The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Directional Label Rectification in Adaptive Graph

Xiaoqian Wang,[1*] **Hao Huang**[2]
[1]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA, 15261
[2]GE Global Research, San Ramon, CA, USA, 94583
xqwang1991@gmail.com, haohuanghw@gmail.com

Abstract

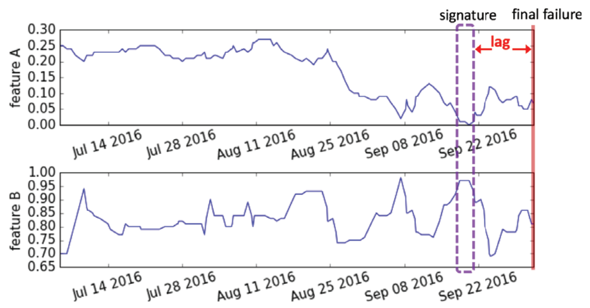
With the explosive growth of multivariate time-series data, failure (event) analysis has gained widespread applications. A primary goal for failure analysis is to identify the fault signature, *i.e.,* the unique feature pattern to distinguish failure events. However, the complex nature of multivariate time-series data brings challenge in the detection of fault signature. Given a time series from a failure event, the fault signature and the onset of failure are not necessarily adjacent, and the interval between the signature and failure is usually unknown. The uncertainty of such interval causes the uncertainty in labeling timestamps, thus makes it inapplicable to directly employ any standard supervised algorithms in signature detection. To address this problem, we present a novel directional label rectification model which identifies the fault-relevant timestamps and features in a simultaneous approach. Different from previous graph-based label propagation models using fixed graph, we propose to learn an adaptive graph which is optimal for the label rectification process. We conduct extensive experiments on both synthetic and real world datasets and illustrate the advantage of our model in both effectiveness and efficiency.

Introduction

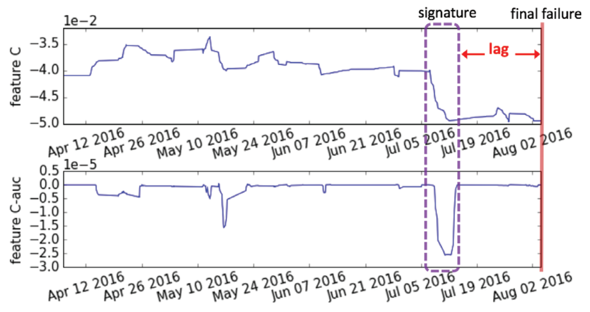
Recent advances in data collection, data storage and their wide application in industrial business have led to the explosion of time-series data. The availability of large-scale temporal datasets facilitates the in-depth study of the dynamic change and the trend evolution in different applications, whereas the complex nature of multivariate time-series data also brings challenges in data analysis.

Given the large volume of multivariate data from different sensors, fault analysis system is designed to identify the fault signatures that characterize the failure events. The fault signature is usually represented by certain feature patterns *w.r.t.* sensor records. When the input is on **fleet level**, where the data involves both failure and healthy assets (data sources), the identified fleet-level fault signature can be naturally employed for the discrimination between failure and healthy assets, thus strongly supports the root cause analysis in early warning and anomaly detection system.

[*]Work done in part during internship at GE Global Research.



(a) Event 1



(b) Event 2

Figure 1: Illustration of two examples in real application with unknown interval between the signature and the final failure. In Fig. 1(a), the signature is characterized by the concurrent occurrence of low value of feature A and high value of feature B, which appears 15 days before the final failure and lasts for only 2 days. In Fig. 1(b), the signature corresponds to a level-shift in feature C, which happens around one month before the failure with the duration of merely 5 days. Such level-shift signature can be depicted by sliding-window based auto-correlation, *i.e.,* feature C-auc.

There are several challenges in fleet-level fault signature discovery for multivariate time-series data: 1) The multivariate sensor records consist of not only features relevant to the failure, but also a lot of redundant or irrelevant features; 2) In a time series of a failure asset, the fault signature is not necessarily adjacent to the final failure. As consequence, even though the onset timestamp of failure is observed, the time when the fault signature appears is usually unknown. Fig. 1

presents two different types of fault signature in industrial application. We can notice that in both cases a lag exists between the signature and the final failure, while the time of occurrence and length of the signature varies.

In real world industrial problems, such "short" signature and "lag" patterns are common and can be caused by different reasons: 1) Time delay between the signature and the failure. As the system design is fairly complex, it may take certain time for the root cause to spread to the whole system and finally trigger the failure; 2) Bad data quality. Severe data loss is sometimes accompanied by a fault signature, which leads to partial capturing of the real signature; 3) Sliding window based time-series feature construction may introduce a lag as shown in Fig. 1(b). Unfortunately, due to the complexity in the large volume of noisy multivariate temporal data, it is extremely difficult for any domain experts or existing machine learning techniques to accurately target such fault signatures.

