Collaborative Anomaly Detection on Blockchain from Noisy Sensor Data

Tsuyoshi Idé
IBM Research, T. J. Watson Research Center
1101 Kitchawan Road, Yorktown Heights, NY 10598, USA
tide@us.ibm.com

***Abstract*—This paper proposes a framework for collaborative anomaly detection on Blockchain. Taking condition-based management of industrial asset as a practical example, we extend the notion of Smart Contract, which has been implicitly assumed to be deterministic, to be able to handle noisy sensor data. By formalizing the task of collaborative anomaly detection as that of multi-task probabilistic dictionary learning, we show that major technical issues of validation, consensus building, and data privacy are naturally addressed within a statistical machine learning algorithm. We envision Blockchain as a platform for collaborative learning rather than just a traceable, immutable, and decentralized data management system, suggesting the direction towards "Blockchain 3.0".**

***Index Terms*—Blockchain, collaborative learning, multi-task learning, anomaly detection, condition-based management**

I. Introduction

The major commercial success of cryptocurrencies has made Blockchain one of the hottest topics in the information technology industry. Scholars, practitioners, and even private investors are actively seeking new applications of Blockchain these days. Although Blockchain was originally proposed as a decentralized protocol for value exchange [1], it has evolved to a distributed ledger platform by extending the original protocol to be able to handle more general transactions beyond money transfer in the form of Smart Contract [2]. Although the generalized version of Blockchain, which is sometimes called "Blockchain 2.0" [3], [4], has opened a new door to a vast array of application fields, it, in turn, has brought in a few major technical challenges, which are becoming major concerns especially in closed or "permissioned" Blockchain networks [5].

The first challenge is how to validate business transactions. Unlike money transfer, in which validating a transaction is readily done by checking the identity of the sender and the receiver and the balance of the account, writing down a policy to validate a general business logic is not straightforward. This is especially true when a part of the business process is done outside the Blockchain network. Examples include food traceability networks [6]. Using the hash pointers of the blockchain, you can guarantee a stored record has never been altered, but "what would stop bad actors in the supply chain from fudging them in the first place" [7]? What we need here is a smart mechanism that automatically down-weights less trustworthy entries.

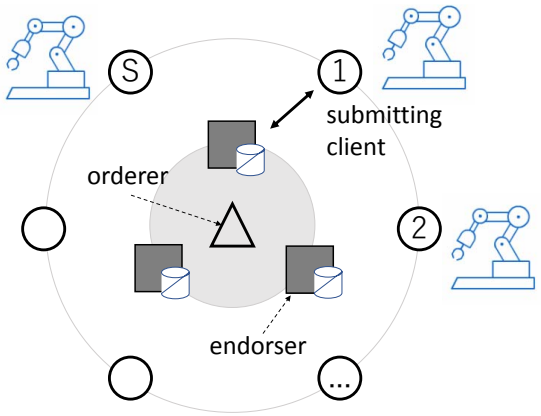

Fig. 1. Illustration of proposed condition-based management system on Blockchain.

The second challenge is how to build a consensus over transaction records. The original Bitcoin protocol relies on the assumption that the Blockchain network is *stochastic*: The network is assumed to have statistically independent validators (or miners) who are incentivized to compete each other for a reward [1]. This assumption allows computing the probability that attackers will successfully falsify the latest transaction records. Nakamoto [1] showed that the probability is negligible as long as honest nodes dominate the network. However, it is evident that the assumption of stochastic competition does not hold in permissioned networks at all. Being away from the ideology of decentralized management, permissioned networks employ a partially centralized approach by having pre-selected (*i.e.* trusted) endorsers validate transactions. The Byzantine fault tolerance mechanism or its variant is typically used to get a consensus among the endorsers [8].

In this paper, we shed a new light on these challenges in the context of internet-of-things (IoT). Specifically, we consider the task of condition-based management (CbM) [9] of industrial assets such as plasma etching tools in semiconductor manufacturing and coal mining drilling machines. Although CbM includes a variety of business tasks in general, we focus on the task of anomaly detection, which is to detect unusual behaviors of the assets from their sensor data before getting actually broken or suffering unplanned service interruptions.

