文章编号：1003-0077(2018)08-0103-08

类比社交网络的进程故障检测方法研究

程自强[1]，黄 荣[2]，杨 洋[1]

(1. 浙江大学 计算机学院，浙江 杭州 310058；
2. 上海海高通信股份有限公司，上海 201612)

摘 要：我们周围充满了各种网络；按照相似的内在机理，可以将它们分为物理网络和信息网络。对于具有明显物理特征的网络，我们可以运用物理常识解释其内部结构或节点的性质；而对于信息网络，我们往往需要结合一些先验知识去理解，社交网络正是这样一个例子。然而，对于那些并非具有显著物理或社交背景的网络，以往并没有明确的分析思路和方法。该文将尝试运用类似于分析社交网络的方法去分析电信 CSB 业务系统服务器集群上的进程网络；具体地预测进程网络中节点的崩溃(故障)状态。在这个特定的进程网络上，这种建模和分析思路得到了较为可信的结果；研究表明，进程节点的运行信息(如 CPU 和内存使用率)、进程间的通信情况以及进程节点在整个网络中的结构特征对于判断该节点的状态具有一定的指导价值，而上述特征在时间维度上的变化量同样反映了进程/端口的状态。

关键词：故障检测；社交网络；统计学习；二分类
中图分类号：TP391 **文献标识码**：A

Process-crash Detection by Analogy with Social Network

CHENG Ziqiang[1], HUANG Rong[2], YANG Yang[1]

(1. College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang 310058, China;
2. Shanghai Seahigh Telecom Corp., Shanghai 201612, China)

Abstract: Various networks surrounding us can be divided into physical networks and information networks according to similar internal mechanisms. For networks with obvious physical characteristics, we can use the basic physical knowledge to explain the nature of its internal structure or nodes; For information networks, we may need to combine some prior knowledge to understand, and social network is such an example. However, there're no clear ways or means for analyzing networks without significant physical or social backgrounds. In this paper, we explore a similar approach of social network analysis to analyze the process network on China Telecom CSB cluster; specifically, to predict the crashing of process on the cluster. Such approach has brought credible results on this particular dataset, and according to our research, the running information such as loads of CPU and memory, communications between processes and the structural features in the process network are valuable in predicting the states of processes and ports; furthermore, the changes of features mentioned above in the time dimension reflect the states of processes or ports.

Key words: fault detection; social network; statistical learning; binary classification.

0 引言

对于一般的信息或社交网络，我们已经得到很多它们的性质和建模方法。例如，六度分离理论(six degrees of separation)[1]，弱连接(weak ties)[2]，无标度(scale-free)[3-4]，以及 Small World[5]，Kronecker Graph[6]等。利用这些知识，结合真实社交网络的结构，我们可以对一个社交网络进行社区(Community)[7-8]划分，对网络中的边进行预测[9]以及对节

收稿日期：2017-09-29 **定稿日期**：2017-10-18
基金项目：国家自然科学基金(61702447)

点对之间的影响进行建模和量化分析[10]等。而对于一个缺少先验背景知识的网络,以往并没有明确的分析方法。

本文将从社交网络的角度,运用社交网络的相关性质和分析方法,去理解服务器集群上的进程网络,并对该网络上的节点即某个进程的状态进行预测。

我们首先对这样一个电信CSB业务系统服务器集群上的进程网络进行建模:

该进程网络可看作图$G(V,E,T)$。其中节点$v\in V$为单个进程,T为当前网络的时间戳,$e\in E$表示进程间的联系,即$e(i,j)$意味着进程i和j在时刻T存在通信(如socket通信),边e的权重w为当前时刻两个进程之间的通信次数(多端口通信)。

和社交网络不同的是,我们缺乏对该进程网络上节点的了解。例如,一个进程何时会与周围的进程产生通信,为什么会产生通信,及我们的预测目标:一个进程是否会发生崩溃。因为进程间的通信通常依赖于某个进程的具体功能和实际服务的使用情况。

即使如此,我们还是可以类比社交网络中节点的相关性质,对该进程网络中的节点做出如下假设:

1) 对应于社交网络中个人的性别、年龄等信息,我们可以将某个进程的占用CPU、内存情况看作进程节点的"固有属性";

2) 对应于社交网络中个人的社交关系,进程之间的通信可以看作进程网络中的边;那么,节点在网络中的中心度[11-12]可以衡量节点的活跃程度以及与外界联系的紧密程度;

3) 对应于社交网络中的个人行为如转发推文等,我们将进程的某一特定状态视作该进程做出的一个行为;具体地,我们把进程崩溃视作一个进程的行为,那么进程网络中进程崩溃这一现象可以类比为社交网络中消息的扩散[13]。