The uncertainty in the interval between signature and final failure introduces uncertainty in labeling each timestamp. It is easy to see that standard supervised frameworks are not suitable for this situation, since they require the label information (most if not all) to be reliable. If we label the timestamps adjacent to failure as positive (relevant to the fault signature) while the rest as negative, traditional supervised learning approaches will fail due to the bad quality of positive labels. For example, in Fig. 1(b), if we simply label all instances within 60 days of the failure as positive while the rest as negative, over 90% of the positive labels will be misleading for the learning process. To address this problem, we propose a novel label rectification model which overcomes the limitations of classic supervised methods on such problem. We would like to point out the following contributions:

- We propose a novel "directional" label rectification algorithm which automatically rectify the initial event labels and distinguish the truly failure-relevant instances.
- We propose to automatically select the most relevant features to the failure.
- We propose to adaptively learn an optimal affinity matrix to represent the graph structure in label rectification.
- Our model is fairly efficient and scalable to large data.

Problem Definition

We begin this section by introducing several terminologies in real industrial failure analysis. A **time series** is a sequence of measurements indexed in time order. An **asset** is a data source of a time series, *e.g.,* a stock in the financial market, or a machine that is monitored over time. Each **instance** is recorded at a certain timestamp of an asset. An **event** is a failure case which happens at certain timestamps in an asset. A **failure mode** is a possible way or mode that a system might fail. In one analytic task, we usually concentrate on one specific failure mode, with a few events collected from one or multiple assets. Our target in failure analysis is to identify the **fault signature** across these events, which indicates the unique feature patterns to distinguish failure events from the majority of healthy data.

Throughout this paper, unless otherwise specified, we use uppercase letters to denote matrices, bold lowercase letters for vectors and non-bold lowercase letters for scalars. For a matrix $Z \in \mathbb{R}^{m\times n}$, we use $Z(i,:)$, $Z(:,j)$ and $Z(i,j)$ to denote its i-th row, j-th column and ij-th element respectively. The Frobenius norm of Z is denoted as $\|Z\|_F = \sqrt{\sum_{i,j} Z(i,j)^2} = \sqrt{\mathrm{tr}(ZZ^\top)}$, and its $\ell_{2,1}$-norm as $\|Z\|_{2,1} = \sum_{i=1}^{m} \|Z(i,:)\|$, where $\|Z(i,:)\| = \sqrt{\sum_j Z(i,j)^2}$ is the ℓ_2-norm of $Z(i,:)$. For a vector $\mathbf{z} \in \mathbb{R}^m$, we denote its i-th element as z_i. $\mathbb{I}_m$ denotes an $m \times m$ identity matrix. $\mathbf{1}$ denotes a vector with all elements being 1.

Specially, we use $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ to denote our input data, where m is the number of features and n is the number of instances. In a fleet level analysis, instances in X usually come from several different assets. The features in X may involve not only raw sensor recordings, but also time series features constructed with sliding-window based methods. With the time constraints incorporated in time series features, instances in X do not necessarily follow any specific time order. More details about time series feature construction can be found in the experimental setup.

With the terminologies and notations clarified, now we are ready to provide a mathematical definition of our problem settings. Given an event asset that fails at a known timestamp, with certain domain knowledge we can reasonably assume that the fault signature happens within a timestamps ahead of the failure (the *initial assumption*). Without loss of generality, we denote $X = [X_a, X_b]$, where $X_a \in \mathbb{R}^{m\times a}$ represents the instances that appear within a timestamps before the failure (*i.e.,* potentially relevant to the failure), and $X_b \in \mathbb{R}^{m\times b}$ denotes all other instances considered to be normal (usually $b \gg a$). In a fleet level multi-asset data, we can easily extend the above setting by concatenating all event (and normal) instances from different assets together in X_a (and X_b). Suppose our data consists of $k-1$ event cases from a same failure mode. We define $E = [E_a, E_b] \in \{0,1\}^{k\times n}$ as the *initial event labels* based on the initial assumption. There is exactly one "1" in each column of E, showing the initial event label of each instance. Instances in E_a have $E_a(j,i) = 1$ for some $j < k$ indicating that the i-th instance in X_a is potentially relevant to the failure in j-th event. Meanwhile, all instances in E_b have $E_b(k,i) = 1$.

Our goal is to automatically identify the feature pattern as well as the occurence timestamps of the fault signature. In other words, we expect to simultaneously select the features representing the signature and identify the subset of instances in X_a that are truly relevant to the failure.

DRAG: Directional Label Rectification in Adaptive Graph

As is discussed above, the key challenge of this problem lies in the label uncertainty. Although the labels in E_b (or most of them, if not all) are reliable, we cannot treat the problem as a classic supervised problem, since labels in E_a are mostly unreliable. However, if we use the popular definition of unsupervised or semi-supervised frameworks and simply ignore the labels in E_a, we lose the potential label information from the initial assumption. In this section we introduce a novel model to handle this situation.

There are two goals of our model: 1) select important features for fault signature; 2) rectify the labels of E_a in an automatic approach. We propose to learn a weight matrix $W \in \mathbb{R}^{m \times k}$ and a label matrix $Y = [Y_a, Y_b] \in \mathbb{R}^{k \times n}$. The values in W show the contribution of features, where the subset of features with the largest weights naturally characterize the signature. Y indicates the label probability distribution, where the instances with larger probability value on the j-th row of Y ($j < k$) can be recognized as relevant to the failure of the j-th event.

Our Framework

First, we briefly discuss the settings for feature selection. According to previous works (Liu, Ji, and Ye 2009), the feature selection task can be formulated as the following problem *w.r.t.* the feature weight matrix W:

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

where the regularization term $\|W\|_{2,1}$ imposes structured sparsity on the feature weight matrix W.