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00024

To address the well-known issue of lack of sufficient amount of anomalous samples for model training, Blockchain as an information sharing platform has a lot of potential in anomaly detection. However, it has not been clear how CbM is made possible on Blockchain. Specifically, the major goal of CbM is to get a set of feasible business rules learned from raw sensor signals with random noise. In such a case, the very notion of validation and consensus require careful reconsiderations because most of the Blockchain implementations assume deterministic business transactions [10].

We argue that there are at least three requirements for a practical Blockchain-based CbM platform. First, related to the first challenge above, it should be able to systematically handle statistical outliers and random fluctuations of data. Second, related to the second challenge, it should be able to build a network-wide consensus while taking care of the individuality of the participant nodes. Third, it should preserve data privacy.

To meet these requirements, we formalize anomaly detection as *collaborative dictionary learning*. In the proposed Blockchain network as illustrated in Fig. 1, common knowledge or *consensus* is represented as a dictionary of patterns. To collaboratively learn the dictionary, the participant nodes ("Submitting Clients") iteratively send the Endorsers intermediate statistics derived from raw data. The Endorsers update the dictionary as a Smart Contract based on a weighted average over the shared intermediate statistics. In this framework, for the first requirement, less trustworthy samples are automatically down-weighted through a statistical machine learning algorithm. For the second requirement, the consensus is naturally achieved as a statistically weighted average, which is also part of the statistical learning procedure. For the third requirement, data privacy is preserved by sharing only aggregated intermediate statistics. To the best of our knowledge, this is the first work to formalize CbM as collaborative anomaly detection on Blockchain. This paper is aimed at making a significant step forward to fill the gap between the reality of CbM and speculative discussions on Blockchain for IoT.

II. Related work

Prior work related to the present study can be categorized into two groups: Blockchain-based IoT platforms and collaborative anomaly detection methods.

Regarding the first category, the importance of Blockchain technologies in IoT was first made widely recognized by IBM's report titled "Device Democracy" [11], which argued that Blockchain would be an ideal platform to manage billions of connected devices in terms of cost, security, and future-proof-ness. Many application scenarios have been discussed since then, such as product provenance tracking [12], IoT device management [13], connected cars [14], location-based recommendation [15], and smart home [16] although most of them are still at a conceptual stage.

In those application scenarios, a crucial but less explicitly discussed technical issue is how to design Smart Contracts. In the connected car scenario, for example, a Blockchain-based automated maintenance system would work if there were a complete set of rules to detect indications of faults. However, detecting fault indications itself is known to be very challenging, and this is indeed the main reason why CbM has been treated as a holy grail in the manufacturing industries. This work clearly differs from those existing work in that the Smart Contract in our system is naturally defined as part of a collaborative machine learning algorithm.

Regarding the second category, collaborative anomaly detection in this paper refers to machine learning algorithms that perform anomaly detection through multi-task learning [17]. Note that distributed (or federated) learning in the standard sense is different from multi-task learning in that the latter typically aims at learning a single model while the former aims at learning multiple models as many as the number of data sources. For example, in Fig. 1, there are S different industrial assets. The goal is to learn S anomaly detection models collaboratively.

Although multi-task learning has been an actively studied topic in data mining and machine learning communities [17], there are not many studies that use multi-task learning specifically for anomaly detection. Most of them use one-class support vector machines (OCSVM) [18], [19] but it is not common to use OCSVM in industrial IoT applications due to its sensitivity to the choice of hyper-parameters and limited interpretability. Recently, an anomaly detection method in the multi-task setting to handle the multi-modality of noisy sensor signals has been proposed [20]. It shares some of our motivations, but it completely lacks the context of Blockchain. Also, it uses advanced mathematical techniques including ℓ_0-regularized optimization, making implementation on Blockchain challenging. To the best of our knowledge, this is the first work on collaborative anomaly detection on Blockchain.

III. Problem setting

This section overviews the proposed Blockchain-based collaborative anomaly detection system.