在社交网络中,我们倾向于认为"朋友的朋友更有可能成为自己的朋友"[14];对应地,在进程网络中,我们可以倾向于认为,一个进程的行为(状态)不仅会影响和它直接有通信的进程,还有可能影响它的"邻居的邻居"。

基于以上假设,我们把一个进程发生崩溃的现象定义为进程网络中一个节点的状态;在给定的时间戳下,网络中的节点可以被分为两类:发生崩溃的节点和没有发生崩溃的节点;因此,预测进程网络中进程的崩溃问题可以转化为针对网络节点的二分类问题[15]。我们仔细地选取节点的相关属性作为节点分类的特征,用SVM(Support Vector Machine)分类器[16]对该模型进行分类,并得到了较为可信的结果。

1 问题定义

我们对电信CSB业务系统服务器集群上的进程网络以及该网络中的节点、边和节点状态做出如下定义:

1.1 进程标识符:

我们用一个进程的如下信息作为其标识符:本地IP,本地主机名,本地进程组,进程描述以及进程号。换言之,上述五个字段可以确定一个唯一的进程。

如果两条日志记录中进程的标识符完全一致,我们认为这是同一进程的记录。

1.2 常驻进程:

我们把存在时长超过某一阈值ΔT的进程定义为常驻进程。一方面我们关心那些运行时间较长的主要进程的状态;另一方面服务器集群上存在临时启动的进程,它们的进程号不断变化,状态表现得极不稳定,干扰我们对进程状态的判断。

常驻进程i体现在日志记录中的形式为:存在进程i的两条记录,时间戳分别为$T_1,T_2(T_1<T_2)$,且$T_2-T_1\geqslant\Delta T$。

1.3 进程网络

该进程网络可以用图$G(V,E,T)$来表示,其中节点$v\in V$为某一具体进程,用1.1中的五个字段来描述,T为当前网络的时间戳,$e\in E$表示进程间的联系,即$e(i,j)$意味着进程i和j在时刻t存在通信,边e的权重w为通信次数。

1.4 进程崩溃

假设常驻进程i在进程日志中连续出现的时间戳分别为$T_1,T_2(T_1<T_2)$,且T_1时刻进程i的标识符为a_1,b_1,c_1,d_1,e_1,T_2时刻进程i的标识符为a_2,b_2,c_2,d_2,e_2。其中a,b,c,d,e分别对应本地IP,本地主机名,本地进程组,进程描述以及进程号。

我们把常驻进程i在时刻T_2为崩溃状态定义为:与T_1时刻相比,除进程号外,进程i的其余四

个标识符均未发生变化，即 $e_1 \neq e_2$。

换言之，一个常驻进程在时刻 T 为崩溃状态意味着该进程此刻的进程号与其上一次在日志中出现的进程号不一致。

1.5　远端进程

我们把在日志记录中本地 IP 或本地主机名为空的进程定义为远端进程。由于日志记录是由本地 probe 探针对正在运行的进程进行遍历得到的，因此日志记录中本地 IP 或主机名为空可以视作该进程不在这个服务器集群上，我们称这类进程为远端进程。

2　数据观察

在对进程状态进行预测前，我们先从整体上对数据做一些基本的分析。

2.1　数据量

本数据集（服务器集群的进程日志）由电信 CSB 业务系统服务器 CSB 节点上的 probe 探针定时探测得到，我们选取了 2016 年 8 月 30 日 14 时至 18 时共 2 858 063 条日志记录进行分析。其中，该时间段内的常驻进程共 973 个，进程崩溃次数为 25 次。在后面的二分类问题中，我们把在时刻 T 发生崩溃的进程定义为正样本，没有发生崩溃的进程定义为负样本。在此数据集上，正负样本比为 0.16‰，是极为稀疏的。