Next, we look into the problem of label rectification for instances in X_a. We start off by reviewing the idea of label propagation in graph-based semi-supervised frameworks (Zhu, Ghahramani, and Lafferty 2003). Suppose we are given a set of partially labeled data $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{m \times n}$ and an affinity matrix $\tilde{S} \in \mathbb{R}^{n \times n}$. The entries in $\tilde{S}$ indicate the pairwise affinity between corresponding instance pairs, where a larger $\tilde{S}(i,j)$ value corresponds to a higher affinity between $\mathbf{x}_i$ and $\mathbf{x}_j$. We expect to learn the label matrix $Y \in \mathbb{R}^{k \times n}$ such that instances that are close to each other tend to have similar labels. We can formulate the problem as the minimization of the quadratic energy function:

$$\min_{Y} \sum_{i,j} \tilde{S}(i,j) \|Y(:,i) - Y(:,j)\|^2. \tag{2}$$

In our problem, we can naturally treat X_b as labeled and adopt Eq. (2) to learn the labels for X_a. However, we need to emphasize three major differences in our case:

- Instances in X_a are not totally unlabeled and are usually collected from different events. Different events may have different patterns. We should integrate the information in E_a so as to avoid mislabeling a potential failure instance from one event to another event.
- We expect a "directional" label retification where labels of X_a are determined according to the affinity with X_b. We want to preclude the label propagation within X_a, since a large portion of labels in E_a could be misleading. So instead of $\tilde{S}$, we use a much smaller affinity matrix $S \in \mathbb{R}^{a \times b}$, which represents the affinity between instances in X_a and X_b, in our label rectification.
- In most previous frameworks, the affinity matrix S is predefined, thus the label propagation is sensitive and dependent on the quality of the input graph. Moreover, as S is constructed in the whole feature space, it is not appropriate when we propagate the labels in a subspace of selected features. In our model, we expect to adaptively learn an optimal graph to represent the pairwise affinity in the selected feature space.

To address the three needs listed above, we consider the following problem for label rectification:

$$\begin{aligned} \min_{S,Y} & \|S - X_a^\top X_b\|_F^2 + \mu \|Y_a - E_a\|_F^2 \\ & + \beta \sum_{i,j} S(i,j) \|Y_a(:,i) - E_b(:,j)\|^2, \\ s.t. \quad & Y\mathbf{1} = \mathbf{1}, Y_b = E_b, S^\top \mathbf{1} = \mathbf{1}, S \geq 0. \end{aligned} \tag{3}$$

It is notable that S is optimized in model (3) thus is more adaptive for label rectification. We include the term $\|S - X_a^\top X_b\|_F^2$ such that S cannot be too different from the graph on the original features, thus avoid any abnormal structure in the learned S. We involve the constraint $Y\mathbf{1} = \mathbf{1}$ to let Y encode the label probability distribution.

Combining (1) and (3), we propose to optimize the following objective function of our Directional label Rectification in Adaptive Graph (DRAG) model:

$$\begin{aligned} & \mathcal{J}(W,S,Y) \\ = & \min_{W,S,Y} \|W^\top X - Y\|_F^2 + \alpha \|W\|_{2,1} + \gamma \|S - X_a^\top X_b\|_F^2 \\ & + \beta \sum_{i,j} S(i,j) \|Y_a(:,i) - E_b(:,j)\|^2 + \mu \|Y_a - E_a\|_F^2, \\ & s.t. \quad Y\mathbf{1} = \mathbf{1}, Y_b = E_b, S^\top \mathbf{1} = \mathbf{1}, S \geq 0. \end{aligned} \tag{4}$$

Optimization Algorithm

We employ the alternating optimization (AO) algorithm for Problem (4), which iteratively minimize the problem over each variable.

Given Y and S, optimize W. The objective function *w.r.t.* W becomes

$$\min_{W} \|W^\top X - Y\|_F^2 + \alpha \|W\|_{2,1}. \tag{5}$$

The cost function in Eq. (5) is convex in W. Following (Nie et al. 2010), we can update W by taking derivative of the function *w.r.t.* W and set it to 0, which yields the following:

$$W = (XX^\top + \alpha B_W)^{-1} XY^\top, \tag{6}$$

where B_W is a diagonal matrix with the i-th diagonal element as $B_W(i,i) = \frac{1}{2\|W(i,:)\|}$.

Given Y and W, optimize S. The objective function *w.r.t.* S can be written as

$$\begin{aligned} \min_{S^\top \mathbf{1} = \mathbf{1}, S \geq 0} & \gamma \|S - X_a^\top X_b\|_F^2 \\ & + \beta \sum_{i,j} S(i,j) \|Y_a(:,i) - E_b(:,j)\|^2. \end{aligned} \tag{7}$$

Eq. (7) can be decomposed into several independent subproblems *w.r.t.* each column of S as follows:

$$\min_{S(:,j)^\top \mathbf{1} = 1, S(:,j) \geq 0} \|S(:,j) - P(:,j)\|^2, \tag{8}$$

where $P(i,j) = X_a(:,i)^\top X_b(:,j) - \frac{\beta\|Y_a(:,i)-E_b(:,j)\|^2}{2\gamma}$. We can update $S(:,j)$ in Eq. (8) according to Theorem 0.1 as follows.

