平均CPU利用率 | (0，100] |  |
| 7 | 平均内存利用率 | (0，100] |  |
| 8 | 时间段 | [0,23] | One-Hot编码 |
| 9 | 接口平均带宽利用率 | (0，100] |  |
| 10 | 接口进数据包数 | (0,+∞) |  |
| 11 | 接口出数据包数 | (0,+∞) |  |
| 12 | 接口进字节数 | (0,+∞) |  |
| 13 | 接口出字节数 | (0,+∞) |  |
| 14 | 设备运行时长 | (0,+∞) |  |
| 15 | 设备ICMP时延 | (0,+∞) |  |
| 16 | 设备ICMP丢包率 | (0，100] |  |
| 17 | 接口进错误包数 | (0,+∞) |  |
| 18 | 接口出错误包数 | (0,+∞) |  |

在上述特征数据表中，对于“设备类型”字段，不同的厂商生产的设备型号不同，同一个厂商也会生产不同型号的设备，因此可以认为各个“设备类型”在数值上互相独立的，它们之间没有线性关联关系，本文在构建训练样本数据时，采用One-Hot编码进行数值规范化(Nomalization)处理 [30]。

同样进行One-Hot编码数值规范化的特征数据还有生产商、设备所在地点、设备运行所在时间段。本文基于以下考虑，选取这些特征数据：

1. 不同的设备生产商生产的设备质量不同，和设备故障有直接关联关系。
2. 设备所在地点不同意味着设备安置在不同的区域、不同的数据中心，区域气候差异、机房温湿度差异都可能对设备故障产生影响；
3. 设备运行时间段的不同，设备繁忙程度就会不一样，那么设备发生故障的概率就可能会不同。
4. 设备出厂月数也是要考量的一个特征数据，因为设备是由各种电子元器件构成，随着时间的增加，设备“年龄”越大发生故障的概率是越高的。

时间段这个特征数据本身是连续数值型的，不能认为“第10小时是第1小时的10倍”，因此设备运行时间段是需要进行One-Hot编码数值规范化处理；

## 构建样本数据集

在完成选取特征数据之后，接下来可着手构建样本数据集。构建样本数据需要考虑到机器学习程序在进行模型训练或者测试时的数据读取方式。一般来讲机器学习在读取训练数据时有两种方式：

表4-11 机器学习程序数据读取方式表

|  |  |
| --- | --- |
| 数据读取方式 | 优缺点对比 |
| 把训练以及测试数据全部加载到内存，机器学习程序直接在内存中读取数据进行训练和测试 | 优点：速度快、效率高；  缺点：只适合少量的数据的应用场景 |
| 分批次把训练以及测试数据加载到内存中，在机器学习程序进行训练或者测试时逐步提供数据 | 优点：可以对海量数据进行训练和测试  缺点：编程稍显复杂、效率略低 |

考虑到电信专线网监控系统数据体量较大，本文选择表4-11中的第二种数据读取方式，即让机器学习程序在训练和测试的过程中，分批次从文件中逐步读入数据。

图4-23 机器学习流程图

在图4-23的流程中，分为数据抽取和机器学习两个阶段。在数据抽取阶段通过编写预处理程序分别从MySQL和HBase两种类型的数据库中把数据抽取出来，生成指定格式的数据文件。

本文数据文件采用txt文件，文件中每一行表示一条样本数据，通过逗号来分隔不同的数据特征维度，最后一列是样本中目标数据（即故障标志：0表示无故障，1表示故障），具体格式定义如下：

|  |
| --- |
| 设备类型，生产商，设备出厂月数，网络接口数，设备所在地点，平均CPU利用率，平均内存利用率，时间段，接口平均带宽利用率，接口进数据包数，接口出数据包数，接口进字节数，接口出字节数，设备运行时长，设备ICMP时延，设备ICMP丢包率，接口进错误包数，接口出错误包数，故障标志\n |

图4-21 机器学习样本数据格式图

样本数据格式定义完毕后，还需要定义每个样本数据文件的数据行的条目，防止单个数据文件过大，从而导致数据加载到内存时出现内存不足的问题。

在本文的实现中定义单个数据文件的样本数据行为1000条， 并且选取电信专线网监控系统2017年全年的小时表汇总数据，由预处理程序从数据库中读入数据，分批生成多个样本数据文件。

本文选取小时粒度的数据来构建样本数据集，基于以下考量：

1. 短时间数据粒度会导致预测时间窗口较短，从而产生更多的预测结果，监控大厅的值班工程师可能会来不及处理大量的预测信息；
2. 长时间数据粒度会导致数据样本预测时间窗口拉长，对故障预测指导的及时性不够。

样本集构建的关键代码如下(Python代码)：

|  |
| --- |
| def generate\_data\_files(total\_rows):  # 数据文件目录名  data\_file\_dir = "datas";  # 单个文件中数据样本的条目  rows\_of\_file = 1000;  # 样本数据行索引  row\_index = 0;  # 数据文件编号  file\_index = 0;  # 用于1批行数据的缓存  batch\_row\_datas = []  for i in range(total\_rows):  # 构造单行数据文本(可通过查询数据库获取，然后字符串拼接而成)  i\_row = input\_row\_data()  batch\_row\_datas.append(i\_row)  row\_index += 1  # 达到1000行数据  if(row\_index % rows\_of\_file == 0):  file\_name = data\_file\_dir + "/" + str(file\_index) + ".data"  # 一批数据写入单个文件  write2file(file\_name, batch\_row\_datas)  file\_index += 1  # 清空 batch\_row\_datas  batch\_row\_datas = []  print(str(file\_index) + " files was generated!") |

