白求恩人工智能详细设计文档

## 一、概述

WEKA是一款免费的、非商业化的、开源的Java类库，为数据挖掘任务提供了一序列的机器学习算法，包括数据预处理工具，分类，回归、聚类、关联规则以及可视化等。

白求恩智能分析系统主要是基于WEKA开发的，根据业务需求以及数据特点，构建WEKA的数据结构Instances，然后调用相应的机器学习算法实现指标的智能分析。

## 二、架构设计

在WEKA中，一个实例用Instance表示，一个样本由多个实例组成，用Instances表示。Instance是WEKA的核心，所有的算法都是使用这个数据结构进行建模，甚至包括数据的预处理，所以在最开始必须将输入的数据转换为Instances对象，接着在数据预处理阶段完成数据缺失的补全和数据平滑的处理，由于WEKA对于数据缺失的处理较为简单，一般使用某个常数填入（或替换），如果不能满足需求，则可以通过自定义的方法对数据的进行预处理，然后调用WEKA数据进行数据建模并预测，最后输出结果。架构如下图所示：

输入数据

数据转换

预处理

数据建模

预测

输出

基于扩展性的考虑，白求恩智能分析系定义了抽象类AbstractAI和接口IntelligentBethune，其中AbstractAI提供了构建Instances的基础方法，在业务没有特殊情况的条件下，可以直接调用相应的方法完成Instances生成；IntelligentBethune定义了predict和getResults方法，其中predict是数据建模和预测，getResults直接返回结果。原则上需要完成智能分析任务的业务应该基于这两个类结合来完成具体的实现。时间序列预测和时间序列异常检测的类图如下：



## 三、时间序列

时间序列是用时间排序的一组随机变量，时间序列的时间间隔可以是秒、分、日、周、月、季度、年、甚至更大的时间单位。

通常，最原始的数据可能是不按时间排序的，这样的数据没有构成时间序列，所以在拿到数据之后的第一步是组织数据、并按升序排序（必须是升序），排完序的时间序列可能并不完整或者总是会存在异常值。时间序列如果缺失数据是无法进行预测的，所以必须对缺失的数据进行补全；而异常的数据，在预测的时候对预测的结果产生很大的影响，导致误差很大，所以必须对异常数据进行平滑处理再进行预测。另外，对时间进行预测的时候，需要指定时间序列的周期。

时间序列的预测是由WEKA的WekaForcaster完成的，该接口需要传入一个具体的机器学习算法来完成数据的建模和预测，比如线性回归、高斯处理(实现了无超参数调整的线性回归)等。通过实验观察，发现线性回归的效果在处理季节性较为明显的数据时要好与高斯处理，但是高斯处理在预测数据为递增的时间序列时要明显好于线性回归，而且对于一些较为极端的数据也能做出更好的预测效果，而现线性回归则在遇到极端数据时，这个值会对预测结果产生较大的影响。根据实验的结果以及目前表空间传过来的递增型的时间序列这个特点，所以针对本需求的空间预测，首先采用高斯处理预测未来一个周期的使用量趋势，然后对这个周期进行一阶差分得到这个周期的空间消耗量并与给定的剩余空间进行比较，依次循环预测、差分和比较，直到消耗的空间等于或者大于剩余空间量，最后返回预测的时间点。

### 排序

Instances提供了排序的方法sort,但是需要提供时间属性的索引index，所以在生成Instances的时候，实现了根据属性值来自动识别是否是时间，AbstractAI接口的dataSets参数是一个二维数组，任意取一行并对每个值进行正则表达式判断，如果是时间，则取这个值的索引位置为时间属性的索引Index，在生成完Instances后调用sort传入index，完成排序。

### 时间序列缺失补全

对已经排序的Instances取相邻的两个时间进行相减得到时间差，把时间差作为主键存储在一个字典中，并把它的value赋值为1，遍历Instances，对时间差进行统计。从字典中取统计量最大的时间差作为时间基准差，再次遍历Instances，计算相邻两个数的时间差，如果时间差不等于基准差时，首先用时间差除以基准差得到缺少的数量，然后取这两个相邻的第一个的时间和值，再取第二个的相邻的值，两个值相加除以缺少的数量得到一个平均值，这个平均值作为缺少数据的增量，按照缺少的数量依次循坏，从相邻的第一个时间和值分别加上时间基准差和增量作为下一个缺失的数据 。通过此方式完成所有缺失数据的补全。