Theorem 0.1. *Given any* $\mathbf{v} \in \mathbb{R}^a$*, the solution to problem*

$$\min_{\mathbf{u}^\top \mathbf{1}=1, \mathbf{u}\geq 0} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|^2 \tag{9}$$

can be formulated as $u_i = \max(v_i + \lambda^*, 0)$ *with* λ^* *satisfying* $\sum_{i=1}^{a} \max(v_i + \lambda^*, 0) = 1$.

Proof. The Lagrangian function of Problem (9) is

$$\mathcal{L}(\mathbf{u}, \lambda, \tau) = \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|^2 - \lambda(\mathbf{u}^\top \mathbf{1} - 1) - \tau^\top \mathbf{u}. \tag{10}$$

Take derivative of function (10) *w.r.t.* $\mathbf{u}$ and set it to 0, we get $\mathbf{u} = \mathbf{v} + \lambda\mathbf{1} + \tau$. According to the KKT conditions, we can derive $\tau^\top \mathbf{u} = 0$, thus $\mathbf{u} = \max(\mathbf{v} + \lambda\mathbf{1}, 0)$. Below we will discuss how to find an appropriate λ^* such that the constraint $\mathbf{u}^\top \mathbf{1} = 1$ can be guaranteed.

Without loss of generity, suppose the entries in $\mathbf{v}$ are listed in the descending order, *i.e.,* $v_1 \geq v_2 \geq \cdots \geq v_a$, and we have some $d \leq a$ satisfying

$$\begin{cases} v_d + \lambda^* \geq 0 \\ v_{d+1} + \lambda^* < 0. \end{cases} \tag{11}$$

From the constraint $\sum_{i=1}^{a} \max(v_i + \lambda^*, 0) = 1$, we can infer

$$\lambda^* = \frac{1}{d}(1 - \sum_{i=1}^{d} v_i). \tag{12}$$

Substitute Eq. (12) into Eq. (11) and we get

$$\begin{cases} \sum_{i=1}^{d}(v_i - v_d) \leq 1 \\ \sum_{i=1}^{d+1}(v_i - v_{d+1}) > 1, \end{cases}$$

thus $d = \max\{l | \sum_{i=1}^{l}(v_i - v_l) \leq 1\}$. □

Given W and S, optimize Y. With the constraint $Y_b = E_b$, we can derive that the problem of updating Y is equivalent to optimizing the following objective *w.r.t.* Y_a:

$$\begin{aligned} \min_{Y_a^\top \mathbf{1}=\mathbf{1}} & \|Y_a - W^\top X_a\|_F^2 + \mu\|Y_a - E_a\|_F^2 \\ & + \beta \sum_{i,j} S(i,j)\|Y_a(:,i) - E_b(:,j)\|^2. \end{aligned} \tag{13}$$

Let D be a diagonal matrix with the i-th diagonal element as $D(i,i) = \sum_j S(i,j)$, then the Lagrangian function of Problem (13) is

$$\begin{aligned} \mathcal{L}(Y_a, \eta) = & \|Y_a - W^\top X_a\|_F^2 + \mu\|Y_a - E_a\|_F^2 \\ & + \beta\mathrm{tr}(Y_a D Y_a^\top) - 2\beta\mathrm{tr}(Y_a S E_b^\top) + \eta^\top(Y_a^\top \mathbf{1} - \mathbf{1}). \end{aligned} \tag{14}$$

Take derivative of Eq. (14) *w.r.t.* Y_a and set it to 0 we get:

$$Y_a = (C - \frac{1}{2}\mathbf{1}\eta^\top)((\mu+1)\mathbb{I}_a + \beta D)^{-1}, \tag{15}$$

where $C = W^\top X_a + \mu E_a + \beta E_b S^\top$. Substitue the solution in Eq. (15) to the constraint $Y_a^\top \mathbf{1} = \mathbf{1}$ we get:

$$\eta = \frac{2}{k}(C^\top - (\mu+1)\mathbb{I}_a - \beta D)\mathbf{1}.$$

Thus

$$Y_a = (C - \frac{\mathbf{1}\mathbf{1}^\top}{k}C)((\mu+1)\mathbb{I}_a + \beta D)^{-1} + \frac{\mathbf{1}\mathbf{1}^\top}{k}. \tag{16}$$

We summarize the alternating optimization steps in Algorithm 1, with the time complexity $O(m^2 nT + abT\log a)$ where T is the number of iterations. We provide convergence and complexity analysis of Algorithm 1 in the supplementary material.

Algorithm 1: Optimization Algorithm for DRAG

```
Input: $X$, $E$, $\alpha$, $\beta$, $\mu$, $\gamma$.
$X = [X_a, X_b] \in \mathbb{R}^{m\times n}$ where $n$ is #instances and
$m$ is #features, while $X_a \in \mathbb{R}^{m\times a}$ contains the
intances initially labeled as event instances and
$X_b \in \mathbb{R}^{m\times b}$ are those initially labeled as normal
(usually $b \gg a$). $E = [E_a, E_b] \in \mathbb{R}^{k\times n}$ are the initial
event labels, where $E(j,i) = 1$ with $j < k$ if $\mathbf{x}_i$ is
initialized as the $j$-th event, otherwise $E(k,i) = 1$.
Output: Feature weight $W \in \mathbb{R}^{m\times k}$, learned affinity
        matrix $S \in \mathbb{R}^{a\times b}$, and instance weight
        distribution $Y \in \mathbb{R}^{k\times n}$.
Initialize $Y = E$ and $Y_a$ to be the first $a$ columns of
  $Y$;
Initialize $B_W = \mathbb{I}_m$;
while Not convergent do
    Update $W$ with Eq. (6) ;
    Update $S$ according to Theorem 0.1;
    Update $Y$ such that $Y_b = E_b$ and $Y_a$ is
  calculated with Eq. (16) ;
end while ;
```

