中山大學
SUN YAT-SEN UNIVERSITY

数据挖掘导论大作业

题目 Title： 异常检测算法 （K-means，孤立森林）

院 系
School (Department)： 数据科学与计算机学院

专 业
Major： 软件工程（电子政务）

学生姓名
Student Name： 孙肖冉

学 号
Student No.： 16340198

时间： 2019 年 7 月 10 日

Date: July 10th, 2019

【摘 要】

结合‘Chatops’概念实现自动对软件系统进行智能运维，而异常检测作为智能运维的基础，检测算法应该实时准确。为了实现 Chatops，选择 Slack 聊天工具对接 github 提出聊天机器人 Hubot，并使用 ansible，Telegraf 与 InfluxDB 结合抓取、存储多台机器性能指标和日志信息，利用 K-means 算法、孤立森林算法对数据进行基于性能指标的异常检测，在本文中会重点介绍异常检测的算法。

【关键词】 智能运维，数据抓取，异常检测

一. 概述

1.1 背景

ChatOps 是一种新的操作范例——今天已经在后台发生的工作被带到一个公共聊天室中。诸如从聊天中部署代码、从 TSDB 或日志工具中查看图表……所有这些都是可以通过 ChatOps 完成的任务的示例。性能问题和可用性问题在大规模分布式系统中成为常态：数据中心、云、集群等。将 ChatOps 与某些算法相结合就能够很容易地通过聊天工具对集群进行监控，操作。实现智能运维的第一步就是异常检测/预测，然后能够对于异常点进行定位，定下告警线，进行预警，最后能够直接作出反应或根据执行命令进行重启或其他操作。

由此可以看出异常检测算法是实现智能化运维的重要部分。

1.2 项目简介

因为本文的重点将放在异常算法的部分，所以在此处将简短介绍一下项目。

项目的要求是：

前端：选择微信、Slack 或者其他聊天工具对接聊天机器人，支持图片结果

数据库：选择一款时序数据库如 InfluxDB 存放性能指标或者日志信息

后端：包含 2-3 种异常检测等智能分析算法

工具：

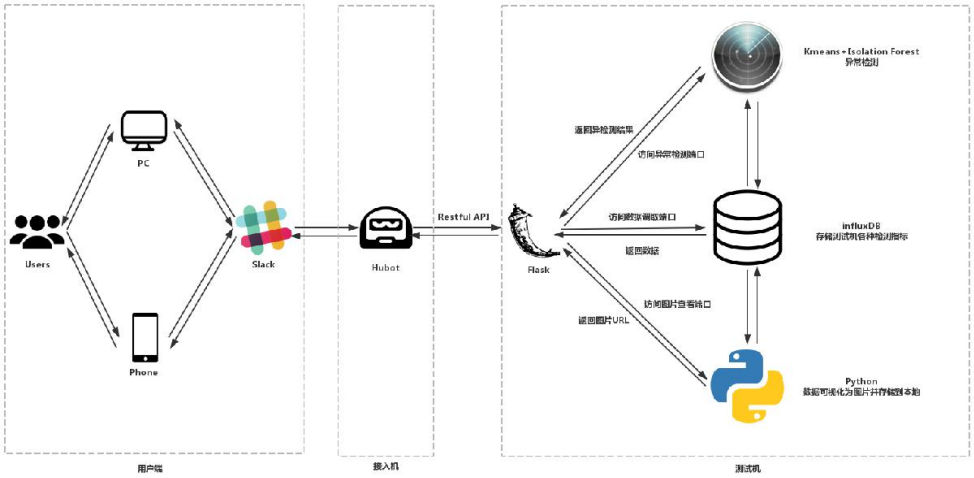
Hubot：Github 的开源聊天机器人

InfluxDB：是一个由 InfluxData 开发的开源时序型数据库。

Telegraf：Telegraf 是一个的代理程序，可收集系统和服务的统计数据，并写入到 InfluxDB 数据库。

Grafana：Grafana 是一个跨平台的开源的度量分析和可视化工具

具体实现的框架：



1.3 小组成员及分工

姓名	学号	分工
孙肖冉	16340198	孤立森林算法，数据抓取，测试
陈远洋	16340048	K-means 算法

感谢 苏依晴 同学的帮助

完整项目代码可在 https://github.com/vyychenyy/Cloud-Platform 查看

1.4 结构简介

本文第一章主要是阐述研究智能运维的背景和意义以及本文所要做的大致工作内容；第二章主要对算法进行详细介绍；第三章将会对算法进行测试与评估；第四章是对本文的总结以及改进方向；最后介绍了参考文献。

