Multi-view Failure Analysis on Multivariate Time-series Data

Hao Huang*, Yunwen Xu† and Shinjae Yoo‡
*Machine Learning Lab, GE Global Research, Email: haohuangcssbu@gmail.com
†Core Machine Learning Team, Amazon, Email: xuyunwen04@gmail.com
‡Computational Science Center, Brookhaven National Lab, Email: sjyoo@bnl.gov

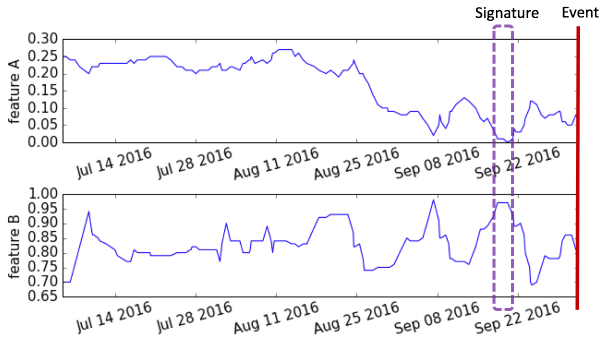
***Abstract*—to be finished**

I. Introduction

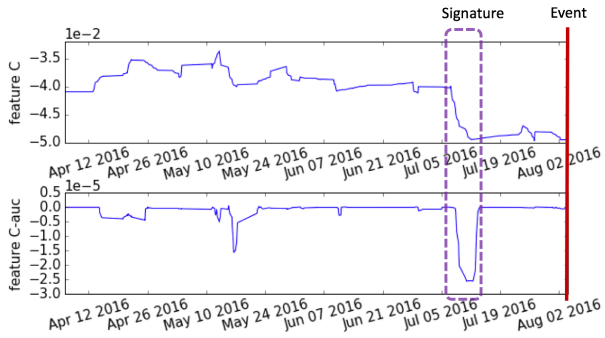
With the rapid advances in sensor design, data storage and networking, we are facing an explosive growth of complexity in analyzing failure (or event) in multivariate time series data. Failure analytics system can be identified from the knowledge data discovery process where the output gives the fault signatures that relate to the failure, while the input data correspond to recorded sensor measurements that are considered as features. These high-frequent, large-scale data often require fleet level analysis, which not only involve the event assets, but also the healthy assets (usually much more than those event ones), therefore a unique fault signature relates to the failure can be found and a detector with low-false-alarm-rate can be created. The output of such signature should consist of or directly relate to the weighting of input sensor, therefore more effectively support the root cause analysis. Furthermore, a forecast model can be built consequently that gives early warning of the same failure in the future.

Besides large-volume of data on the fleet level, we are also facing the following challenges in finding such signatures: 1) The fault signatures usually exist in multivariate time series involving a lot of assets, which makes it hard to detect by any univariate and uni-asset analytics; 2) While the event time can be known, the time that faulty signatures happen is usually unknown, and could be different from case to case, which makes it difficult to detect by any (semi-)supervised machine learning techniques.

Figure 1 shows the two signature examples from ground truth by domain experts. In Figure 1(a) the signature appears 15 days before the failure, and the signature is with small value on feature A and large value on feature B. While in Figure 1(b) the signature appears almost one month before the failure, where there is a level-shift on feature C that can be depicted by auto-correlation (auc). It is interesting and important to notice that 1) both of these examples show significant lag between signature and failure, and 2) the lasting time of signature are different which is also unknown for the analytics. As a matter of fact, this "lag" patterns are usual and could be with different reasons. One of the reasons is that the input sensors could be indirectly related to the failure. For example, an engine stall event may happen because some component is crack. But due to the complexity of the engine design, it takes certain time for such fault propagate to the whole system which leads to the final failure. Right before the crack, the vibration sensors of the faulty component can show strong vibrating patterns, which can be viewed as the faulty signature. But once the crack happen, this component becomes unconnected to the whole system and the vibration sensor recording become stable. Therefore the signature cannot be found right before the final failure but only before the time of crack.