Related Work

To encode the local manifold structure in the data, several spectral methods have been proposed in unsupervised (He and Niyogi 2004; Nie, Wang, and Huang 2014; Wang et al. 2015) and semi-supervised learning (Zhu, Ghahramani, and Lafferty 2003; Zhu 2005) frameworks. The basic assumption is that instances with larger similarity possess a higher probability to belong to the same cluster.

Several works incorporate such manifold assumption in unsupervised feature selection by learning a mapping matrix, which encodes the importance of features in preserving the local data structure (He, Cai, and Niyogi 2006; Cai, Zhang, and He 2010). Nevertheless, without considering label information, unsupervised methods are not able to detect features with discriminative power.

Table 1: Description of comparing methods.

Category	Description	Methods
#1	Update feature weights and instance weights iteratively.	Coselect (Tang and Liu 2013)
#2	First conduct classification and then feature selection.	SVM-FS (Guyon et al. 2002)
#3	First conduct anomaly detection and then feature selection.	IForest-FS (Liu, Ting, and Zhou 2012), 1SVM-FS (Schölkopf et al. 2001), GMM-FS (Mahadevan et al. 2010)
#4	First conduct feature selection and then anomaly detection.	FS-GMM

Table 2: Summary of dataset statistics: number of assets, events (k-1), features (m), initial event instances (a) and normal instances (b).

Dataset	# assets	# k	# m	# a	# b
Synthetic	1	1	90 (s=12)	50	600
Data A	29	2	464(s=464)	342	65239
Data B	1597	3	209 (s=19)	454	2810667

In contrast, (semi-)supervised feature selection methods integrate the label information (Liu, Ji, and Ye 2009; Chang et al. 2014), and some concentrates on time-series analysis (Yoon and Shahabi 2006). However, these methods require the initial label information to be trustworthy, which cannot be guaranteed in our problem setting. If we employ the previous methods to our rough initial label setting, they may fail due to the low quality of positive labels, and the learned features will represent an incorrect data structure thus be misleading. Moreover, previous methods use a fixed affinity graph which is not suitable to represent the data structure in the selected feature subspace.

As illustrated a prior, our work is unique from three perspectives: 1) it simultaneously outputs instance weights (for X_a) and feature weights while requires very rough initial label information; 2) it uncovers the consistent feature pattern across failure events while also considers the pattern variety among different failure types; 3) it adaptively learns the optimal graph structure for label rectification.

There are a number of recent works concentrating on time series anomaly detection (Batal et al. 2012; Zhou et al. 2016; Cheng et al. 2016), some of which are built on auto-regression (e.g. ARIMA, VAR).

(Batal et al. 2012) converts time series into intervals of temporal abstractions and detect discriminative subsequence patterns according to predefined abstract states. This method is sensitive to the setting of abstract states and cannot handle high dimensional data. On the contrary, DRAG directly learns the weight of features via optimization without any complex conversion and is well applicable to large number of features. (Cheng et al. 2016) proposed to rank causal anomalies according to the vanishing correlations on sensor graph. This method assumes the existence of an invariant network on sensors/features, which may not hold in real world applications where the underlying data distribution varies over time. Furthermore, this method is only applicable for single asset analysis. Comparatively, our DRAG model allows dynamic patterns in normal set (X_b) and is able to deal with multiple assets (data sources).

Experiments

Experimental Setup

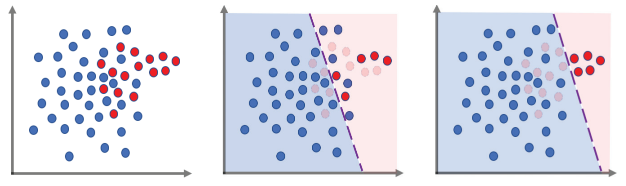
Currently, there are not many studies on label rectifying with feature selection. To the best of our knowledge, we compare with four categories of related methods and summarize their description in Table 1.

All experiments are conducted on a standard laptop (Quadcore Intel i7 CPU@2.7 GHz). We use linear kernel for all SVM models (implementation is based on libsvm). Note that all hyper-parameters of the comparing methods are set according to the reported best or tuned to get the best performance. Specifically, 1) for GMM models, we set the number of components as 6; 2) for Coselect, we set $\alpha = 0.1$, $\beta = 0.01$, $\lambda = 0.3$, and construct the linking graph R with the following strategy: if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same event in X_a then $R(i,j) = 1$, otherwise zero; 3) for DRAG, the relative magnitude relation of entries in Y_a and W is relatively stable *w.r.t.* parameters, so in Algorithm 1 we set the following by default: $\alpha = m/k$, $\beta = 1/b$, $\mu = s/2a$ and $\gamma = 1$.

