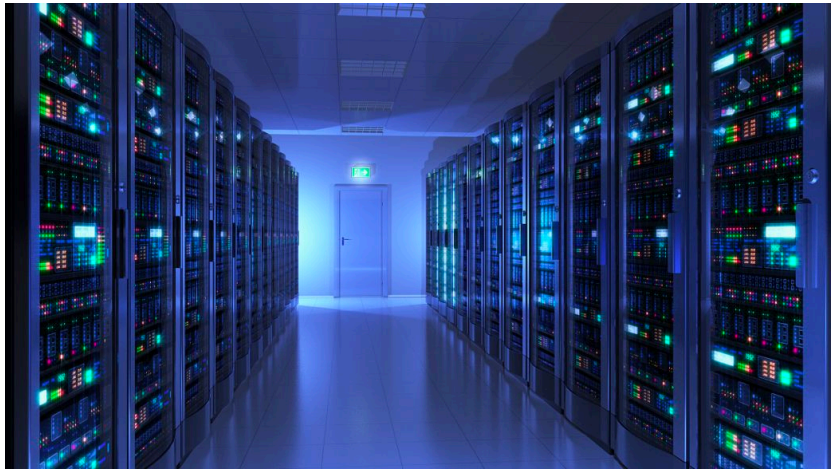
Practical and White-Box Anomaly Detection through Unsupervised and Active Learning

Yao Wang wangyao18@mails.tsinghua.edu.cn

Background

- Enterprises rely on complex systems
  - Systems are made up by many hardware and software
  - There exists many component dependencies and call relationships
  - The required operation and maintenance manpower grows with the expansion of scale
- Failure occurs from time to time
  - Bad impact on business, resulting in loss of users
  - Faults may spread between components
  - The loss increases as the fault continues



System failure affects business

On March 3, 2019, some ECS servers in the Alibaba Cloud North China region appeared IO HANG, and a large number of Internet company websites/Apps were paralyzed.

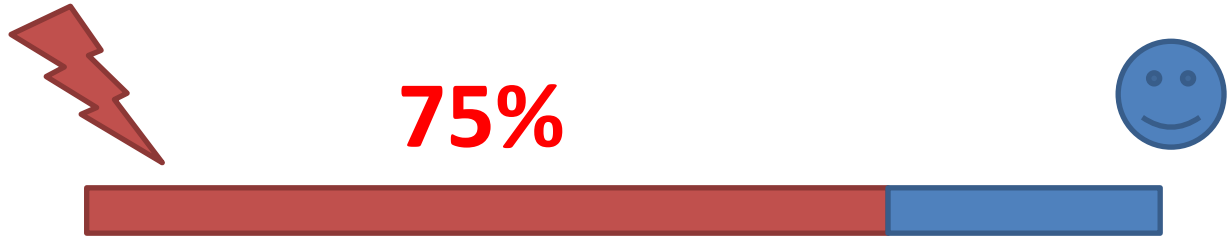
On the evening of April 23, 2019, the Ctrip App was down, and many services such as hotel, ticket search, and reservations were down for more than 2 hours.

On October 22, 2019, the GitHub database failed, some data was inconsistent with the latest version, which lasted more than 24 hours.

The key to finding faults: anomaly detection

75% of fault recovery time is used for fault detection[1]

Timely/real-time anomaly detection can improve the timeliness of fault detection, which is the key to troubleshooting and ensuring stable operation of the system.



[1] da Rocha Fonseca, Paulo Cesar, and Edjard Souza Mota. "A survey on fault management in software-defined networks." *IEEE Communications Surveys & Tutorials* 19.4 (2017): 2284-2321.

Key Performance Indicator

- **Data source**
  - Network (Traffic, Bandwidth)
  - Business (PV, Response Time)
  - Metric (CPU usage, Mem usage)
  - Application (Database, MiddleWare, JVM, .etc)
  - ……