2.2　特征分布

对常驻进程 i 在时刻 T，我们定义以下特征：

1）CPU 占用率；

2）内存使用量；

3）与之存在通信的进程数量（即进程网络中节点的度）；

4）与其他进程的通信总量（即进程网络中节点的边权之和）；

5）存在通信的远端进程数量；

6）i 在进程网络 $G(,,T)$ 中的中心度，包括 betweenness[17] 和 closeness[18]。

因此，可以得到正负样本的特征整体分布：

从图 1～图 2 来看，正负样本在 CPU 和内存这两个“固有属性”上存在一定的差异：对于负样本即没有发生崩溃的进程，它们的 CPU 使用率集中在

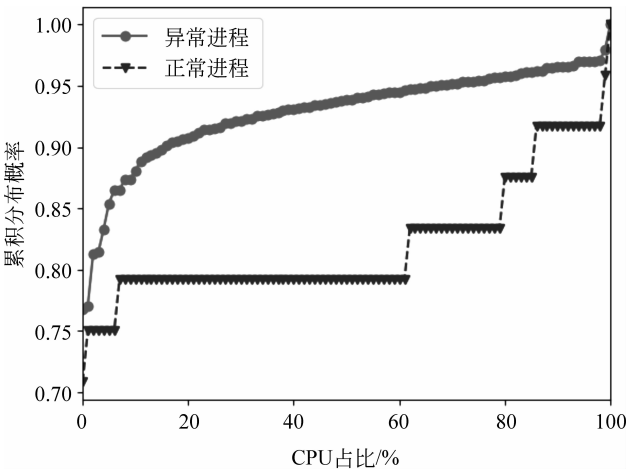


图 1　CPU 占用率的整体分布

其中横坐标为进程 CPU 占比（百分比），纵坐标为累积分布概率；圆形点线为正样本分布曲线，三角形点线为负样本分布曲线。

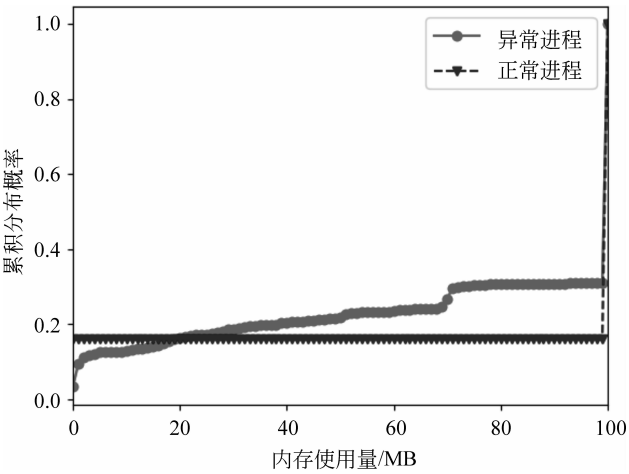


图 2　内存占用量的整体分布

其中横坐标为进程内存使用量（100MB），纵坐标为累积分布概率；圆形点线为正样本分布曲线，三角形点线为负样本分布曲线。

20％以下和 100％以上，内存使用量的分布较为分散；而正样本的 CPU 使用率分布较为分散，内存使用量却集中在 0 和 10GB 以上这两个区间。根据经验，这种分布是可以理解的。因为，一个较大的进程发生崩溃后刚刚启动时往往需要重新加载，内存使用量自然比较多；而对于正在运行的进程，如果是计算密集型的，CPU 使用量会较高。否则，一般不会占用太多的 CPU 资源。

图 3～图 5 为其他非结构特征的整体分布。从各图中可以看到，存在通信的进程数量这一特征在正负样本之间没有显著区别，即进程网络中节点的度大多为 1；与其他进程的通信总量也不具有太大的参考价值。尽管从分布来看，正样本的通信总量分布更为平均（即节点的边权之和范围较大），但由于正样本过于稀疏，实际上在负样本中边权之和落