We use both synthetic and real world datasets in our evaluation. The synthetic dataset [1] is constructed on the basis of Yahoo Benchmark Dataset for Time Series Anomaly Detection (Laptev, Amizadeh, and Youssef Mar 25 2015). We randomly select 12 features from the A2Benchmark dataset and detrend them. For simplity, we assume that all the instances are collected from one single asset and there is only one failure event at the last timestamp. We initially set the number of event instances (a, number of instances in X_a) to be 50. To make the task more challenging, we add a few anomalies at random timestamps prior to the last 100 instances as noise. The target anomaly signature happens at timestamps 621 to 624 on feature L and K. With the ground truth available, we can test whether the methods are able to distinguish the truly relevant instances and features by calculating the F1 score of the positive (event) class on the selected features.

For real world data, we use two industrial fleet-level datasets with different failure modes. Statistics of the datasets are presented in Table 2. In real data, the size of initial event instances (a) should be determined based on domain knowledge and are application-specific. In our datasets, we let a equal to the number of instances within 90 days before each failure event. Since the ground truth of key features and occurence timestamps of the fault signature in

[1]https://goo.gl/16lM7h



(a) Initial dataset (b) Worse instance selection (c) Better instance selection

Figure 2: Illustration of our evaluation metric for real data. Given the initial dataset (a) which consists of X_a (red) and X_b (blue), the goal is to find the top-q instances from X_a and the top-p features from the multivariate input, such that a clear separation exists between the selected q instances and X_b on the top-p feature. Here we evaluate each method using the F1-score in linear SVM classification, which naturally indicates how "well" the selected instances are spereated from X_b. In this example, with the same selected features, (c) indicates a better instance selection than (b). We can evaluate the quality of feature selection in a similar way.

real data is usually not available, we cannot directly assess the methods as in standard classification problems. In the following, we design an evaluation metric for real data.

Evaluation metric for real data: As we recall, the goal of the task is to uncover the feature pattern of the signature and identify the instances in X_a that are truly relevant to the failure. A common assumption in anomaly detection is that failure instances and normal instances conform to different patterns, *i.e.,* different distributions (Chandola, Banerjee, and Kumar 2009). So in other words, we expect to identify top-q instances in X_a and top-p features, such that the top-q instances are clearly separable from X_b in the top-p feature space. Since $b \gg a$, we can use F1-score to as a quality check of the separation. Practically, linear separation has better interpretability and is prefereable for domain experts, therefore, we use linear SVM to evaluate the methods. That is, we use F1-score of the positive (event) class to measure whether the selected q instances in X_a and instances in X_b are linearly separable on the selected features. We illustrate our evaluation metric in Figure 2.

In the real world analysis we usually focus on a limited number of features to narrow down the root cause. Here we fix $p = 3$ and run experiments with $q \in [3, 20]$. Similarly, we test the feature stability of the comparing methods with $p \in [3, 15]$ and fixing $q = 3$. To test the stability of the comparing methods, we use down-sampling on X_b such that $b = 5a$ and generate sketches of the dataset. We run each algorithm 10 times and report the average performance.

Time-series feature construction: We adopt sliding-window-based methods to build time series features, such that the temporal information can be incorporated into the data. Given the window size ρ, each operation in the construction is an algorithm $g : \mathbb{R}^\rho \to \mathbb{R}$, that takes a time series window $(x'_1, x'_2, ..., x'_\rho)$ from one feature of a certain asset as input, and outputs a single real number. We apply two different operations in the experiments: autocorrelation (of univariate raw sensor) and correlation (of pairwise raw sensors). There are also other ways of time series feature construction in previous literature (Fulcher and Jones 2014), *e.g.,* EMD, and FFT.

We first smooth the data using moving average in a window of length ρ and then conduct time-series feature construction. We include both raw sensor record and time series features, and the total number of features is $m = 2r + r(r-1)/2$ where r is the number of raw sensors. In the experiments, we set $\rho = 20$ and scale each feature of X to $[0, 1]$. Specially, in Data A, since s is large and the information in raw sensor record is enough for prediction, it is not necessary to construct time-series features for this dataset.

Experiments on Synthetic Data

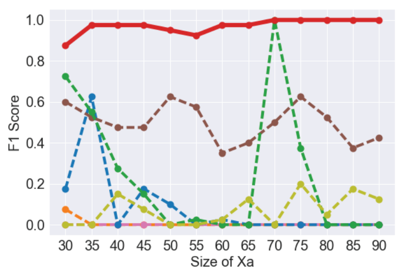
In this subsection, we conduct experiments on the synthetic data as a sanity check. We first test the feature and instance stability using the evaluation metric discussed in previous subsection. As we can see from the results in Fig. 3(b) and 3(c), DRAG consistently outperforms all comparing methods in both settings. From Fig. 3(b) we can find that the superiority of DRAG is more obvious when the number of selected features is low, which indicates that DRAG is able to correctly target the root cause with only a few selected features. As p increases, the comparing methods gain improvements in the performance as they gradually include root cause features. In Fig. 3(c), we can see a clear downward performance trend with increasing q in the comparing methods. This is because more failure-irrelevant instances are misclassified to be positive as the selected instances become more. However, DRAG still performs well with large q, which illustrates the strong robustness of our model.