二．算法描述

2.1K-means 算法

2.2 孤立森林算法

2.2.1 介绍

孤立森林算法是一种离群点检测算法，它尝试直接去刻画数据的“疏离”程度，较为简单易用，但是该算法对于异常数据有两点要求：

1）异常数据跟样本中大多数数据不太一样

2）异常数据在整体数据样本中占比比较小

孤立森林算法是无监督的异常检测算法，在实际应用时需要注意的是：

1）如果训练样本中异常值的比例较高，违背了先前提到的异常检测的基本假设，可能最终结果会受影响

2）异常检测根具体的应用场景紧密相关，算法检测出的“异常”不一定是我们实际想要的。所以，在阈值设定时，要根据具体场景进行调整，以免检测出并不异常的“异常数据”

2.2.2 具体实现

基于孤立森林算法的异常检测包括两个步骤：训练阶段，基于训练集的子样本来建立孤立树；测试阶段，用孤立树为每一个测试样本计算异常分数。

1）训练阶段：

在训练阶段，iTree 的建立是通过对训练集的递归分割，直到数达到了指定高度。或所有的样本都被孤立，树的高度限制 l 与子样本数量 ψ 的关系为

l=ceiling(log2(ψ))，它近似等于树的平均高度。我们只关心路径长度较小的那些点，它们更有可能是异常点。

详细的训练过程如算法 1 和算法 2 所示。其中子样本大小 ψ 树的数量 t 的经验值：ψ=256，t=100

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - subsampling size
Output: a set of t *iTrees*

Initialize $Forest$
for $i = 1$ to t **do**
  $X' \leftarrow sample(X, \psi)$
  $Forest \leftarrow Forest \cup iTree(X')$
end for
return $Forest$

Algorithm 2 : $iTree(X')$

Inputs: X' - input data
Output: an *i*Tree

if X' cannot be divided **then**
  return $exNode\{Size \leftarrow |X'|\}$
else
  let Q be a list of attributes in X'
  randomly select an attribute $q \in Q$
  randomly select a split point p between the max and min values of attribute q in X'
  $X_l \leftarrow filter(X', q < p)$
  $X_r \leftarrow filter(X', q \geq p)$
  return $inNode\{Left \leftarrow iTree(X_l),$
    $Right \leftarrow iTree(X_r),$
    $SplitAtt \leftarrow q,$
    $SplitValue \leftarrow p\}$
end if

2）评估阶段：

在评估阶段，每一个测试样本的异常分数由期望路径长度 E(h(x))得到，E(h(x))是将样本通过孤立森林中的每一棵树得到的。具体过程见算法 3

Algorithm 3 : $PathLength(x, T, hlim, e)$

Inputs : x - an instance, T - an iTree, $hlim$ - height limit, e - current path length; to be initialized to zero when first called

Output: path length of x

1: **if** T is an external node or $e \geq hlim$ **then**
2: return $e + c(T.size)$ {$c(.)$ is defined in Equation 1}
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, hlim, e + 1)$
7: **else** {$x_a \geq T.splitValue$}
8: return $PathLength(x, T.right, hlim, e + 1)$
9: **end if**

在编码过程中发现 sklearn.ensemble 库中有直接可以调用的函 IsolationForest(),并且可以使用 decision_function（）进行异常评分，得分越低代表离群程度越大，是异常点的可能性越大。这样评定是否异常的阈值的设定就很重要了，在针对 net 性能指标时，我们观察到当阈值为-0.178 时，效果最好。

三．实验结果分析

3.1 数据集类型

训练/测试数据格式

XX.csv 文件（可直接从 InfluxDB 导出）

name	time	bytes_recv	bytes_sent
net	1.5599E+18	383997594	2847855197
net	1.5599E+18	384416224	2851082949
net	1.5599E+18	384823182	2854179151
net	1.5599E+18	385246948	2857406657
net	1.5599E+18	385686591	2860782724
net	1.5599E+18	386119529	2864076628
net	1.5599E+18	386558851	2867444112
net	1.5599E+18	386983239	2870650517
net	1.5599E+18	387481391	2874486372
net	1.5599E+18	387813616	2877031643
net	1.5599E+18	388243053	2880310646
net	1.5599E+18	388672911	2883601468
net	1.5599E+18	389105390	2886887371
net	1.5599E+18	389519162	2890045435