(a) Event 1



(b) Event 2

Fig. 1. Two examples of fault signatures with lag to final failures (events). In Figure 1(a) the signature appears 15 days before the failure, with small value on feature A and large value on feature B. While in Figure 1(b) the signature appears almost one month before the failure, where there is a level-shift on feature C that can be capture by auto-correlation (auc).

It can be easily imagined that, if we natively label the samples right before failure as "1" and perform classic supervised machine learning algorithm [1], we are not able to find such fault signature. The key difference of the problem, compared to the popular definition of supervised problems, is that **our label "1"s are uncertain and even most of them could be wrong**. For example in Figure 1(a), without any prior knowledge if we label all samples 30 days before the

[1] In a classic supervised setting, the event samples are considered as positive and usually labeled as "1", while the other are considered as normal and labeled as "0".

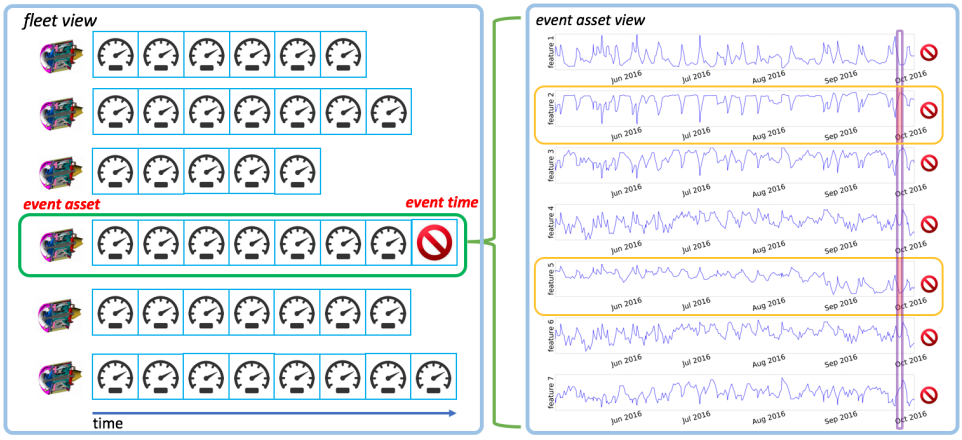


Fig. 2. Illustration of Multi-view Failure Analysis on Multivariate Time-series Data. Given the event asset and event time on the fleet level data, our model can provide the fault signature (marked by red shadow) on the feature level (rounded by yellow box) and timestamp level (rounded by purple box).

final failure as "1", the classic feature selection like random forest, lasso-based or mutual-information based methods will be easily failed, since only 10% of "1" are correct and the rest should be treated as 0 (normal class). To handle this kind of problem, this paper proposes a Multi-view Failure Analysis on Multivariate Time-series Data (MAMT) which overcomes the limitations of classic supervised algorithms on such problem. Our contributions are as follows:

(1) We proposed a novel fleet analytics model that aims to select key features that relate to the failure on multivariate time-series data, which can effectively support the root cause analysis.
(2) Our proposed model, compared with the existing supervised methods, automatically detect the time of signature by filtering the "wrong labels".
(3) Our proposed model, compared with the existing methods, achieves a similar or even lower time and space computational complexity, but a more desired effectiveness.

Figure 2 illustrates that such multi-view analytics is unique and different from the existing works on any supervised feature selection or instance selection for multivariate temporal data. The left side shows the data on the fleet-level view and the event asset and event time is already known. The right side shows the multivariate time series for the event asset, and the fault signature on the feature level (rounded by yellow box) and timestamp level (rounded by purple box) are unknown. Without any more prior knowledge (other than the event asset and event time), our model can provide the fault signature (marked by red shadow) on multi-view level.