KPI
Key Performance Indicator

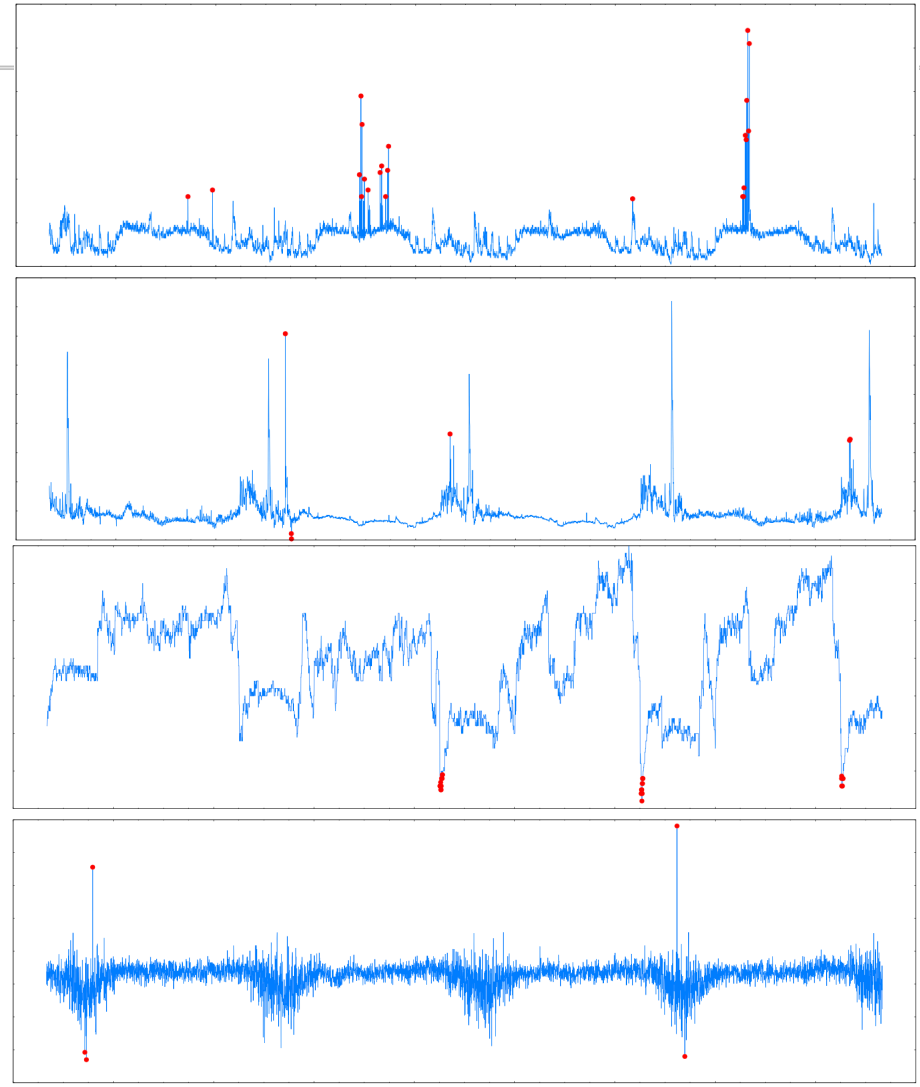
- **Data Format**
  - Time Series Data

Data
KPI-A,Timestamp1,ValueA1
KPI-A,Timestamp2,ValueA2
KPI-B,Timestamp1,ValueB1
KPI-B,Timestamp2,ValueB2
……

Definition

- **Anomaly**
  - Violation of the obvious law of historical data over the same period
  - Large spike in a short time
  - Strend change, and beyond a certain range compared with historical data

- **Detection: Identify the above anomalies through a specific method**

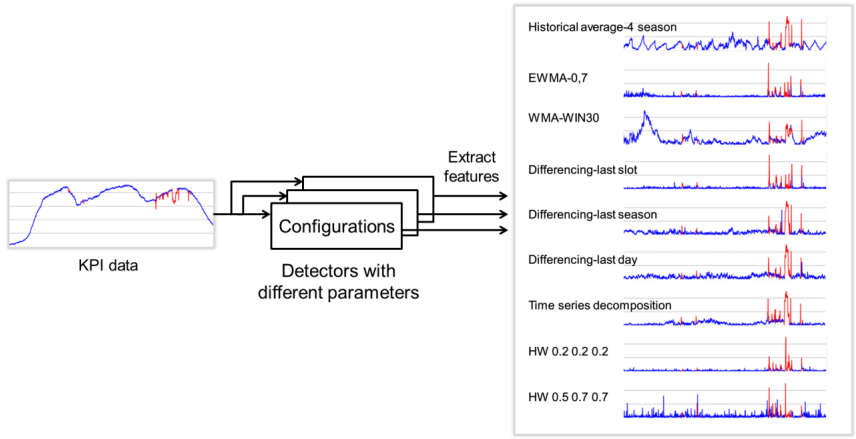


Challenges

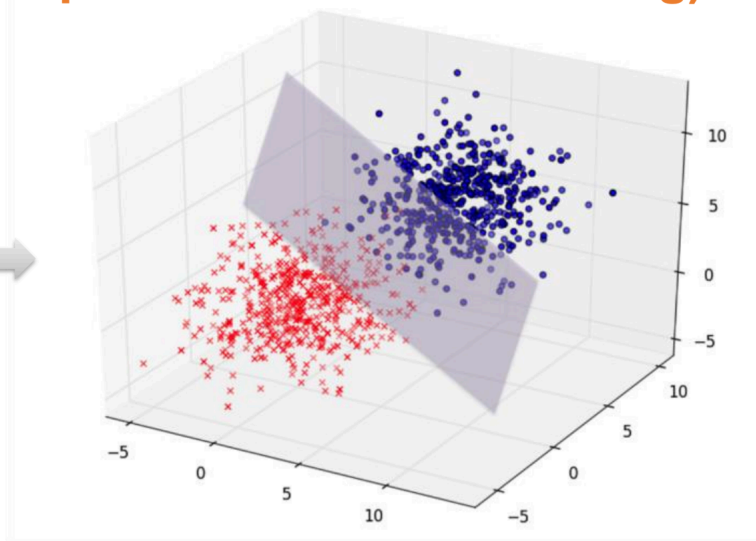
- KPI varies: Periodicity, glitches, missing data……
- Avoid the reappearance of false positive
- Limited resources
- Amount of KPI is rapidly growing
- Lack of labels

[ACM IMC'15] Opprentice[2]

Supervised learning, relying on labels, using Random Forest to select suitable features from a large number of detectors.



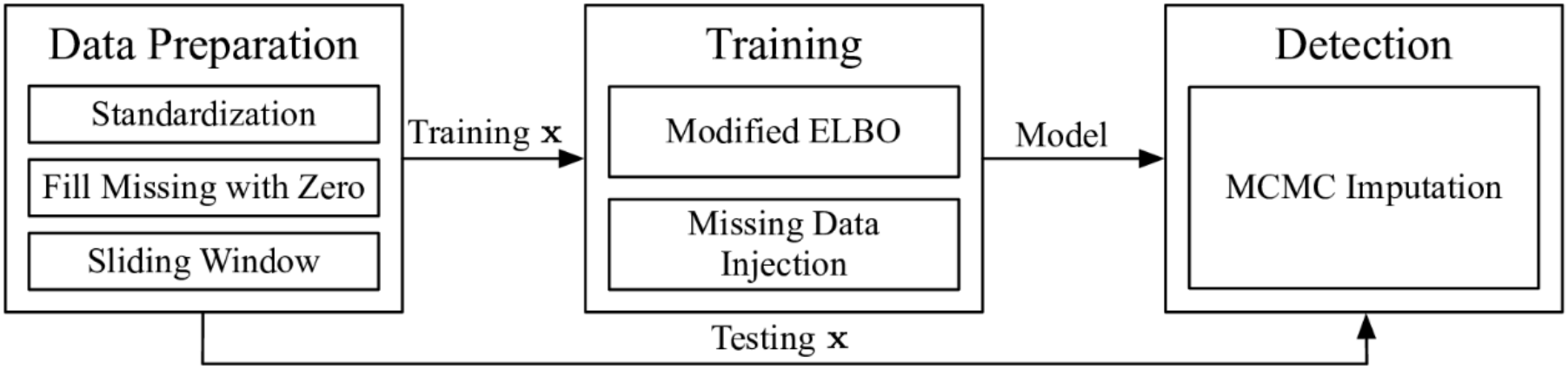
Classification in the feature space (Supervised machine learning)