### 特殊异常数据平滑处理

特殊异常数据是针对当前需求中明确了的无效数据，即为了保证时间序列的完整性，如果某一个指标在某个时间点数据不存在，则该时间点的数据赋值为-1，针对这种无效数据，采用了时间序列的缺失补全相似的方法，首先统计连续的时间序列的值为-1的数量，然后计算平均值，如果从第一个开就是-1，那么就从第一个不无为-1的数据向前依次递减平均值作为前一个时间点的数据，否则依次向后递增平均值作为下一个时间点的数据。

### 周期（波长）的确定

周期的确定主要是为了对时间序列进行分解和预测天数的时候用到，所以必须在正式进行预测前把周期确定下来。WekaForecaster在数据建模的时候会自动计算周期并且开始预测，默认会对下一个时间点进行预测，如果使用整个时间序列导致耗时过高，所以确认周期的阶段只需要少量的数据即可，当前取时间序列的20%。

### 递增数据的判断

如果相邻的时间数据后一个总是大于前一个，毫无疑问数据是递增的，或者经过统计相邻时间的数据增长的数量大于相邻时间的数据减少的数量，也可以认为是递增的，但是可能会存在误判的情况，为了更好的判断是否递增数据，首先分别统计数据增长的数量和数据减少的数量，当数据数据增长的数量除以总量>减少的数量除以数据增长的数量，则把时间序列当做递增数据。

### 正常通道的确定

通常情况下，时间序列是正态分布（高斯分布）的。

首先，通过STL算法将时间序列分解为季节性、趋势性和残差，STL算法的必须提供时间序列的数据和周期。

其次，是对原始数据进行核密度概率分布来确定需要在季节性的基础上增加多少（或者减少）个标准差，从而确定数据的正常通道，最后判断时间序列数据是否是递增的，如果是递增数据则需要再加上趋势性数据。

### 表空间剩余使用率预测

时间序列总是存在或多或少的异常数据，为了提高预测的精准度，需要对时间序列进一步优化数据，把异常数据尽可能消除。

#### 数据平滑

使用正常通道的数据与原始数据进行比较，如果相应的时间点上原始数据大于正常通道值，则使用正常通道的值来修改原始数据的值。

#### 预测

将平滑数据和周期传给WekaForecaster完成数据的建模并开始预测，传入周期的目的是预测未来一个周期。将预测得到的数据不断累加并与剩余的表空间进行比较，如果预测得到的数据之和小于剩余表空间，将预测到的数据加入到平滑数据中，并再次调用WekaForecaster进行预测，如此重复向下预测，当所有预测的数据之和大于剩余表空间时，终止向下预测。

### 对象异常增长分析

#### 异常增长的判断

直接将原始数据和正常通道进行比较，在遍历的过程中，如果原始数据在某个点的值超出正常通道的值，则说明时间点的数据存在异常，取该索引作为主键，把该点（原始数据-正常通道）/正常通道作为异常级别存储到字段中。

要判断最后一个时间点是否异常，直接取Instances的最后一个索引到异常字典中查找，如果存在就异常值返回，否则返回0，0表示无异常。

#### 贡献度(权重)的计算

通常，一个数据只有一个指标，这个指标的贡献度则为1；对于由多个指标构成的数据集，有多种方法计算每个指标对这个数据的贡献度，比如基于平均值的贡献度计算，基于核概率密度的贡献度计算等。经过分析，一个序列中概率密度最高的值更能体现它对这列数据中的重要性，可以认为其他数据是基于该值的随机数。计算基于核概率密度的贡献度，首先求出每个指标的核概率密度分布，然后求出概率密度最高的值，然后求所有的指标的核概率密度最高的值的和，并将每个指标的核概率密度最高的值除以和得到值作为每个指标的贡献度。

#### 返回的异常数据的结构说明

返回的异常数据是一个JSONObject对象，其中每个字段的说明如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 备注 |
| name | String | 指标名称 |  |
| weight | Double | 指标的贡献度（权重） | 如果小于0，说明无效数据太多 |
| timestamp | String | 异常的时间点 |  |
| threshold | Double | 正常通道值 |  |
| errorLevel | Double | 异常级别 | 0表示正常 |
| increment | Double | 增量值 | 与前一个时间点差 |
| children | JSONObject | 子节点 |  |

具体示例：

|  |
| --- |
| {  "children": [],  "errorLevel": 0.0,  "increment": 0.0,  "name": "AUD$",  "threshold": 38797312.0,  "timestamp": "2017-11-02 15:00:00",  "weight": 0.3336787758602168  } |