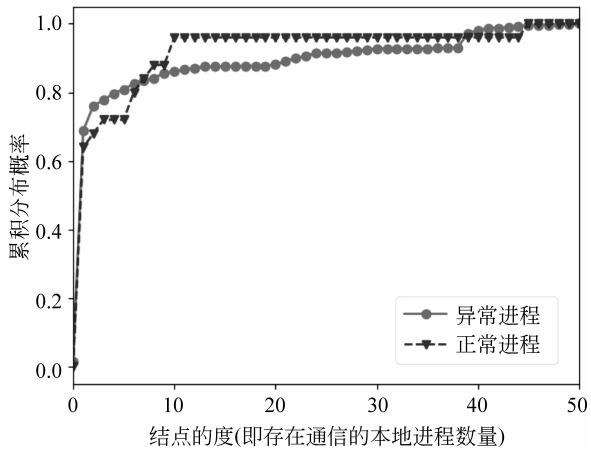


图3 存在通信的进程数量的整体分布

其中横坐标为存在通信的进程数量(即进程节点在进程网络中的度),纵坐标为累积分布概率;圆形点线为正样本分布曲线,三角形点线为负样本分布曲线。

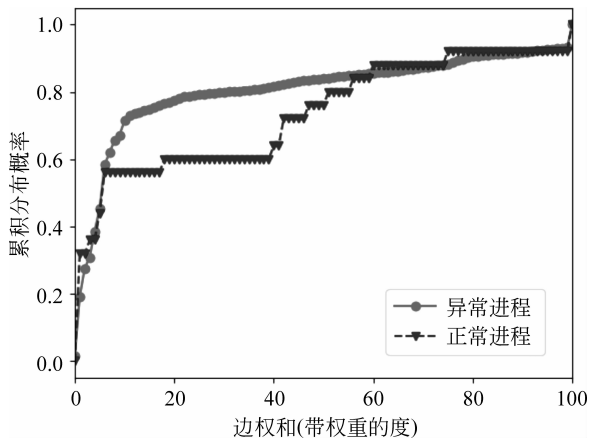


图4 与其他进程的通信总量的整体分布

其中横坐标为通信总量的值(即进程节点在进程网络中的边的权重之和),纵坐标为累积分布概率;圆形点线为正样本分布曲线,三角形点线为负样本分布曲线。

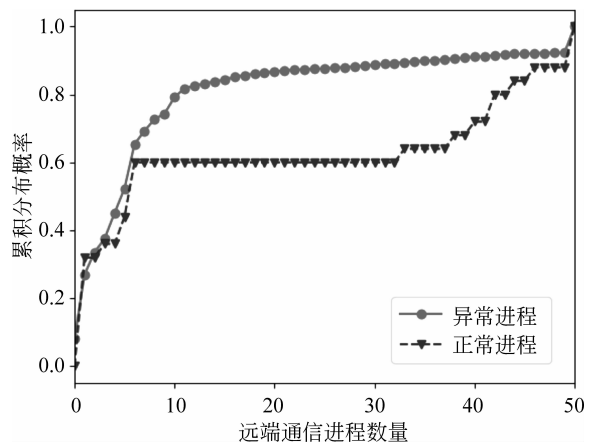


图5 与远端进程通信的整体分布

其中横坐标为存在通信的远端进程数量,纵坐标为累积分布概率;圆形点线为正样本分布曲线,三角形点线为负样本分布曲线。

在40至80之间的进程数量相对于正样本总量仍是十分巨大的;而存在通信的远端进程数量这一特征同样很难对正负样本做出区分。也就是说,正负样本在这三个非结构特征上没有明显差异。

我们再对该进程网络中的结构特征的分布情况进行考察。再次回到我们的目标,即我们希望能对进程状态做出判断:对于给定的进程,其是否为崩溃状态。对于较大规模的服务器集群,其上运行着大量进程。我们不难想象,如果一个进程关联越多的其他进程,那么该进程就越重要,其对服务器的负载就越重,崩溃的可能性就越大。

因此,我们选择进程网络图上的结构特征来衡量一个进程的重要性或者核心程度,我们希望通过进程节点的中心度[11-12]来帮助我们对进程状态做出分析。图6~7给出了进程节点的中心度的分布情况。

