Pluto: Sample Selection for Robust Anomaly Detection on Polluted Log Data

1 APPENDIX

1.1 Alghorithms

Algorithm 1 PollutionEstimator

Input: The dominance of all clusters $\{dom_i\}_{i=1}^m$, the percent threshold p

Output: The set of highly polluted cluster IDs C_h, the set of slightly polluted cluster IDs C_s

1: $C_h \leftarrow \emptyset$
2: $C_s \leftarrow \emptyset$
3: $C_h \leftarrow$ FindMinDominanceClusters($\{dom_i\}_{i=1}^m, p$)
4: $C_{max} \leftarrow$ FindMaxDominanceClusters($\{dom_i\}_{i=1}^m, p$)
5: $dom_{elbow} \leftarrow$ FindLargestAcsendingElbow($\{dom_i\}_{i=1}^m$)
6: **if** $dom_{elbow} \neq Null$ **then**
7: **for each** $c \in C_{max}$ **do**
8: **if** $dom_c > dom_{elbow}$ **then**
9: $C_h \leftarrow C_h \cup c$
10: $C_s \leftarrow \{i\}_{i=1}^m - C_h$
11: $\{r_i\}_{i=1}^m \leftarrow$ AnomalyRatioEstimator($C_s, C_h, \{dom_i\}_{i=1}^m$)
12: **return** $C_h, C_s, \{r_i\}_{i=1}^m$

PollutionEstimator Algorithm. Given the dominance of each cluster, and a percent threshold p, Algo 1 returns the sets of slightly and highly polluted clusters C_s and C_h. Line 3 detects the top p percent clusters of minimal dominance for clusters with high pollution (Mild). For clusters with high pollution (Extreme), Line 4 first finds the top p percent clusters with maximal dominance as the candidate set. Line 5 finds the largest elbow of the dominance. For each cluster in the candidate set, Line 7-9 add a cluster to C_h if its dominance is larger than the elbow. With the set C_h, Line 10 finds the slightly polluted clusters and adds them into C_s. Line 11 estimates the anomaly ratio for each cluster. Line 12 returns the estimated pollution levels and the anomaly ratios for all clusters.

Complexity Analysis. Given m clusters, Line 3, 4, 11 and the for loop in Line 7-9 are all bounded by $O(m)$. For Line 5, we leverage the Kneedle algorithm [8] to find the elbow, whose time complexity is bounded by $\sum_{i=1}^m O(i^2)$. Together, the whole time complexity of Algo 1 is $\sum_{i=1}^m O(i^2)$.

PlutoSubsetSelection Algorithm. Given the required inputs, Algo 2 returns the selected subset by three steps.

- *Step 1: SVD* (Line 2-6). For each cluster, Line 3 applies SVD to obtain the eigenvectors and eigenvalues. Line 4 calculates the dominance with the first and second eigenvalues, and the first and second eigenvectors in Line 5-6 are saved for later usage in Step 3.
- *Step 2: Pollution Estimation* (Line 7). Given the cluster dominance, Line 8 obtains the slightly and highly polluted cluster with the estimated anomaly ratios via Algo 1.
- *Step 3: Subset Selection* (Line 8-15). Step 3 performs subset selection in two cases. For the slightly polluted cluster case in Line 10-11, we select the samples close to the first eigenvector and remove the samples close to the second eigenvector. For the highly polluted cluster case, we select the samples far away from both eigenvectors in Line 13-14. At last, we union the subsets of the clusters as the *base subset* in Line 15, which is returned in Line 16.

Complexity Analysis. The complexity of Step 1 is bounded by the complexity of SVD. Let $|\mathcal{D}|$ be the total number of sequences, d be the embedding dimension, and m be the number of clusters, the complexity of SVD is $O(m|\mathcal{D}|^2 d + md^3)$ [2]. The complexity of Step 2 is bounded by the cost of Algo 1 ($\sum_{i=1}^m O(i^2)$). Under the assumption of a large dataset with $|\mathcal{D}| \gg m$, the complexity of Step 2 is negligible compared to the complexity of SVD. The complexity of Step 3 is bounded by the cosine similarity computation, which is $O(|\mathcal{D}|d^2)$. Therefore, the overall complexity of Algo 2 is $O(m|\mathcal{D}|^2 d + md^3 + |\mathcal{D}|d^2)$, which will be mainly dominated by the complexity of SVD under the assumption of a large dataset with $|\mathcal{D}| \gg d$.