[2] Dapeng Liu, Youjian Zhao, Haowen Xu, Opprentice: Towards Practical and Automatic Anomaly Detection Through Machine Learning, **ACM IMC 2015**, Tokyo, Japan, Oct 2015.

[WWW'18] Donut[3]

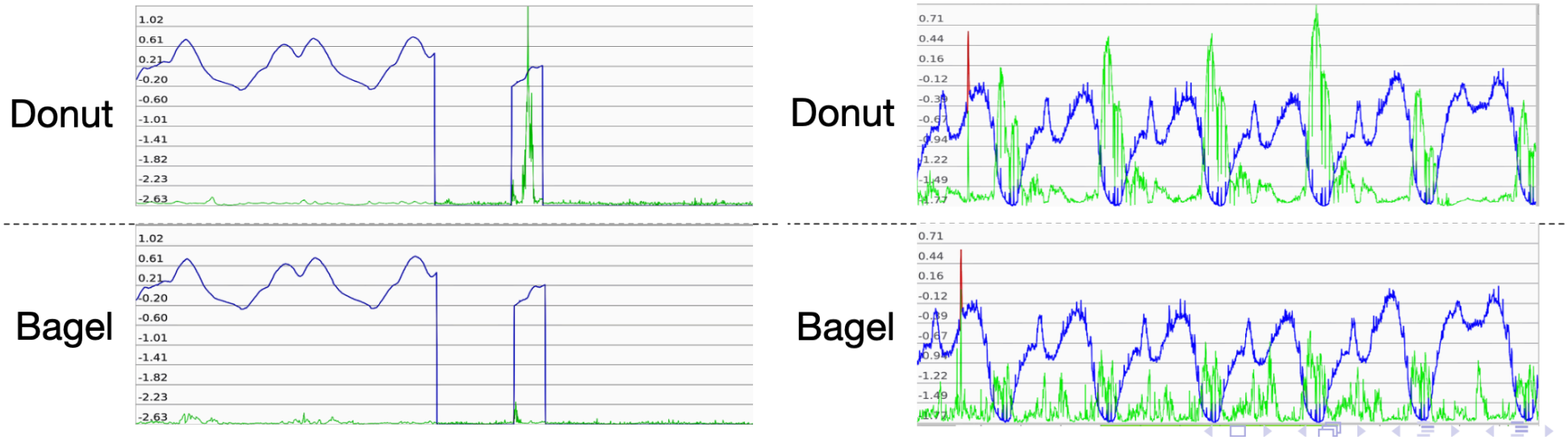
Unsupervised learning, using variational autoencoder (VAE) to build deep generative models, can obtain very accurate detection results on periodic KPI data.



[3] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications, **WWW 2018**, Lyon, France, 23-27 April 2018

[IPCCC'18] Bagel[4]

Improve some defects of Donut, add time information to reconstruct the normal mode of KPI, add random inactivation layer to avoid overfitting, and thus improve the detection effect of strong time-related anomalies.