A. Data set

Figure 1 shows an overview of the proposed system, where we follow the taxonomy of Hyperledger Fabric [5]. There are three major players in the system. The *submitting client* (or simply *client*) corresponds to an industrial asset being monitored for anomaly detection. There are S clients in the network indexed by $s = 1, 2, \ldots, S$. The clients are assumed to be of the same type of equipment, but they can behave quite differently, depending on operational conditions. Each client privately has a data set as a result of repeated measurements. For the s-th client, the data set is written as

$$\mathcal{D}^s = \{\boldsymbol{x}^{s(1)}, \ldots, \boldsymbol{x}^{s(N^s)}\}, \tag{1}$$

where $\boldsymbol{x}^{s(n)} \in \mathbb{R}^M$ denotes the n-th sample of the M-dimensional measurement. We assume that $\mathcal{D}^s$ has been *centered* to have zero mean. In the CbM scenario, measurements are typically performed sequentially and the index n specifies a time index. All the clients have assumed to have the same

measurement system. For example, if the first sensor measures the temperature of a part in the fifth client, the first dimension of, *e.g.*, $\boldsymbol{x}^{s=10}$ must be the temperature of the same part of the 10th client. The total number of samples N^s can differ among the clients. Some clients may have fewer samples and others may have more. Also, in our unsupervised anomaly detection setting, $\mathcal{D}^s$ is assumed to have taken under a normal condition. Outliers may exist in the data, but at least $\mathcal{D}^s$ has to be dominated by normal samples.

Although we will use CbM as a running example, it is clear that our model applies to other applications, such as medical record analysis.

B. Anomaly score

Based on the data set $\mathcal{D} \equiv \{\mathcal{D}^1, \ldots, \mathcal{D}^S\}$, the clients collaboratively learn an anomaly detection model, which computes a real number called the *anomaly score* as a function of a new sample and model parameters learned from $\mathcal{D}$. The standard choice for the anomaly score function is the logarithmic loss [21]. For the s-th client, it is formally written as

$$a^s(\boldsymbol{x}^s \mid \boldsymbol{\theta}_{\text{gl.}}, \boldsymbol{\theta}_{\text{lo.}}) = -\ln p(\boldsymbol{x}^s \mid \boldsymbol{\theta}_{\text{gl.}}, \boldsymbol{\theta}_{\text{lo.}}), \tag{2}$$

where $p(\boldsymbol{x}^s \mid \cdot, \cdot)$ is the (multivariate) probability density function for one sensor measurement, $\boldsymbol{\theta}_{\text{gl.}}$ symbolically denotes the global model parameters shared by all the clients, and $\boldsymbol{\theta}_{\text{lo.}}$ denotes the local parameter kept within each client (see Sec. IV for the definition). Since $\mathcal{D}^s$ is from normal state operations, the density function p can be viewed as a summary of the normal state.

In the Blockchain terminology, the global parameter $\boldsymbol{\theta}_{\text{gl.}}$ is a mathematical representation of *consensus*. As described in the next section, $\boldsymbol{\theta}_{\text{gl.}}$ consists of a set of parameters defining a probabilistic mixture model.

C. Network participants

As shown in Fig. 1, there are three types of participants in the network: clients, endorsers, and order. The goal of the proposed Blockchain-based system is to collaboratively learn $\boldsymbol{\theta}_{\text{gl.}}$ and $\boldsymbol{\theta}_{\text{lo.}}$ through iterative communication among clients, endorsers, and orderer.

As in ordinary Blockchain networks, a client communicates with an *endorser* node to send intermediate statistics derived from raw data $\mathcal{D}^s$ and request to send back the latest global parameter $\boldsymbol{\theta}_{\text{gl.}}$. The *orderer* checks the time stamps and possibly job IDs to make sure that the provided intermediate statistics belong to the same computation round (see Sec. IV for the algorithmic detail). Once the orderer confirmed that all the statistics are in place, or predefined time has passed, the orderer broadcasts a message that the block is confirmed. This sequence is repeated until convergence.

Although only one orderer is shown in the figure for simplicity, there can be multiple orderers. Similarly, we have omitted details such as Certificate Authorities. For details, refer to the documentation of major Blockchain implementations such as Hyperledger Fabric [5].

IV. Collaborative dictionary learning from noisy multivariate real-valued data

This section presents the mathematical details of Smart Contract executed by the endorsors and partly by the clients. Our goal is to give an algorithm to compute Eq. (2). For industrial applications, it is important to make sure that the distribution $p(\boldsymbol{x}^s \mid \boldsymbol{\theta}_{\text{gl.}}, \boldsymbol{\theta}_{\text{lo.}})$ captures various different patterns in the training data $\mathcal{D}$ while preserving high interpretability. To achieve an optimal balance, we employ a multi-task Gaussian mixture model with a sparsity-enforcing prior. We first focus on describing the core machine learning algorithm with less attention to the Blockchain terminology, then we paraphrase the algorithm in the light of Smart Contract.