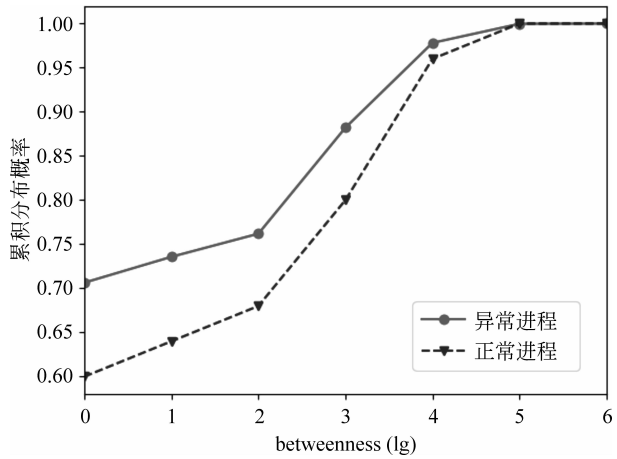


图6 betweeness的整体分布

其中横坐标为betweenness的十进对数(log10),纵坐标为累积分布概率;圆形点线为正样本分布曲线,三角形点线为负样本分布曲线。

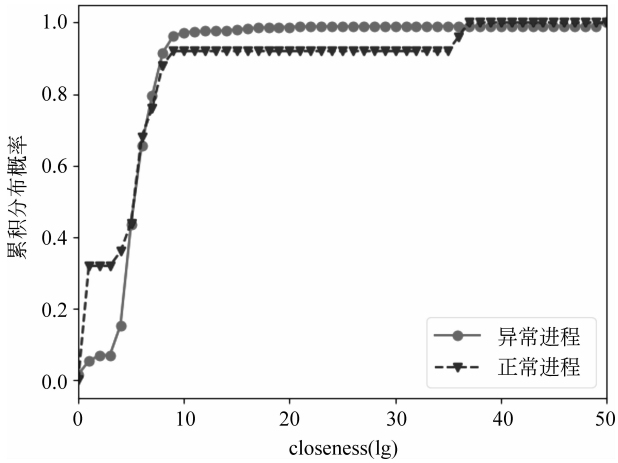


图7 closeness的整体分布

其中横坐标为closeness的值(log10),纵坐标为分布概率;纵坐标为累积分布概率;圆形点线为正样本分布曲线,三角形点线为负样本分布曲线。

遗憾的是，正负样本的中心度并没有显著的区别：正负样本的 betweenness 的整体分布几乎一致。而对于 closeness，尽管正样本看似在 30 至 40 上的某个区间有异常的分布，但注意到负样本中也有近百分之二的进程的 closeness 大于 50。我们更倾向于认为这些都是 closeness 较大的进程，至于 closeness 的值究竟是 30 还是 50，这是不重要的。

事实上，这并不与“中心度高的重要进程更容易崩溃”这一假设矛盾。往往一个中心进程崩溃会导致与其存在通信的其他进程或者该进程的子进程产生异常，而日志记录是在某个特定的时间点通过 probe 探针生成。因此，很有可能在生成日志记录时，以该中心进程为核心的进程组都进行了重启，故而正样本的中心度分布和样本的整体分布没有较大区别。

值得注意的是，betweeness 和 closeness 具有一定的相关性。尽管正负样本关于中心度特征的分布相似，但二者作为特征训练分类器的效果也许会有意想不到的效果，在实验中我们也将看到这一点。

2.3　时间间隔分布

我们对进程的静态特征做了整体分析，但是我们还应当注意到，进程的崩溃是一个过程，时间维度上的特征也许会较好地反映进程的状态。

我们考虑进程从正常状态到崩溃状态的时间间隔，体现在日志记录中即为同一进程的进程号不同的两条连续日志记录的时间戳的差。如果进程一直保持着正常状态（没有崩溃），我们倾向于认为其在日志记录中出现的时间戳应当是比较稳定的，即不会突然在一段时间内没有日志记录。而对于发生崩溃的进程，由于其重启等因素，可能会有较长时间间隔没有日志记录的现象。

具体地，我们定义进程 i 的一个时间间隔 $\Delta T=T_2-T_1$，其中 T_1，T_2 为 i 的连续两条日志记录的时间戳（$T_2>T_1$）；定义 $\Delta T=T_2-T_1$ 为正样本的时间间隔，如果恰好在 T_2 时刻进程号发生变化。

图 8 为正负样本的时间间隔的整体分布。再则，我们发现在时间间隔这一特征上没有显著区别。由于 probe 探针每三分钟采样一次，因此时间戳的差集中在 3、6 或 9 等数值上。

除了时间间隔外，对于进程 i 的两条日志记录 T_1，T_2，我们在分类问题中还会考虑 T_1，T_2 上的特征差，即进程的特征随时间的变化量。后面将看到，这一考虑是十分有效的。

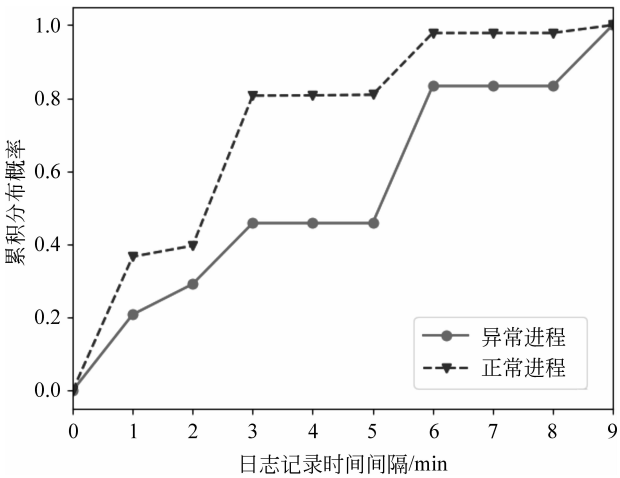


图 8　时间间隔的整体分布

其中横坐标为时间间隔（分钟），纵坐标为累积分布概率；圆形点线为正样本分布曲线，三角形点线为负样本分布曲线。

3　实验方法

我们将用分类问题的思路判断给定进程的状态。由于进程状态只有正常和崩溃两种，因此目标简化为二分类[15]问题。

问题：给定一个进程的相关描述，输出该进程所处的状态（崩溃与否）。

3.1　训练方法

我们人工提取 2.2 中提及的关于进程的非结构化和结构化特征作为输入，通过训练集得到一个 SVM 分类器，对测试集中的每个进程输出判断结果。

其中，由于正负样本比过小（正样本过少），我们采用过采样（over-sampling）[19]的方法生成训练集，用交叉验证[20-21]的方式对分类器进行训练，然后对正样本极少的原始数据集进行测试。