Moreover, since the ground truth of fault signature in synthetic data is available, we also test the stability of the methodsx when we use different a value in the initial event label assignment. Fixing the real number of failure instances as 4, we vary the size of X_a in the range of $\{30, 35, \ldots, 90\}$ and evaluate the methods by comparing the selected instances against the ground truth. From the results in Fig. 3(a), we can notice that the performance of DRAG is fairly robust to the change of a value. As a increases, more noise and misleading information are involved in the data, thus the comparing methods are impacted more or less. However, the performance of DRAG is fairly stable when the size of X_a varies. This is important to ensure that DRAG performs well even without an accurate estimate of a in practice.

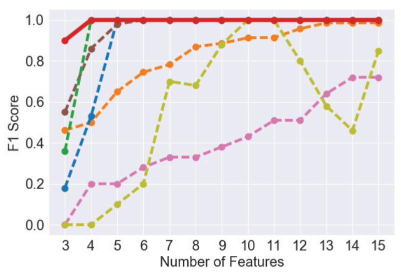
Experiments on Real World Data

We summarize the comparison results on real world data in Fig. 5 and 6. It is worth mentioning that in the real failure analysis involving mutiple events, since the events come from the same failure mode, we expect to identify the common subset of features associated with the failure across events. Whereas, the importance order of the selected features in each event can be different. As a consequence, in the experiments we first select features across all events and then evaluate the performance *w.r.t.* each event separately.

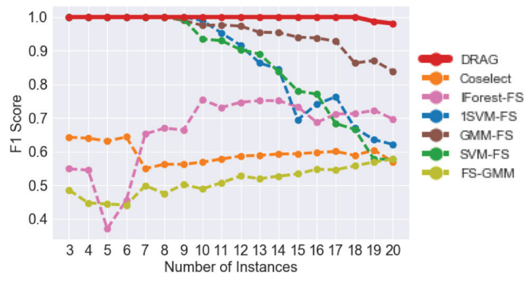
From the comparison in Fig. 5 and 6, we find that DRAG performs well. In most cases DRAG performs better than all



(a) Varying X_a size



(b) Feature stability



(c) Instance stability

Figure 3: Results on synthetic data. (a) F1-score againt the ground truth when X_a size varies. (b) feature stability with different number of selected features p while the number of instances $q = 3$; (c) instance stability with different q while fixing $p = 3$.

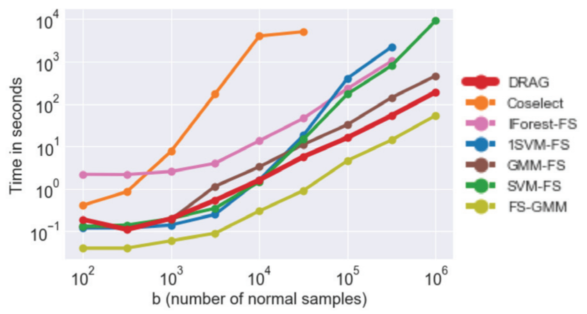


Figure 4: Running time comparison on Data B with varying size of X_b.

counterparts for at least 15%. In the results of feature stability, methods in category 3 of Table 2 tend to select similar top 3 instances, so they are fairly similar in the performance for both Data A and Data B. As for the instance stabiity comparison, since the number of selected features is small, we can notice that the methods in category 1, 2 and 3 of Table 2 sometimes fail miserably due to the bad quality of the selected features. This is because these methods select features and instances separately, and any mistake in one step can have a huge impact on the following step. On the contrary, DRAG and Coselect conduct feature and instance selection simultaneously thus perform much better. In DRAG, we learn an adaptive and optimal graph in each iteration, which guarantees the quality of the affinity matrix hence the performance.

Running Time Comparison

In this subsection, we report the running time of each method with different data scales. We run the experiments on Data B with varying size of X_b and plot the running time comparison in Fig. 4. We report the real time of the training process, including both of the instance and feature selection steps. We can notice that Coselect is not scalable since this method requires the full graph constructed on the whole data. In contrary, DRAG uses a much smaller affinity matrix constructed between X_a and X_b, thus is much more efficient in implementation. FS-GMM is the fastest method because it conducts GMM after selecting the top 3 features. From the results we can find that DRAG is among the fastest methods and is very well scalable to large data.

Conclusion

In this paper, we put forward a directional label rectification model with adaptive graph for failure analysis in multivariate time-series data. To deal with the label uncertainty problem, we proposed to dynamically rectify the initial event labels according to the affinity with normal instances. Unlike previous methods using a fixed graph as input, we adaptively learned an optimal affinity matrix to represent the graph structure in the subspace of selected features. The experimental results verified that our model is effective in simultaneously selecting failure-relevant features and instances. The results on scalability indicates that our model is very well suited for large data.

References

Batal, I.; Fradkin, D.; Harrison, J.; Moerchen, F.; and Hauskrecht, M. 2012. Mining recent temporal patterns for event detection in multivariate time series data. In *KDD*, 280–288. ACM.

Cai, D.; Zhang, C.; and He, X. 2010. Unsupervised feature selection for multi-cluster data. In *KDD*, 333–342. ACM.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM Comput. Surv.* 41(3):15:1–15:58.

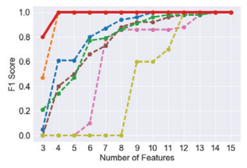
Chang, X.; Nie, F.; Yang, Y.; and Huang, H. 2014. A convex formulation for semi-supervised multi-label feature selection. In *AAAI*, 1171–1177.