Algorithm 2 PlutoBaseSelection

Input: Source dataset $\mathcal{D}$, Anomaly ratio r of $\mathcal{D}$, Threshold p, m Clusters of embedding matrix $\{C_i\}_{i=1}^m$,

Output: The base subset $\bar{\mathcal{D}}$

1: $\bar{\mathcal{D}} \leftarrow \emptyset$
/* Step 1: SVD for each cluster */
2: **for each** $C_i \in \{C_i\}_{i=1}^m$ **do**
3: $\boldsymbol{U}, \Sigma, \boldsymbol{V} = \text{SVD}(\boldsymbol{C_i})$
4: $\{dom_i\} \leftarrow \{dom_i\} \cup \frac{\text{The First Eigenvalue In } \Sigma}{\text{The Second Eigenvalue In } \Sigma}$
5: $\{\boldsymbol{v}_{1,i}\} \leftarrow \{\boldsymbol{v}_{1,i}\} \cup$ The First Column Of $\boldsymbol{V}$
6: $\{\boldsymbol{v}_{2,i}\} \leftarrow \{\boldsymbol{v}_{2,i}\} \cup$ The Second Column Of $\boldsymbol{V}$
/* Step 2: Pollution estimation */
7: $C_h, C_s, \{r_i\}_{i=1}^m \leftarrow$ **PollutionEstimator**($\{dom_i\}_{i=1}^m, p$)
/* Step 3: Selection */
8: **for each** $\boldsymbol{E_i} \in \{\boldsymbol{E_i}\}_{i=1}^m$ **do**
9: **if** $i \in C_s$ **then**
/* Slightly polluted clusters */
10: $\bar{\mathcal{D}}_i \leftarrow$ FindMaxCosSimSamples($\boldsymbol{C_i}, \boldsymbol{v}_{1,i}, 1 - r_i$)
11: $\bar{\mathcal{D}}_i \leftarrow \bar{\mathcal{D}}_i -$ FindMaxCosSimSamples($\boldsymbol{C_i}, \boldsymbol{v}_{2,i}, r_i$)
12: **else**
/* Highly polluted clusters */
13: $\bar{\mathcal{D}}_i \leftarrow$ FindMinCosSimSamples($\boldsymbol{C_i}, \boldsymbol{v}_{1,i}, r_i$)
14: $\bar{\mathcal{D}}_i \leftarrow \bar{\mathcal{D}}_i \cap$ FindMinCosSimSamples($\boldsymbol{C_i}, \boldsymbol{v}_{2,i}, r_i$)
15: $\bar{\mathcal{D}} \leftarrow \bar{\mathcal{D}} \cup \bar{\mathcal{D}}_i$
16: **return** $\bar{\mathcal{D}}$

PlutoPipeline Algorithm. Given the required inputs, Algo 3 returns the trained anomaly detection model. In Line 1-2, we warm

Algorithm 3 PlutoPipeline

Input: Original dataset $\mathcal{D}$, Anomaly ratio r, Threshold p, Cluster num m; Model $f_\theta(\cdot)$; Warm up epochs W, Refine gap G, Refine iter number R, Training epochs T

Output: The anomaly detection model $f_\theta(\cdot)$

/*Step 1: Warm up */

1: **for each** $e \in \{1, \ldots, W\}$ **do**

2: Train($f_\theta(\cdot), \mathcal{D}$)

3: $\boldsymbol{E} \leftarrow f_\theta(\mathcal{D})$

/*Step 2: Iterative selection refinement */

4: $\bar{\mathcal{D}} \leftarrow$ IterativeSelection($\mathcal{D}, \boldsymbol{E}, r, p, m, f_\theta(\cdot), G, R$)

/*Step 3: Train on refined subset */

5: Re-initialize($f_\theta(\cdot)$)

6: **for each** $e \in \{1, \ldots, T\}$ **do**

7: Train($f_\theta(\cdot), \bar{\mathcal{D}}$)

8: **return** $f_\theta(\cdot)$

up the model $f_\theta(\cdot)$ on the original source dataset $\mathcal{D}$ to obtain the sequence embeddings in Line 3. Line 4 returns the final subset $\bar{\mathcal{D}}$ returned by the IterativeSelection in Algo **??**. Then, in Line 6-7, we re-train the model on $\bar{\mathcal{D}}$ for T epochs.

1.2 Implementation Details

PCA [12], IsolationForest [7], LogCluster [6], OCSVM [9]. All implementations are from the public repository [1] and [2].

Deeplog [1]. The implementation is from the public repository [1]. We follow the default settings for all four datasets. Both the batch size and the hidden size are set to 32 with a window size of 10. The learning rate is set to 1e-3. We train the Deeplog for 200 epochs.