II. Problem Definition

In this section, we will mathematically define our problem setting. We start off by formally defining our notation. For a matrix $Z \in \mathbb{R}^{m\times n}$, $Z^\top$ denotes its transpose, and Z^{-1} denotes its (pseudo)inverse. We also use entry-wise norms denoted by $\|Z\|_p$, where $p = 2$ gives (Frobenius norm) $\|Z\|_F^2 = \sum_{ij} z_{i,j}^2 = \mathrm{tr}(Z^\top Z)$, and $p = (2,1)$ gives the $\ell_{2,1}$ norm $\|Z\|_{2,1} = \sum_{i=1}^m \|z_{i,:}^2\|$ where $z_{i,:}$ denotes the ith row of Z. For a vector $(w_1, ..., w_m) \in \mathbb{R}^{m\times 1}$, $\mathrm{diag}(w_1, ..., w_m) \in \mathbb{R}^{m\times m}$ denotes a diagonal matrix with $w_1, ..., w_m$ as its diagonal entries. Let $\mathbb{I}_m$ denote an identity matrix of dimension $m \times m$.

Let $\mathbf{X} = [x_1, x_2, ..., x_n] \in \mathbb{R}^{m\times n}$ where x_i represents a sample ($m \times 1$ vector) at a certain timestamp in a time series feature space, and $\mathbf{f} = [f_1, f_2, ..., f_m]$ be the set of features. On a fleet level analytics, $\mathbf{X}$ may involve samples from a few assets. And the feature set may not only include the raw sensors output, but also the transformed time series features, which could be created by sliding-window-based methods like mean, variance and auto-correlation. We will further discuss the ways of time series feature transformation in Section VI.

Now consider we have one event asset and the event time is already known. With certain domain knowledge, we can confidently assume that the fault appears no earlier than a timestamp samples right before the failure. Without loss of generality, let $\mathbf{X_a} = [x_1, x_2, ..., x_a] \in \mathbb{R}^{m\times a}$ represent the a timestamp samples right before the failure, and $\mathbf{X_b} = [x_{a+1}, x_{a+2}, ..., x_{a+b}] \in \mathbb{R}^{m\times b}$ represent all the other samples we consider as normal and irrelevant to the failure, and usually $b \gg a$. We denote $\mathbf{X} = [\mathbf{X_a}, \mathbf{X_b}] \in \mathbb{R}^{m\times n}$ where $n = a + b$. The output we expect is a feature score vector $\mathbf{W} \in \mathbb{R}^{m\times 1}$, and a instance score vector $\mathbf{Y_a} \in \mathbb{R}^{1\times a}$, where each element represents the contribution of the feature/instance to the fault. The larger the value is, the more relevant the feature/instance is to the fault.

III. Algorithm

The key difference of this problem, compared against the popular definition of supervised problems, is that while our label "0"s (or most of them, if not all) are reliable since we know there is not similar failure happened at those time or assets, our label "1"s are uncertain and even most of them could be wrong. In this section we introduce our framework to solve this problem, which consists of supervised feature selection and dynamic and directional label propagation.

A. Supervised Feature Selection

Intuitively, the requirement of failure signature analytics makes it closely connected to supervised feature selection. Thereby we first introduce a representative feature selection

method based on $\ell_{2,1}$ norm regularization [3] as our baseline feature selection algorithm.

$$\min_{W} \|W^\top X - Y\|_F^2 + \alpha \|W\|_{2,1}, \tag{1}$$

where $X \in \mathbb{R}^{m \times n}$ is the input dataset with n samples and m features, and $Y \in \mathbb{R}^{1 \times n}$ is the label vector in a binary classes setting with "1" means the corresponding sample is considered as interesting and "0" means it is normal sample. The final output $W \in \mathbb{R}^{m \times 1}$ gives a feature weighting vector represents the importance of each feature to the classification.

The problem in Eq.1 can be interpreted as a generalized $\ell_{2,1}$ norm regularization problem [3], where the first term is a smooth convex loss function, and the second term controls the capacity of W and also ensures that W is sparse in rows, with the parameter α controlling the sparsity of W.