Cheng, W.; Zhang, K.; Chen, H.; Jiang, G.; Chen, Z.; and Wang, W. 2016. Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations. In *KDD*. ACM.

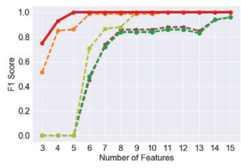
Fulcher, B. D., and Jones, N. S. 2014. Highly comparative feature-based time-series classification. *IEEE Trans. Knowl. Data Eng.* 26(12):3026–3037.

Guyon, I.; Weston, J.; Barnhill, S.; and Vapnik, V. 2002. Gene selection for cancer classification using support vector machines. *Mach. Learn.* 46(1):389–422.

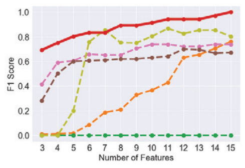
He, X., and Niyogi, P. 2004. Locality preserving projections. In *NIPS*, 153–160.



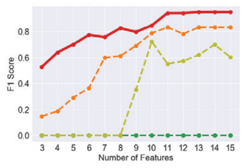
(a) Feature stability: Data A, 1st event



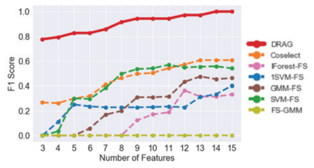
(b) Feature stability: Data A, 2nd event



(c) Feature stability: Data B, 1st event

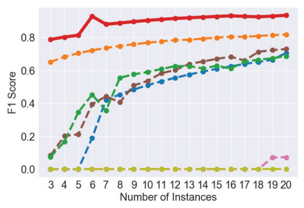


(d) Feature stability: Data B, 2nd event

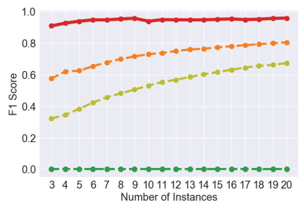


(e) Feature stability: Data B, 3rd event

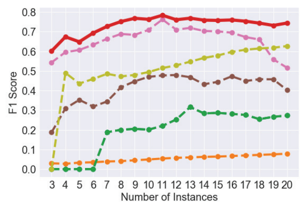
Figure 5: Experimental results on two real world datasets with different number of selected features p. Feature stability is evaluated by fixing the number of selected instance $q = 3$.



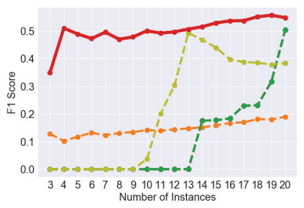
(a) Instance stability: Data A, 1st event



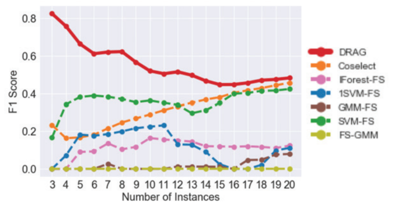
(b) Instance stability: Data A, 2nd event



(c) Instance stability: Data B, 1st event



(d) Instance stability: Data B, 2nd event



(e) Instance stability: Data B, 3rd event

Figure 6: Experimental results on two real world datasets with different number of selected instances q. Instance stability is evaluated with number of selected features $p = 3$.

He, X.; Cai, D.; and Niyogi, P. 2006. Laplacian score for feature selection. In *Advances in neural information processing systems*, 507–514.

Laptev, N.; Amizadeh, S.; and Youssef, B. Mar 25, 2015. Announcing a benchmark dataset for time series anomaly detection. In *https://research.yahoo.com/news/announcing-benchmark-dataset-time-series-anomaly-detection*.

Liu, J.; Ji, S.; and Ye, J. 2009. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *UAI*, 339–348. AUAI Press.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2012. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data* 6(1):3:1–3:39.

Mahadevan, V.; Li, W.; Bhalodia, V.; and Vasconcelos, N. 2010. Anomaly detection in crowded scenes. In *CVPR*, 1975–1981. IEEE.

Nie, F.; Huang, H.; Cai, X.; and Ding, C. H. 2010. Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization. In *NIPS*, 1813–1821.

Nie, F.; Wang, X.; and Huang, H. 2014. Clustering and projected clustering with adaptive neighbors. In *KDD*, 977–986. ACM.

Schölkopf, B.; Platt, J. C.; Shawe-Taylor, J.; Smola, A. J.; and Williamson, R. C. 2001. Estimating the support of a high-dimensional distribution. *Neural Comput.* 13(7):1443–1471.

Tang, J., and Liu, H. 2013. Coselect: Feature selection with instance selection for social media data. In *SDM*, 695–703. SIAM J. Appl. Math.

Wang, X.; Liu, Y.; Nie, F.; and Huang, H. 2015. Discriminative unsupervised dimensionality reduction. In *IJCAI*, 3925–3931.

Yoon, H., and Shahabi, C. 2006. Feature subset selection on multivariate time series with extremely large spatial features. In *ICDM Workshops*.

Zhou, D.; He, J.; Cao, Y.; and Seo, J.-s. 2016. Bi-level rare temporal pattern detection. In *ICDM*.

Zhu, X.; Ghahramani, Z.; and Lafferty, J. D. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 912–919.

Zhu, X. 2005. Semi-supervised learning literature survey.