经过上述数据预处理程序进行处理后，生成一批样本数据文件，每个样本数据文件为文本格式的明文。每一行为一条样本数据，共计18个特征维度和1个行尾的故障标志(0或者1)。其中有些特征维度在后续机器学习特征转换时，需要进行One-Hot编码处理，从而对特征的维度进行扩展。

## 构建预测模型

在完成样本数据构建完毕后，本文着手对故障预测建立机器学习模型。

在机器学习中特征向量描述为：

（x1,x2,…xn）

在无法判断选用什么样的模型对网络故障预测最为合适的情况下，考虑引入简单实用的线性模型建立故障预测模型：

Yprobility = w1 \* x1 + w2 \* x2 + … + wn \* xn + b (4.18)

其中 (w1,w2,…wn) 为模型权重向量， b 为模型的偏值 (bias)，对该模型进一步抽象，可以表示如下：

Yprobility = XWT + b (4.19)

其中X = （x1,x2,…xn）, W = (w1,w2,…wn)。

网络故障预测模型选定后，本文基于Tensorflow深度学习框架，建立基本的神经网络模型通过如下几个步骤实现。



图4-24 基于Tensorflow的神经网络模型构建流程

1. 样本数据读取和数值规范化

由于样本数据集是一系列文本文件的集合，因此本文采用Tensorflow提供的数据文件队列的方式来实现，通过分批次对文件中的样本数据进行读取，并且在每个批次的数据中，对样本数据的顺序进行打乱，从而使样本数据能够在模型上得到比较均匀的训练。

1. 定义需要进行训练的权重参数变量和偏值变量

|  |
| --- |
| with graph.as\_default():  with tflow.name\_scope("variables"):  # 待计算的权重值向量，由于目标为1个输出概率值，因此权重值矩阵的维数为 n x 1  # 其中n 为输入特征向量的维数， 由于这里的输入特征的维数为43  # 即针对这43维度的样本数据，有1套权重值参数  W = tflow.Variable(tflow.zeros([43, 1], dtype= tflow.float32), name="weights")  # 偏值bias  b = tflow.Variable(0.0, dtype= tflow.float32, name="b")  with tflow.name\_scope("global\_ops"):  initial = tflow.global\_variables\_initializer() |

在上述实现代码中，权重值参数的维度数为43，是因为对数据特征维度“生产商”和“时间段”需要进行数值规范化处理，这里采用One-Hot编码对这两个特征维度进行规范化处理，处理完毕后特征维度在原有18个特征维度基础上扩展到43个特征维度。

1. 定义训练模型、激发函数、损失函数及训练函数

在这一步实现中，本文把前面定义的故障预测模型转换为Tensorflow实现代码，由于预测目标是二分类问题(即故障或者非故障)，因此本文选择Tensorflow提供的Sigmoid概率分布函数来作为激发函数，模型的预测值相对于样本数据中的真实数据之间的差距由Tensorflow提供的交叉熵损失计算函数来实现[31]，而对损失函数的值的收敛则为模型训练的过程，模型训练由训练函数完成，可采用Tensorflow提供的梯度下降法对损失函数进行优化。

其关键实现代码如下：

|  |
| --- |
| # 模型定义  def model\_function(X):  return tf.matmul(X, W) + b  # 激发函数(Sigmoid概率分布函数)  def inference(X):  return tf.sigmoid(model\_function(X));  # 损失函数(交叉熵计算损失)  def loss(X, Y):  return tf.reduce\_mean(tf.nn.sigmoid\_cross\_entropy\_with\_logits(logits=model\_function(X), labels=Y))  # 训练函数  def train(loss):  # 学习率超参数  learning\_rate = 0.0001  # 梯度下降法优化器  return tf.train.GradientDescentOptimizer(learning\_rate).minimize(loss) |

## 训练及测试

Tensorflow以图计算的方式来搭建神经网络，训练的过程是通过不断迭代运行预先定义的神经网络图(Graph)，让数据在Tensorflow图上一遍又一遍的流动计算，在计算的过程中不断更新权重参数值和偏值，用梯度下降法(或者误差反向传播算法)不断优化损失函数的值，从而完成模型的训练。本文在对模型的两个关键参数进行调整优化的时候，不断观察模型训练的优化情况，最终选择效果较好的一组参数用于实际预测中。下面为各个梯度的训练次数情况及超参数学习率（learning\_rate）的值，观察到模型在样本数据集上表现情况。

表4-12 模型训练及测试优化情况表

|  |  |  |
| --- | --- | --- |
| 训练次数 | 学习率 | 预测准确性 |
| 10000 | 0.001 | 40% |
| 10000 | 0.0001 | 80% |
| 10000 | 0.00001 | 60% |
| 100000 | 0.001 | 70% |
| 100000 | 0.0001 | 80% |
| 100000 | 0.00001 | 60% |

通过分析上表中数据结果可以发现，在训练次数为10000的情况，学习率为0.0001时有比较理想的预测结果（达到80%准确性），通过提高训练次数和减小学习率有时反而出现预测准确性下降的情况。