OC4Seq [11]. The implementation is from the public repository [3]. We follow the default settings for all four datasets. The input dim, output dim and batch size are all set to 64. The layer number is set to 2 and the learning rate is set to 1e-3. We train the OC4Seq for 200 epochs.

Logbert [3] and Logbert-. The implementation is from the public repository [4]. We follow most of the default settings. The Logbert variants have the following settings: 256 as the hidden size, 4 layers of encoders with 4 attention heads. The batch size is set the 32 and the mask ratio is set to 0.5. We use 1e-3 as the learning rate for BGL, ThunderBird, and Spirit datasets, and 1e-4 for the HDFS dataset. LogBert uses Masked Language Model (MLM) and (Volume Hypersphere Minimization) VHM tasks for training, with the default weights 1.0 and 0.1. LogBert- uses only the MLM task for training. We train Logbert variants for 200 epochs.

Co-teaching [4]. The implementation is from the public repository [5] with the implementation of LogBert-. We use the default setting of Co-teaching. The noisy rate is given with the real anomaly ratio of the training set. The gradual number is set to 20 and the exponent of the forget rate is set to 1. We use the loss of the (MLM) task for the Co-teaching sample selection. To accommodate the different sequence lengths, we use the normed sequence loss by the sequence length for Co-teaching.

[1]https://github.com/logpai/loglizer
[2]https://github.com/yzhao062/pyod
[3]https://github.com/KnowledgeDiscovery/OC4Seq
[4]https://github.com/HelenGuohx/logbert
[5]https://github.com/bhanML/Co-teaching

ITLM [10]. The implementation is from the public repository [6] with the implementation of LogBert-. We set the empirical good sample ratio to 0.8 for all four datasets. Similar to Co-teaching, we use the normed sequence loss as the selection criterion for ITLM. We run ITLM after G epochs of training. The G settings are the same with Pluto.

FINE [5]. The implementation is from the public repository [7] with the implementation of LogBert-. Since FINE works in each class of data, same with Pluto , we apply the clustering algorithm first and perform FINE on each cluster. Following the default setting, we set the cosine similarity threshold to 0.5. Since FINE selects subsets recursively, we run FINE after G epochs of training. The G settings are the same with Pluto .

Pluto. The implementation is provided in Supplementary with the implementation of LogBert-. For all datasets, we set the cluster number m to 20. The percent threshold of highly polluted clusters p is set to 80 (BGL), 90 (HDFS), 95 (ThunderBird), 85 (Spirit). The selection gap G is set to 10 (BGL), 5 (HDFS), 15 (ThunderBird) and 10 (Spirit). The selection iteration number R is set to 10 (BGL, HDFS), 1 (ThunderBird), and 16(Spirit).

Anomaly Detection Criteria. For pattern mining and similarity comparison methods, including PCA, IsolationForest, LogCluster, and OCSVM, we report the best F-1 from all thresholds generated by roc_curve function of the python package sklearn. For OC4Seq, we store the anomaly score and report the best F-1 from all thresholds generated by roc_curve function of the Python package sklearn. For methods using the top t log key prediction, including DeepLog, Logbert variants, FINE, Co-teaching, ITLM, and Pluto, we select the same t for all the methods, the log keys whose gound-truth are not in the top t predictions are considered abnormal, and the anomaly score of the sequence is defined as the ratio of the abnormal log keys in the sequence.

[6]https://github.com/yanyao-shen/ITLM-simplecode
[7]https://github.com/Kthyeon/FINE_official

REFERENCES

[1] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 1285–1298.

[2] Gene H Golub and Charles F Van Loan. 2013. *Matrix computations*. JHU press.

[3] Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*. IEEE, 1–8.

[4] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems* 31 (2018).

[5] Taehyeon Kim, Jongwoo Ko, JinHwan Choi, Se-Young Yun, et al. 2021. Fine samples for learning with noisy labels. *Advances in Neural Information Processing Systems* 34 (2021), 24137–24149.

[6] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. 2016. Log clustering based problem identification for online service systems. In *Proceedings of the 38th International Conference on Software Engineering Companion*. 102–111.

[7] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[8] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. 2011. Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*. 166–171. https://doi.org/10.1109/ICDCSW.2011.20

[9] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.

[10] Yanyao Shen and Sujay Sanghavi. 2019. Learning with bad training data via iterative trimmed loss minimization. In *International Conference on Machine Learning*. PMLR, 5739–5748.

[11] Zhiwei Wang, Zhengzhang Chen, Jingchao Ni, Hui Liu, Haifeng Chen, and Jiliang Tang. 2021. Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 3726–3734.

[12] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael Jordan. 2009. Largescale system problem detection by mining console logs. *Proceedings of SOSP'09* (2009).