B. Dynamic and Directional Label Propagation

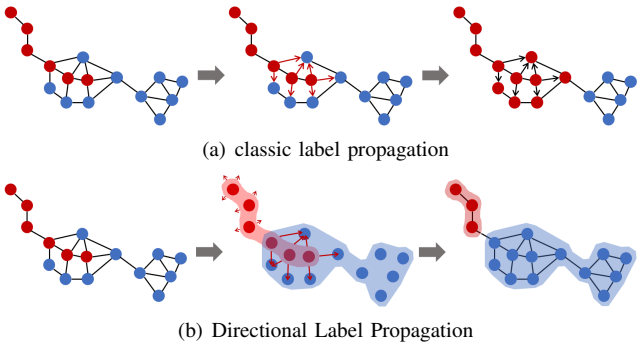
As we mentioned earlier, what makes this problem unique to the popular feature selection, is that the label vector Y is not able to be relied on. In some application cases it could be over 90% mislabeled. Given such serious noisy labeling situation, the popular feature selection, and even the existing work on handling noisy labels [5], [2] will easily fail since their assumption is only a small portion of labels are wrong.

Here we introduce a Dynamic and Directional Label Propagation which can iteratively filter out wrong labels while maintaining those right ones. We start off by briefly describing the classic Label Propagation technique [6], [4]. Consider a regularization framework on graph, the cost function associated with label propagation of Y is defined as

$$\min_{Y} \sum_{i,j=1}^{n} \tilde{A}_{ij} \|Y_i - Y_j\|_F^2 + \mu \sum_{i=1}^{n} \|Y_i - E_i\|^2, \tag{2}$$

where $Y \in \mathbb{R}^{1 \times n}$ is the label vector in a binary classes setting with "1" means the corresponding sample is considered as interesting and "0" means it is normal sample, $E \in \mathbb{R}^{1 \times n}$ is the label initialization, and $\tilde{A} \in \mathbb{R}^{n \times n}$ is a normalized affinity matrix. The problem in Eq.2 can be interpreted as a trade-off optimization problem, where the first term is a global smoothness meaning that two close samples should share the same labels, and the second term is a local fitness meaning that the final label should not be dramatically different from the initial seed, with parameter μ controlling the trade-off. Label Propagation has strong connection with random walk, in the way that the label propagate out with the process of random walk [4], which built upon a stable transition probability among samples. Figure 3(a) shows the illustration of classic label propagation in a binary setting. We can see that given the initial label seed and a well constructed graph, the final label shows good classification result.

However, we have a different purpose in our problem setting: **our goal is more on filtering out the wrong "1" rather than expanding the right "1"**. Therefore, instead of constructing an omnidirectional random walk, we model the propagation through a directional random walk as shown in Figure 3(b). Given the whole dataset as $\mathbf{X} = [\mathbf{X_a}, \mathbf{X_b}] \in \mathbb{R}^{m \times n}$



(a) classic label propagation

(b) Directional Label Propagation

Fig. 3. Illustration of classic and our proposed label propagation. Red indicates the samples labeled as "1" and blue as the "0".

with m features and n samples, and $\mathbf{X_a} \in \mathbb{R}^{m \times a}$ is initially labeled as "1" while $\mathbf{X_b} \in \mathbb{R}^{m \times b}$ as "0", we can construct a similarity matrix $A \in \mathbb{R}^{n \times n}$. We split matrix A into 4 blocks

$$A = \begin{bmatrix} A_{aa} & A_{ab} \\ A_{ba} & A_{bb} \end{bmatrix}, \tag{3}$$

where A_{aa} describes the transition within $\mathbf{X_a}$ and A_{ab} describes the transition from $\mathbf{X_a}$ to $\mathbf{X_b}$ and so forth.

In our problem setting, we assume that the samples in $\mathbf{X_a}$ that labeled incorrectly in the initial seed are closed to (partial of) $\mathbf{X_b}$ on all the feature subspace, and only those labeled correctly as "1" are separable from $\mathbf{X_b}$ in certain feature subspace. Therefore in a specifically designed label propagation, we hope the wrong "1" is propagated out to its "0" neighborhood and the right "1" can be maintained. Detailedly speaking, we allow the "1" labels propagate from $\mathbf{X_a}$ to $\mathbf{X_b}$ only, but not propagate back to $\mathbf{X_a}$. Also we do not pay attention to the propagation within $\mathbf{X_b}$ since no relevant event happened here and all these samples are assumed to be normal. Furthermore, we only allow self-loop exist in $\mathbf{X_a}$ so the right "1" can be conserved even there is high density in $\mathbf{X_a}$. Therefore matrix A is redefined as

$$A = \begin{bmatrix} \mathrm{diag}(A_{aa}) & A_{ab} \\ \mathbf{0}_{ba} & \mathbf{0}_{bb} \end{bmatrix}, \tag{4}$$

where the lower part of A are set to be $\mathbf{0}$ matrix. Figure 3(b) shows the illustration of our proposed Directional Label Propagation. We can see that given directional control, the wrong labels are "flipped" and only the right ones conserved.