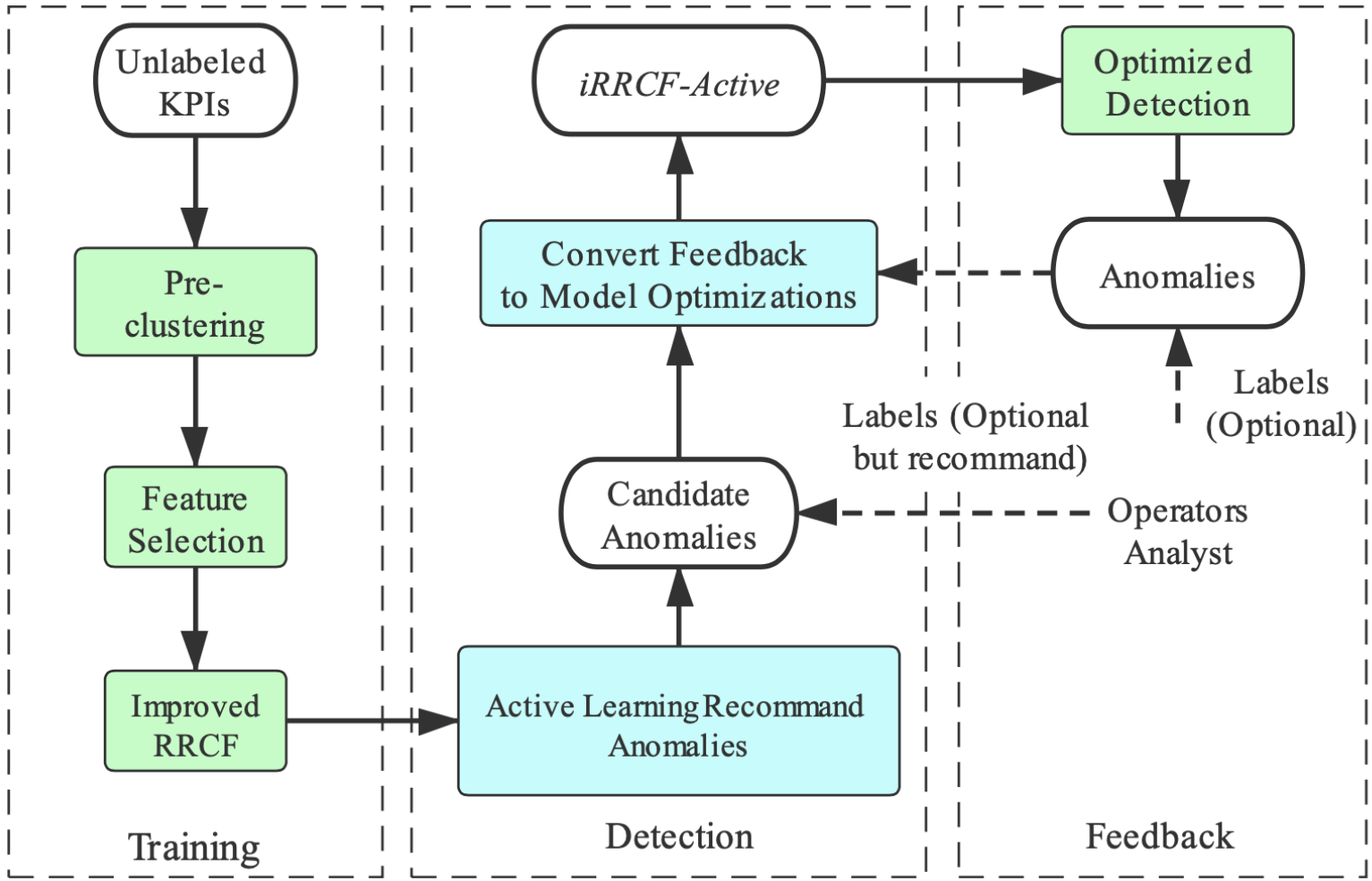


[4] Zeyan Li, Wenxiao Chen, Dan Pei, Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder, **IPCCC 2018**, Orlando, Florida, USA, November 17-19, 2018.

Comparison summary

	Traditional statistical learning	Supervised learning	Traditional unsupervised learning	Unsupervised deep learning model	RRCF-Active
Adapt to KPI variations	Bad	Good	Normal	Good	Good
Low tuning overhead	Bad	Good	Good	Normal	Good
Low labeling overhead	Good	Bad	Good	Good	Good
Fast	Good	Normal	Normal	Normal	Good
Require less resources	Good	Normal	Good	Bad	Good
Learn artificial experience	Bad	Good	Normal	Bad	Good

RRCF-Active Architecture

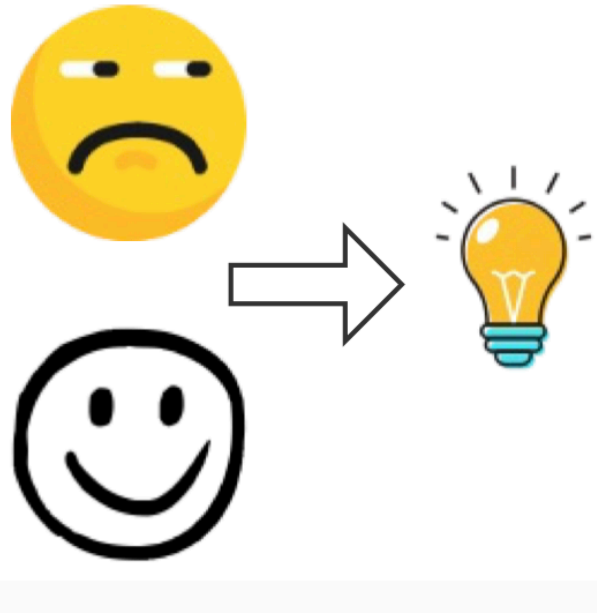


Algorithm Design

Challenge: High-quality labels are difficult to obtain in actual scenes

There are **too many KPIs** to label, and each KPI has a long time span. I will miss many anomalies, and the selecting operations on the timeline are easy to mislabel.

Possible anomalies have been selected, just **click to confirm** whether it is correct or not?



1. **Unsupervised learning**
2. Improve labeling efficiency by **actively learning** recommended labels

Algorithm Design

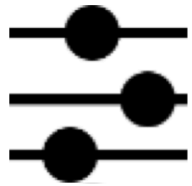
Donut/Bagel

High consumption of training resources and high requirements for KPI periodicity

.....