3.2 测试与评价

3.2.1.手动注入异常点

通过更改数据进行检测（多次更改数据取平均值）

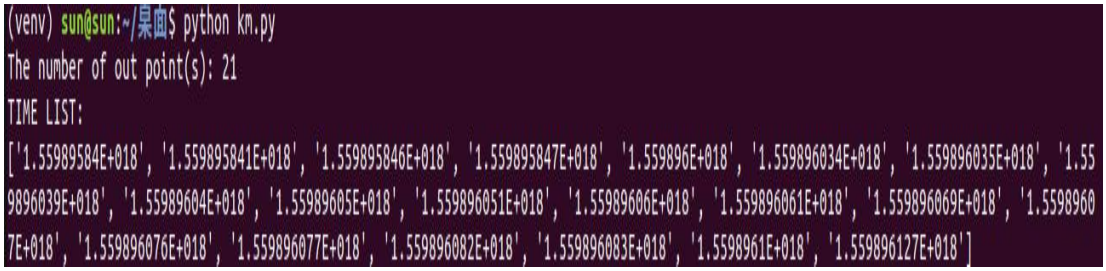
算法	预计输出	实际输出	正确率
KM 算法	2 异常点对应时间	2	100%
	12 异常点对应时间	8	67%
	22 异常点对应时间	19	86.3%
	52 异常点对应时间	51	98.0%
孤立森林算法	2 异常点对应时间	2	100%
	12 异常点对应时间	11.5	95.8%
	22 异常点对应时间	21	95.45%
	52 异常点对应时间	50	96.1%

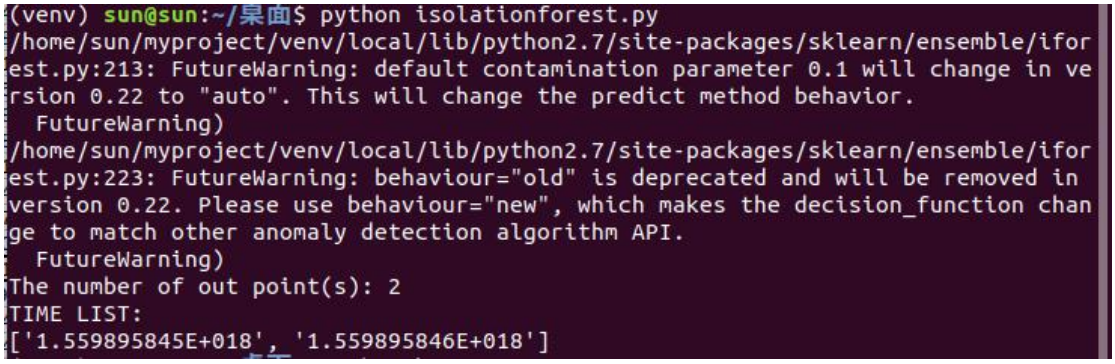
3.2.2 利用 TC 方式发送丢包或延时的数据 1 分钟进行注错（100%）

算法	预计输出	实际输出	正确率
KM 算法	60 异常点对应时间	60	100%
孤立森林算法	60 异常点对应时间	60	100%

当我们将丢包率改成 10%时，孤立森林算法并未检测到异常，KM 算法的正确率大概在 60%

3.2.3 部分测试结果截图：





由测试结果可以看出，当手工注错偏差值较明显时，两个算法的正确率都比较高，但是一旦偏差值不明显时，异常点就很难被检测到。

四．总结与展望

现实中有关异常点的检测往往需要多种指标的结合判断，例如当网络传输出现异常时，cpu 的利用率也是会出现明显的变化。

我主要负责的是孤立森林算法，设置阈值就花费了我不少精力。且孤立森林算法并不适合高维数据，由于每次切空间都是随机选取一个维度，建完树之后仍会有大量的维度信息没有被使用，导致算法可靠性降低。当异常出现的过于频繁

时，孤立森林算法的正确率将会大大降低。所以在利用该算法时应该要牢记 2.2.1 中的两个要求，一定要判断是否符合这两点。我们尝试调整了参数，例如树的个数，样本大小；发现原始值的效果最佳。

关于 K-means 算法，因为正确率较高，我们并未尝试调整参数的值，例如聚簇的个数，之后可以进行尝试。

五．参考文献

[1] Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation forest.” Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on. IEEE, 2008.