3.2　标签的提取

我们按照 1.4 给出的方式，人工从原始数据集中提取进程标签（正负样本）。即首先过滤日志得到常驻进程，对每个常驻进程，判断其上一条时间戳的日志记录的进程号是否与当前时间戳的进程号一致；若不一致，则认为在当前时间戳进程发生崩溃，采集为正样本，否则为负样本。

3.3　特征选取

我们将 2.2 中提及的非结构化和结构化特征组合起来作为一个进程的基本特征向量。除此之外，

我们还增加考虑特征随时间的变化量。

给定进程 i，时间戳 T_0 和其截至 T_0 时刻所出现的日志记录的时间戳序列 T_i，$i \geqslant 0$，满足

$$T_{k+1} < T_k, k \geqslant 0,$$

我们定义进程 i 在 T_0 时刻的 k 阶特征变化量

$$\Delta f_k = g\{f(T_0) - f(T_k), T_0 - T_k\},$$

其中 $f(T)$ 为 T 时刻进程 i 的基本特征向量，$g(v,t)$ 为 time_decay[22] 函数，定义为：

$$g(v,t) = v * t * e^{1-t},$$

即对向量 v 乘时间间隔系数 $t * e^{1-t}$。

我们将上述 k 阶特征变化量（k 为参数）和基本特征向量拼接起来，得到一个进程的完整特征，作为样本数据的输入。

4 实验结果

4.1 实验设置

本数据集（电信CSB业务系统服务器集群的进程日志）时间跨度为2016年8月30日14时至18时，共2 858 063条日志记录进行分析。其中，常驻进程973个，进程崩溃次数为25次，正负样本比为0.16‰。

进程状态的判断是一个二分类问题，因此选取precision，recall和F1得分作为评价标准。由于正负样本比过低，实验过程中，首先对负样本进行随机采样，使得正负样本比分别为0.1和0.05。然后采用过采样和交叉验证的方式进行训练，每次训练和测试过程重复10次，结果取平均值。

4.2 实验结果

首先考虑具体的某个基本特征对分类效果的影响：

表1给出了不同特征对分类结果的影响的比较。可以看到，正负样本比对实验结果的影响是明显的：正负样本比越低，训练得到的分类器进行崩溃检测的效果越差；此外，横向地与没有剔除任何特征的分类器相比，我们可以发现，CPU和内存这两个特征对区分正负样本的作用是显著的，剔除其中任何一个都会使得F1-score的值有明显的下降。这个结果和特征的分布具有一致性，因为正负样本的CPU占用率和内存使用量的分布有着明显的区别；而对于剔除节点的度、边权和以及图的结构化特征betweeness和closeness，我们可以发现随着正负样本比的降低，F1-score反而在增加。

表1 单个特征对分类结果的影响

剔除特征	正负样本比	precision	recall	F1
无	0.1	0.112	0.672	0.142
	0.05	0.078	0.524	0.084
CPU	0.1	0.081	0.568	0.121
	0.05	0.056	0.488	0.077
内存	0.1	0.133	0.616	0.139
	0.05	0.048	0.524	0.074
节点的度	0.1	0.118	0.612	0.143
	0.095	0.051	0.476	0.074
边权和	0.1	0.106	0.58	0.132
	0.05	0.113	0.528	**0.092**
远端进程数量	0.1	0.133	0.488	0.118
	0.05	0.183	0.336	**0.111**
Betweeness	0.1	0.081	0.788	0.146
	0.05	0.053	0.772	0.089
Closeness	0.1	0.114	0.644	0.136
	0.05	0.048	0.632	0.081

为了考虑不同特征之间的相关性的影响，我们把上述的特征分为三类：1）进程的运行信息，即CPU和内存使用情况；2）进程的通信情况，体现为进程网络中节点的度（或带权重的边权和）以及存在通信的远端进程数量；3）进程网络中节点的结构特征，即betweenness和closeness。

图9、图10展示了不同特征对分类结果的影响。可以看到，第二类特征极大地提高了分类器的recall值，但precision值很低。这是因为有大量的负样本在第二类特征上和正样本具有相同的值，在只有第二类特征的条件下，分类器倾向于认为大部分样本都是正样本。因而崩溃预测在没有第二类特征的分类器上达到了最好的效果。

而第一类和第三类特征都可以在一定程度上反映进程的状态。这正如我们在特征分布中看到的那样，正负样本的CPU和内存使用情况的分布不同。而第三类特征即进程节点的中心度尽管分布相似，但二者具有紧密的相关性，结合在一起考虑便可以作为崩溃检测的指标之一。