SVD ARIMA WMA SMA EWMA Holt–Winters

Statistical learning method based on regression



Need to carefully adjust the parameters

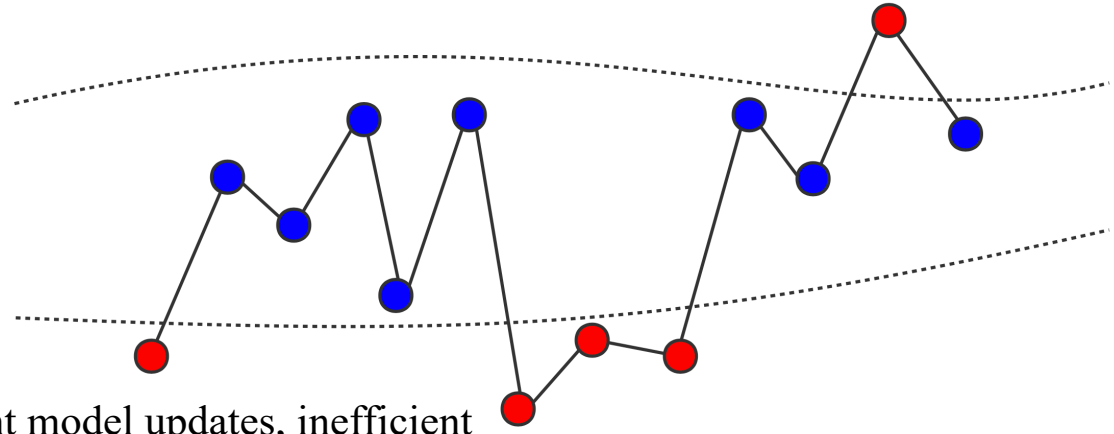
Isolation Forest/RRCF

Re-Train

Slightly adjustment

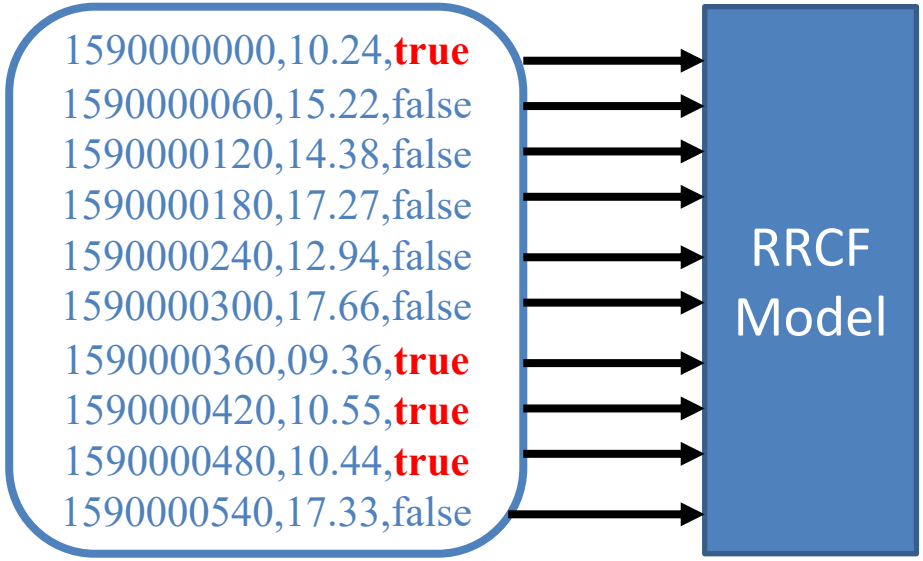
Can’t support precise intervention for a single case

Optimization – Update Process

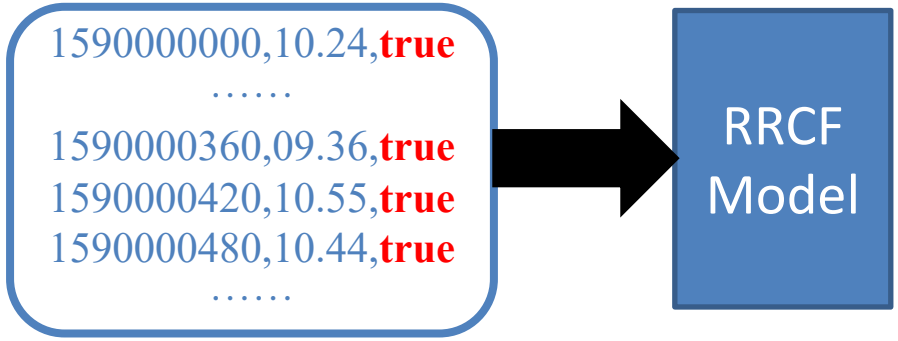


Challenge: **Poor performance of native RRCF**

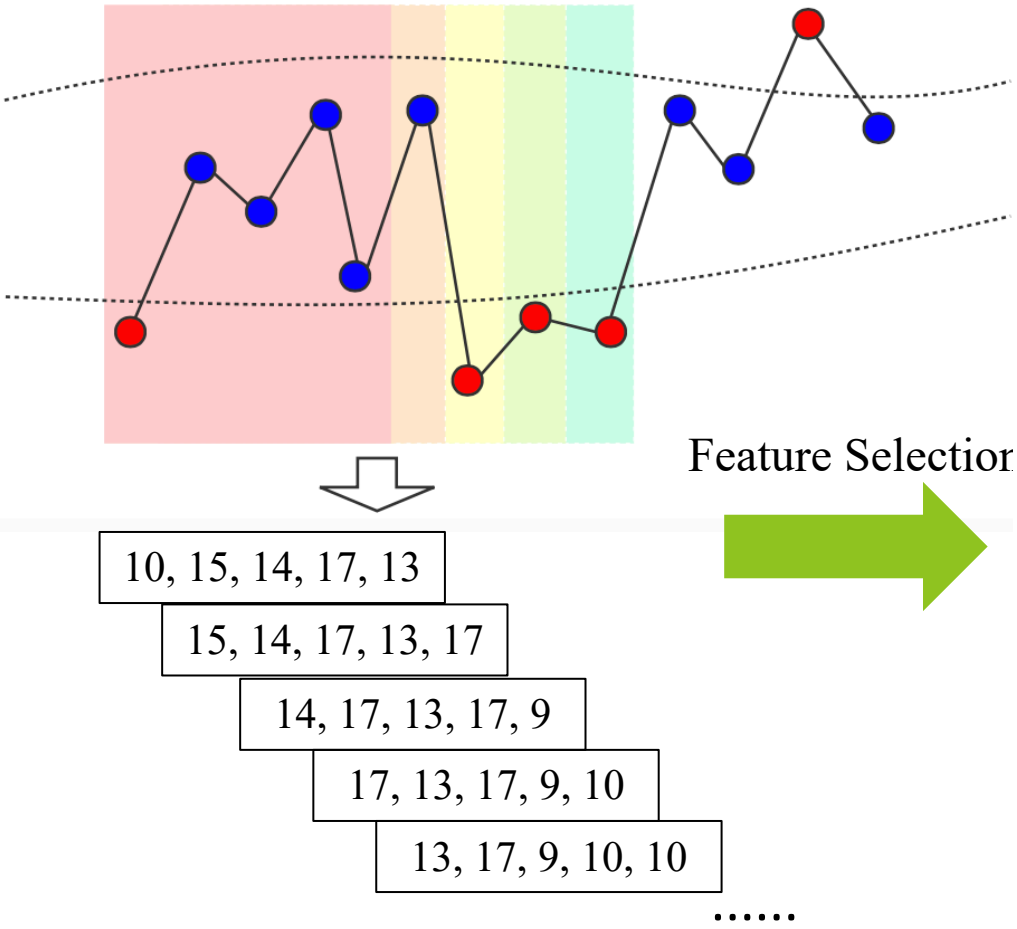
Frequent model updates, inefficient



Filter update samples, reduce the number of updates, and improve update efficiency



