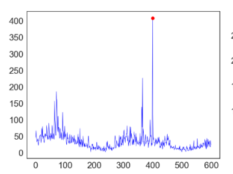
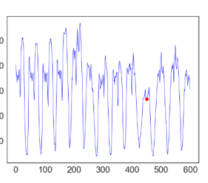
# AIOps方法论

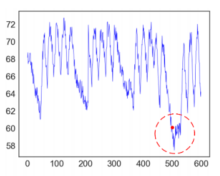
1. 异常检测
   1. 基于时间序列的异常检测
      1. 异常类型
         1. 空间异常，明显偏离历史正常行为的值



* + - 1. 上下文异常，整体上本身取值在正常范围内，但相对上下文表现为异常



* + - 1. 概念漂移，该观测点之后数据的统计特征发生了变化或漂移



* + 1. 算法框架
       1. 基于概率密度估计与距离度量的方法，TEDA、RDE、MRE
          1. 滑动窗口设定，用于处理连续时间序列

滑动窗口分为主滑动窗口和子滑动窗口，

主滑动窗口包含一定时间步，子滑动窗口将主滑动窗口划分为固定数量的时间序列，

主滑动窗口每滑动一次，就将当前观测值 添加到中并从中删除最久远的观测值

主窗口中的观测值数量记为N,子窗口的数量记为m,每个子窗口的观测值数量记为n

* + - * 1. 基于概率密度的描述子，用于将给定窗口内的序列处理为概率密度函数

概率密度描述子为一个p维向量，其中的每个值都是概率密度函数的值

利用KDE估计每个子窗口的概率密度函数

确定目标点序列:y\_i=W\_{min}+\frac{1}{p}(W\_{max}-W\_{min})(j-\frac{1}{2}),j=1,...,p

确定带宽：h=(\frac{4}{3N})^{\frac{1}{5}}\sigma,其中\sigma为当前主窗口标准差。

计算目标点序列的密度值：\hat{f}(y\_i|k)=\frac{1}{nh}\sum\_{i=N-nk+1}^{N-kn+n}K(\frac{y\_i-x\_i}{h}),K(x)=\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2},j=1,...,p;k=1,...,m,其中x\_i为窗口内的取值。

利用k维树及其最近邻搜索算法计算上述密度值

构造描述子序列(\hat{f}(y|1),...,\hat{f}(y|k),y=(y\_1,...,y\_p)

* + - * 1. 基于距离的异常检测规则

使用L1距离计算相邻描述子之间的距离，构造为一个距离序列D=\{d\_i,i=1,...,m\}；

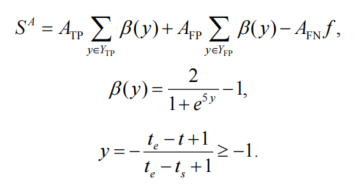
定义异常度量指标，包括平均距离或扩展最大距离

平均距离是D的均值D\_{avg}

扩展最大距离\widetilde{D}\_{max}=maxD+minC,其中C为距离序列D的距离序列

基于异常度量指标确定，当前观测值是否为异常值：l\_t=\begin{cases}1,& if s\_{t-1}=0 \& s\_t=1\\ 0,& otherwise\end{cases},其中s\_t=\begin{cases}1,&if d\_1>\widetilde{D}\_{max}\&d\_2\leq D\_{avg}\\0,otherwise\end{cases},其中d\_1表示当前观测值的的描述子与前序描述子的距离；

* + - 1. 基于时间序列建模的方法
         1. ARIMA，ARMA等
         2. 随机过程(？)
      2. 基于贝叶斯法则的参数变化检验
         1. 贝叶斯在线突变点检验
      3. RNNs，LSTM，分层时序记忆网络HTM等
    1. 评价指标
       1. 一个异常区间检测得分为S^A,其中A表示不同报警状况的权重，[t\_s,t\_e]为异常区间。



* + - 1. 算法的最终得分S\_{NAB}^A=100\cdot\frac{S^A}{rA\_{TP}},其中r为异常区间的个数

1. 根因分析
   1. 问题
      1. 关联系统故障传播迅速
         1. 由于系统间的复杂联系，一旦系统中的某个组件产生异常行为或发生故障，就会导致其关联组件也发生故障，这种异常行为会在系统内迅速传播开来，从而触发大量紧急、密集的并发告警
      2. 分组运维沟通成本高
         1. 大型 IT 系统的运行需要多个不同的运维小组共同维护，相关小组收到的告警信息通常局限于告警对象本身，小组之间往往需要不断的、反复的沟通，然后各自对可能的故障组件逐一排查，此外，系统故障通常是断断续续的、不确定的且不断变化的，很难以高保真度在事后进行场景重现和分析，难以总结出可重复利用的经验知识。
      3. 系统日志提供的告警信息可能是模糊的、不一致的和不完整的，例如告警存在误报和漏报。
      4. 系统短时间内可能出现多个相关或不相关的根源故障
   2. 异常传播类型
      1. 资源冲突
         1. 不同的应用程序之间共享着相同的设施资源导致资源的争用。当故障发生时，基础设施和应用程序之间可能会产生性能干扰。
      2. 服务调用传播
         1. 当某一应用程序出现异常时，同一服务调用链中的其他调用组件可能因此无法正常运行，异常沿着服务调用链传播。
   3. 分析思路
      1. 在捕获系统异常之间的可能传播路径从而推断根因
   4. 算法
      1. 基于故障传播图的排序算法
         1. 构造故障传播图
            1. 基于配置管理数据库构造资源配置图
            2. 基于PreciseTracer技术构建服务调用图
            3. 图中包含应用、设备两类节点，调用和部署两种关系
         2. 临近节点相似度评估
            1. 指标选择
            2. 基于指标数据相似度计算
         3. 基于节点相似度构造转移概率矩阵
         4. 基于随机游走等方法对节点进行排序
            1. 基本假设：某节点的统计停留概率越大，它是根因的概率越大
      2. 基于实现行为异常关系TBAC，基于相关性的排序
      3. 基于距离排名的DBR，最佳传播子图求解，等级由候选节点到其他节点的最小总距离确定
   5. 评估指标
      1. MAP：表示算法排名前 K 的输出结果的平均精度
      2. AFP: 后续依据算法的输出结果逐个检查节点直至找到真实根因所需的平均时间成本
      3. RT: 指算法从获取告警列表到输出根因列表的的总体运行时间