图11给出了特征变化量对分类结果的影响的比较，其中 $k=i$ 表示我们将 $1,2,\cdots,i$ 阶特征变化

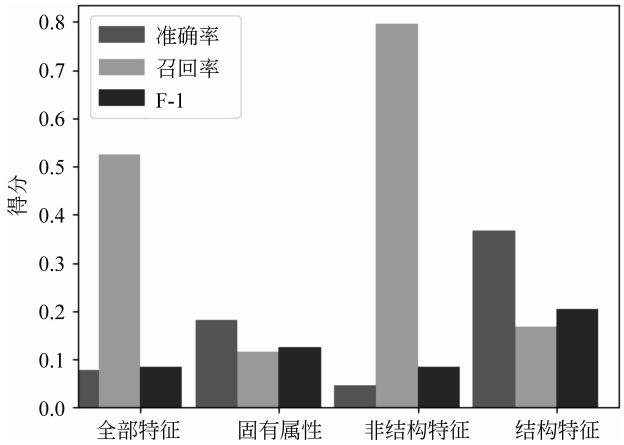


图 9　不同特征对分类的影响(1)

其中横坐标表示选取哪些特征。

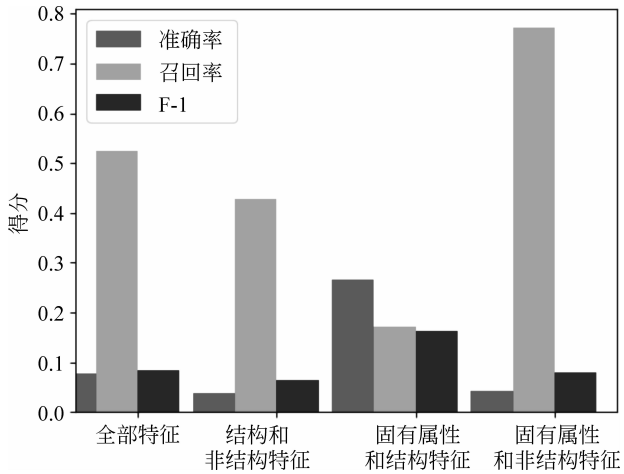


图 10　不同特征对分类的影响(2)

其中横坐标表示选取哪些特征。

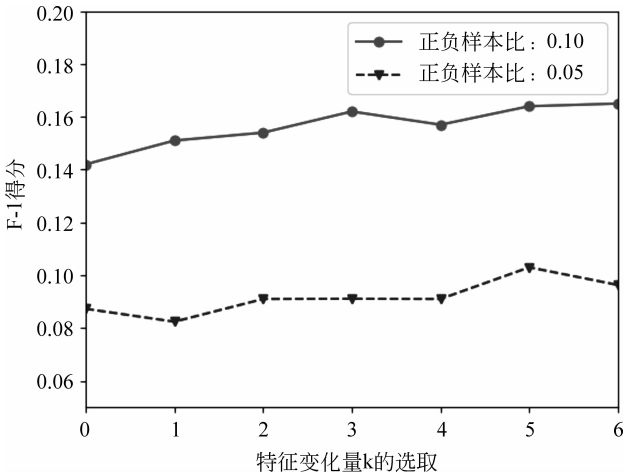


图 11　不同特征变化量的 F1-score

其中横坐标为考虑特征变化量的阶(0 表示没有考虑特征变化量)，纵坐标为 F-1 得分。

量均拼接加入进程的特征中。我们可以明显地看到，特征变化量对分类结果有正相关的影响，即考虑特征变化量越充分(k 越大)，分类结果越好。

这说明进程的崩溃不是突然的，一个进程在发生崩溃的前后，其 CPU、内存使用情况以及和其他进程的通信等属性往往会有突出的变化：比如在 PC 上，往往一个进程占用内存过大会出现崩溃，崩溃前内存使用量增加的趋势则反映了其崩溃的可能性。

从社交网络的角度来看，进程网络也是在不断变化的，网络节点也就是某个进程在时间这一维度的变化带有着丰富的信息。实验结果也表明，我们将时间信息加入进程的特征进行训练，得到了效果更好的分类器以对进程的崩溃进行检测。