Optimization – Feature Selection

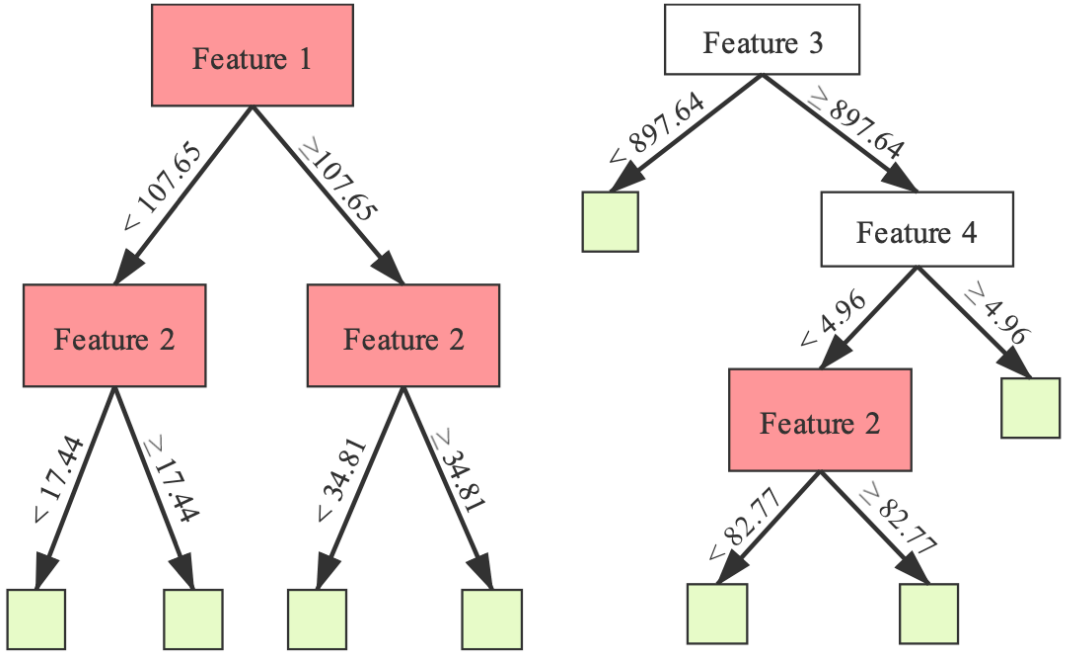


Challenge: The algorithm has **poor adaptability to different KPIs**

FEATURES AND THEIR APPLICABLE CONDITIONS

Feature	Applicable Condition
Median	Few spikiness [29]
Standard deviation	Stationary
Difference from previous period point	Seasonal
Difference from previous point	Trendiness
Second-order difference from the first point	Meet the above 4 points
Third-order exponential average	Predictability