C. Our Proposed Framework

With our solutions to both feature side and label side, we are ready to introduce the Multi-view Failure Analysis on Multivariate Time-series Data (MAMT) framework. Considering both selecting the key feature (Eq.1) and rectifying the labels (Eq.2), MAMT is to solve the following optimization

$$\begin{aligned} \mathcal{M}(W, \hat{Y}) = \min_{W, \hat{Y}} & \|W^\top XB - \hat{Y}B\|_F^2 + \alpha \|W\|_{2,1} + \gamma \|\hat{Y}\|_{2,1} \\ & + \delta \left[\sum_{i,j=1}^{n} \tilde{A}_{ij} \|\hat{Y}_i - \hat{Y}_j\|_F^2 + \mu \sum_{i=1}^{n} \|\hat{Y}_i - E_i\|^2 \right], \end{aligned}$$

where $\hat{Y} = Y \circ E$ is the Hadamard product of Y and E, while $E = [\mathbf{1}_a, \mathbf{0}_b] \in \mathbb{R}^{1 \times n}$ is the initial labels with $\mathbf{1}_a =$

$[1, 1, ..., 1] \in \mathbb{R}^{1\times a}$ corresponds to $\mathbf{X_a}$ and $\mathbf{0}_b = [0, 0, ..., 0] \in \mathbb{R}^{1\times b}$ corresponds to $\mathbf{X_b}$, and $B = \text{diag}([\beta\mathbf{1}_a, \mathbf{1}_b])$ is the sample weights in a diagonal format. The reason of bringing $\hat{Y}$ is because we want to "compress" the label of $\mathbf{X_b}$ all the time, and the reason of bringing B is to handle the imbalance problem between $\mathbf{X_a}$ and $\mathbf{X_b}$. Please note that we also bring in a $\ell_{2,1}$ norm regularization on $\hat{Y}$ to emphasize its scarcity in the final solution. In Section III-D we will verify that the $\mathcal{M}(W, \hat{Y})$ is joinly convex with W and $\hat{Y}$.

It is difficult to optimize W and $\hat{Y}$ simultaneously. Therefore we adopt an alternating optimization to solve this problem, which works well for a number of practical optimization problems[1], [5].

Given $\hat{Y}$, optimize W.

Given W, optimize $\hat{Y}$.

D. Proof and Early Stopping

E. Whole algorithm

IV. Complexity Analysis and Lighter Implementation

V. Related Work and Discussion

(1) L.
(2) Manifold on Xa.
(3) multi-events.

VI. Experiment

VII. Conclusion

References

[1] Yinfu Feng, Jun Xiao, Yueting Zhuang, and Xiaoming Liu. Adaptive unsupervised multi-view feature selection for visual concept recognition. In *Asian Conference on Computer Vision*, pages 343–357. Springer, 2012.

[2] Benoít Frénay, Gauthier Doquire, and Michel Verleysen. Estimating mutual information for feature selection in the presence of label noise. *Computational Statistics & Data Analysis*, 71:832–848, 2014.

[3] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 339–348. AUAI Press, 2009.

[4] Feiping Nie, Shiming Xiang, Yun Liu, and Changshui Zhang. A general graph-based semi-supervised learning with novel class discovery. *Neural Computing and Applications*, 19(4):549–555, 2010.

[5] Jiliang Tang and Huan Liu. Coselect: Feature selection with instance selection for social media data. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 695–703. SIAM, 2013.

[6] Jun Wang, Tony Jebara, and Shih-Fu Chang. Graph transduction via alternating minimization. In *Proceedings of the 25th international conference on Machine learning*, pages 1144–1151. ACM, 2008.