5　总结

本文从社交网络的角度去分析由进程节点构成的网络，并选取节点的特征进行训练以对节点故障(崩溃)进行预测。

通过选取不同的特征(如进程运行信息、通信情况、进程节点的结构化特征以及时间维度的特征等)对进程崩溃进行预测，我们可以得到如下结论：

1) 由于数据集上正样本的稀疏性，训练数据的正负样本比对训练结果有显著的影响。

2) 相比于进程的通信情况，进程节点的结构特征以及进程运行信息(如 CPU 占用率，内存使用量等)对于判断一个进程是否会崩溃更具参考价值。

3) 从时间的维度看，进程的运行和通信信息的变化量更能反映该进程的状态。

6　未来的工作

针对电信 CSB 业务系统进程故障检测这一问题，我们还可以从以下两方面着手考虑：

1) 类比社交网络中的信息扩散[12]，我们可以把进程崩溃看作进程的一种行为。通过对进程节点之间的影响力进行建模，可以预测进程的崩溃情况。

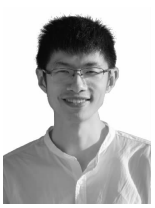
2) 除了人工地选取进程特征进行训练，我们还可以运用 graph embedding[23] 的方法对进程网络进行建模，用 embedding 的结果作为进程特征训练分类器。

参考文献

[1] Newman Mark, Albert-László Barabási, Duncan J. Watts. The Structure and Dynamics of Networks[M]. Princeton, NJ: Princeton University Press, 2006.

[2] Granovetter M. The strength of weak ties[J]. American Journal of Sociology, 1973, 78(6), 1360-1380.
[3] Onnela J P, Saramaki J, Hyvonen J, et al. Structure and tie strengths in mobile communication networks [J]. Proceedings of the National Academy of Sciences, 2007, 104 (18): 7332-7336.
[4] Choromański K, Matuszak M, Mie¸Kisz J. Scale-free graph with preferential attachment and evolving internal vertex structure[J]. Journal of Statistical Physics, 2013, 151 (6): 1175-1183.
[5] Milgram S. The Small World Problem[N]. Psychology Today. Ziff-Davis Publishing Company, 1967(5).
[6] Leskovec Jure, Chakrabarti Deepayan, Kleinberg Jon, et al. Kronecker graphs: an approach to modeling networks[J]. Journal of Machine Learning Research, 2010, 11: 985-1042.
[7] Fortunato S. Community detection in graphs [J]. Physics Reports, 2010, 486 (3-5): 75-174.
[8] Malliaros F D, Vazirgiannis M. Clustering and community detection in directed networks: a survey[J]. Physics Reports, 2013, 533 (4): 95-142.
[9] Liben-Nowell D, Kleinberg J. The Link Prediction Problem for Social Networks[C]//Proceedings of the 12th annual ACM International Conference on Information and Knowledge Management (CIKM'03), 2003: 556-559.
[10] Gomez-Rodriguez M, Leskovec J, Krause A. Inferring networks of diffusion and influence[C]//Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2010: 1019-1028.
[11] Freeman L. A set of measures of centrality based on betweenness[J]. Sociometry, 1997, 40: 35-41.
[12] Newman M E J. Networks: An Introduction. Oxford [M]. UK: Oxford University Press, 2010.
[13] Rodriguez M G, Balduzzi D, Schölkopf B. Uncovering the Temporal Dynamics of Diffusion Networks [C]//the 28th International Conference on Machine Learning (ICML), 2011: 561-568.
[14] Kadushin C. Understanding social networks: Theories, concepts, and findings[M]. Oxford: Oxford University Press, 2012.
[15] John Shawe-Taylor J S, Nello Cristianini. Kernel Methods for Pattern Analysis[M]. Cambridge: Cambridge University Press, 2004.
[16] Cortes C. Support-vector networks [J]. Machine Learning, 1995, 20(3): 273-297.
[17] Brandes, Ulrik. A faster algorithm for betweenness centrality[J]. Journal of Mathematical Sociology, 2001, 25 (2): 163-177.
[18] Sabidussi G. The centrality index of a graph[J]. Psychometrika, 1966, 31: 581-603.
[19] Japkowicz N. The class imbalance problem: significance and strategies[C]//Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000), Special Track on Inductive Learning, 2000.
[20] Geisser, Seymour. Predictive Inference [M]. New York, NY: Chapman and Hall, 1993.
[21] Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection[C]//Proceedings of the 14th International Joint Conference on Artificial Intelligence, San Mateo, CA: Morgan Kaufmann, 1995, 2(12): 1137-1143.
[22] Leike A. Demonstration of the exponential decay law using beer froth[J]. European Journal of Physics, 2001, 23: 21.
[23] Mohar, Bojan. A linear time algorithm for embedding graphs in an arbitrary surface[J]. SIAM Journal on Discrete Mathematics, 1999, 12 (1): 6-26

程自强(1996—),学士,主要研究领域为社交网络、数据挖掘。
Email: petecheng@zju.edu.cn



杨洋(1988—),博士,助理教授,主要研究领域为社交网络挖掘。
Email: yangya@zju.edu.cn

黄荣(1968—),研究员,主要研究领域为大数据技术及智能算法。
Email: rhuang@seahigh.com