Optimization – Node Cut Dimension Selection

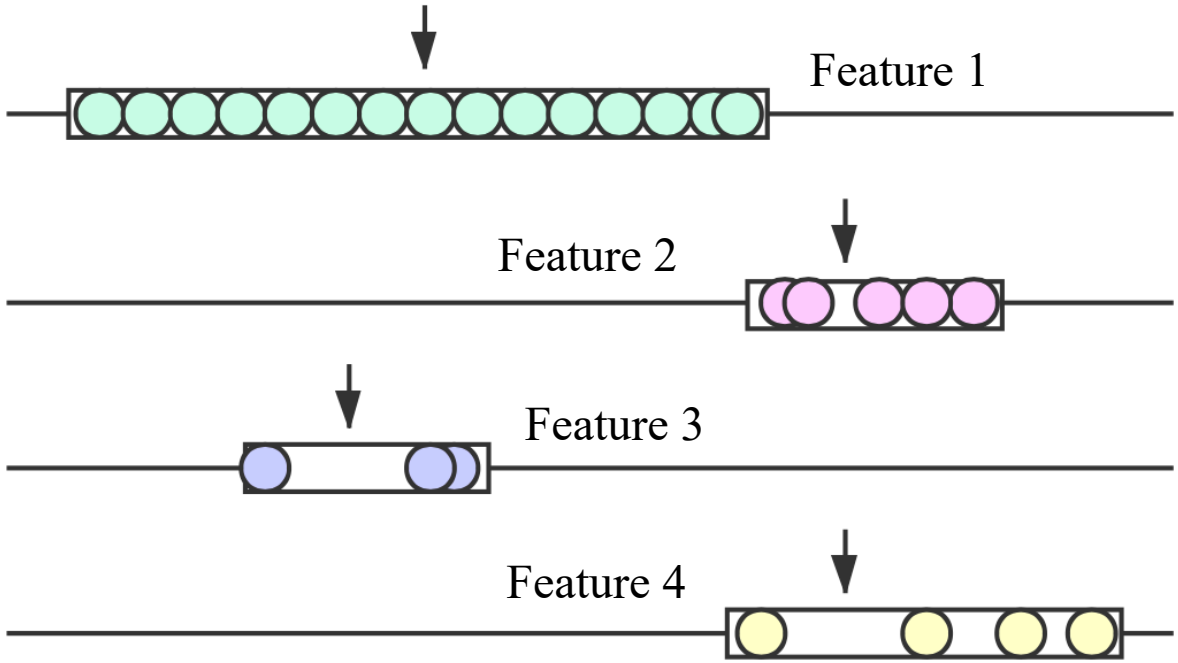


$$g_i = max_{x \in S} x_j - x_{j-1}$$

$$p_i = \frac{g_i}{\sum_j g_j} + \frac{l_i}{\sum_j l_j}$$

Cut Feature selection affects decision tree quality

Optimization – Node Cut Threshold Selection

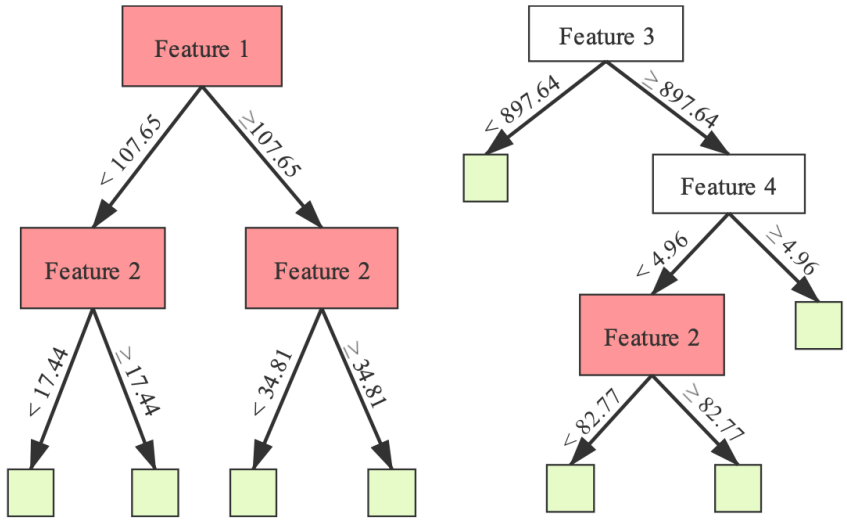


RRCF: Feature1, split randomly

Optimization: Feature 3, cut at the most discriminative position

1) Divide the feature extracted from the training set into N intervals $[l_0, h_0, l_1, h_1, \ldots, l_{N-1}, h_{N-1}]$
2) Compute the density of feature in every interval $d_i = Count(p, p \in [l_i, h_i])$
3) Choose a random interval i proportional to $\frac{d_i}{\sum_j d_j}$
4) Choose $X_i \sim Uniform[l_i, h_i]$

Optimization – Anomaly Score Calculation



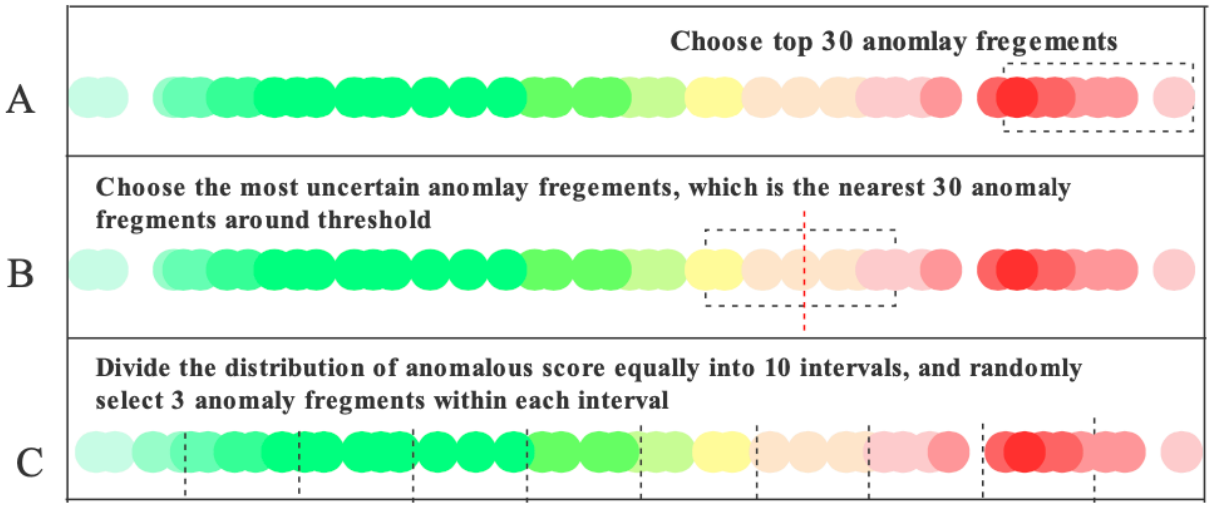
Deeper node has few siblings

1) Find the leaf node $Node$ of the sample x_i in each tree T
2) Count the number of samples in the subtree rooted by sibling and parent of $Node$, denoted by $S_{Node.sibling}$ and $S_{Node.parent}$, and calculate $CoDisp_{Node} = \frac{S_{Node.sibling}}{S_{Node.parent}}$
3) Go up one level in the tree, $Node = Node.parent$
4) Repeat 2 to 3 for N times, where N is the number of features used
5) $CoDisp_T$ is the max value of each $CoDisp_{Node}$
6) The final $CoDisp_{x_i}$ for sample x_i is $\overline{CoDisp_T}, T \in forest$

$$CoDisp_{Node} = \frac{S_{Node.sibling}}{S_{Node.parent} \times Node.depth}$$

Optimization – Active Learning

Challenge: **Adjust algorithm through labels**



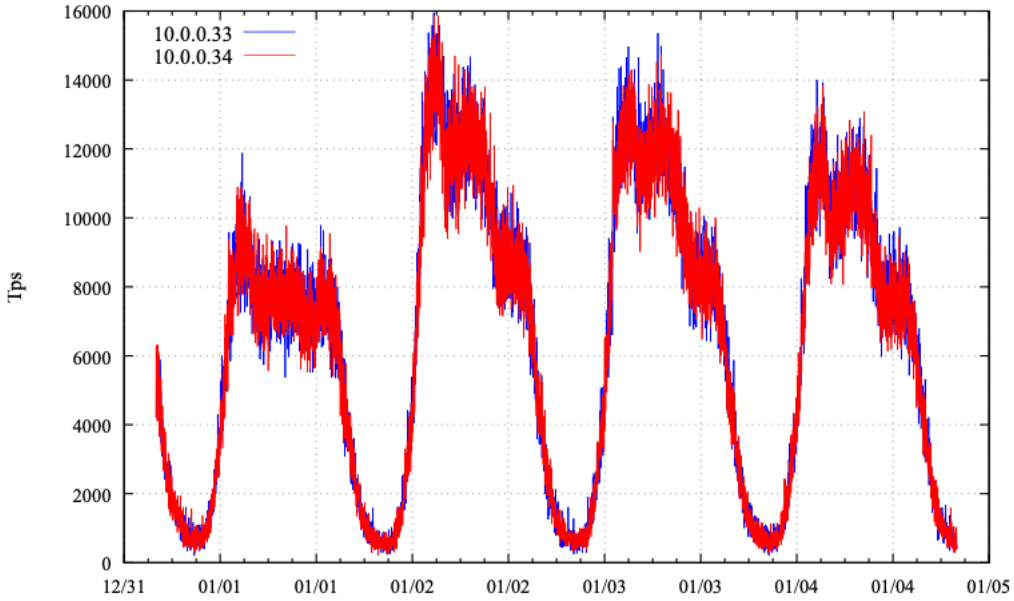
Recommand strategy

Algorithm 2 Update Weight of Trees based on Labels

Require:

Number of trees in forest m
Number of labels n
Anomaly score metric $CoDisp\_M[i][j]$ for label i calculated from tree j
labels provided by users $label$
function GET TREE WEIGHT(m, n, $CoDisp\_M$, $label$)
    $tw \leftarrow Zeros(1, m)$
    for $i = 0 \rightarrow n$ **do**
        if $label[i] == 1$ **then**
            $tw[:] \leftarrow tw[:] + \delta \times CoDisp\_M[i, :]$
        end if
    end for
    return tw
end function

Pre-Clustering



Distance: Dynamic Time Wrapping
Clustering: DBSCAN

29 KPIs -> 12 Group

Model Update Process

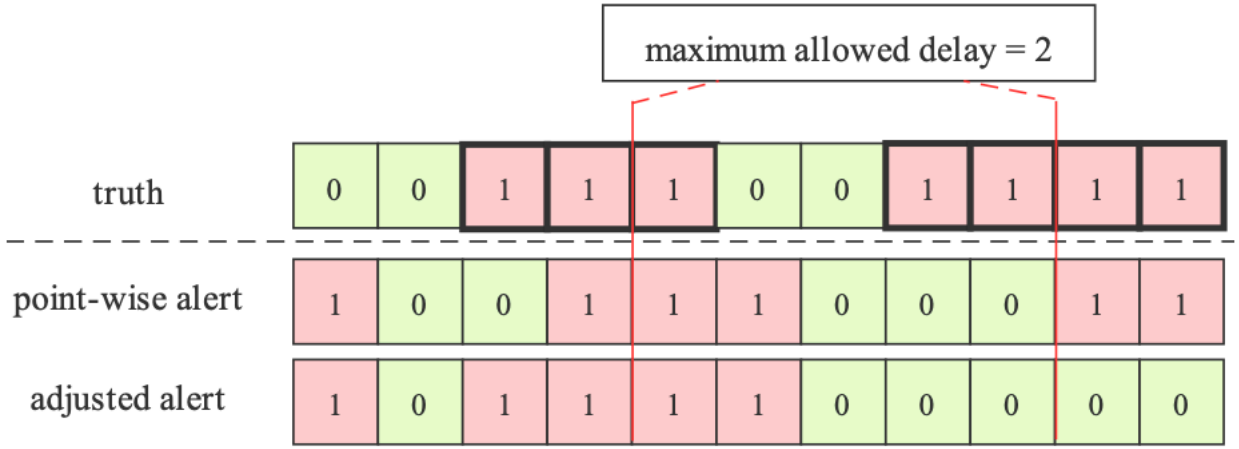
Algorithm 1 Obtain Anomaly Score and Update Model

Require: New data point p, the original tree $tree$

```
1: function GET SCORE(p, tree)
2:     Node ← tree.root
3:     if isinstance(Node, LEAF) then
4:         if p is extreme or already skip 100 points then
5:             INSERT_POINT(p, tree)
6:         end if
7:         return CODISP(p)
8:     end if
9:     if p[Node.dim] <= Node.threshold then
10:        return GET SCORE(p, Node.left)
11:    else
12:        return GET SCORE(p, Node.right)
13:    end if
14: end function
```

1. At least Update for every 100 points
2. Only use extreme points to update (first anomaly in a continuous segment or a normal but extreme value)

Evaluation



OVERALL PERFORMANCE

Method	Best F1-score		Train Time		Detection Time	
	α	β	α	β	α	β
Donut	0.72	0.46	10.9	1.7	2.1	0.43
IF-Active	0.76	0.58	1.4	0.43	1.2	0.25
Isolation Forest	0.58	0.28	**0.17**	**0.09**	**0.11**	**0.06**
Random Forest	0.69	0.44	0.4	0.14	0.5	0.07
Original RRCF	0.70	0.31	0.48	0.37	12.1	3.8
iRRCF-Active	**0.89**	**0.81**	0.93	0.88	7.8	1.6

7 months

29 KPIs

1 min sampling interval

Thank you!

Yao Wang
2020/07