A. Observation model and priors

First, we define the observation model of the s-th client by

$$p(\boldsymbol{x}^s \mid \boldsymbol{z}^s, \boldsymbol{\mu}, \Lambda) \equiv \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}^s \mid \boldsymbol{\mu}_k, (\Lambda_k)^{-1})^{z_k^s}, \tag{3}$$

where $\mathcal{N}(\boldsymbol{x}^s \mid \boldsymbol{\mu}_k, (\Lambda_k)^{-1})$ is the Gaussian probability density with the mean $\boldsymbol{\mu}_k$ and the precision matrix Λ_k. On the l.h.s. (left hand side), $\boldsymbol{\mu}$ and Λ are collective notations representing $\{\boldsymbol{\mu}_k\}$ and $\{\Lambda_k\}$, respectively. Also, $\boldsymbol{z}^s$ is the indicator variable of cluster assignment. As usual, $z_k^s \in \{0, 1\}$ for all s, and $\sum_{k=1}^{K} z_k^s = 1$. As explained later, by giving a large enough K and letting the algorithm prune irrelevant clusters, we can automatically get an optimal number of clusters as long as reasonable initialization is used. Also, notice that the parameters $\{\boldsymbol{\mu}_k\}$ and $\{\Lambda_k\}$ do not have the index of s. This means that they are shared by all the clients as a result of collaborative learning. We call $\{\boldsymbol{\mu}_k, \Lambda_k\}_{k=1}^K$ the *dictionary* because the list should cover major distributional patterns in the training data.

Following [20], we use the Gauss-Laplace prior on $(\boldsymbol{\mu}_k, \Lambda_k)$ and the categorical distribution on $\boldsymbol{z}$:

$$p(\boldsymbol{\mu}_k, \Lambda_k) \propto \mathcal{N}(\boldsymbol{\mu}_k | \mathbf{0}, (\lambda_0 \Lambda_k)^{-1}) \exp\left(-\frac{\rho}{2}\|\Lambda_k\|_1\right) \tag{4}$$

$$p(\boldsymbol{z}^s | \boldsymbol{\pi}^s) = \prod_{k=1}^{K} (\pi_k^s)^{z_k^s} \ \text{ s.t. } \ \sum_{k=1}^{K} \pi_k^s = 1, \ \pi_k^s \geq 0, \tag{5}$$

where $\|\Lambda_k\|_1$ is the absolute sum of all the entries of Λ_k. The parameter $\boldsymbol{\pi}^s$ is determined as a part of the model while ρ, λ_0 are given constants.

Our basic strategy for model learning is maximum a posteriori (MAP). In this case, since $\mathsf{Z} \equiv \{\boldsymbol{z}^s\}$ is unobserved latent variables, MAP estimation seeks the maximizer of the log marginalized likelihood:

$$L(\Lambda, \boldsymbol{\mu}, \boldsymbol{\pi}) \equiv \ln \sum_{\mathsf{Z}} \prod_{k=1}^{K} p(\boldsymbol{\mu}_k, \Lambda_k) \times \prod_{s=1}^{S} \prod_{n=1}^{N^s} p(\boldsymbol{z}^{s(n)} | \boldsymbol{\pi}^s) p(\boldsymbol{x}^{s(n)} \mid \boldsymbol{z}^{s(n)}, \boldsymbol{\mu}, \Lambda), \tag{6}$$

where $\boldsymbol{z}^{s(n)}$ is the indicator variable for the n-th sample of the s-th client and $\boldsymbol{\pi} \equiv \{\boldsymbol{\pi}^s\}$.

B. EM algorithm for collaborative density estimation

Because of the summation, direct maximization of Eq. (6) is hard. Luckily, we can derive a collaborative version of EM (expectation-maximization) iteration by slightly modifying the standard solution for the Gaussian mixture [22]. To derive the lower bound through Jensen's inequality, we introduce a distribution over Z,

$$Q(\mathsf{Z}) = \prod_{s=1}^{S} \prod_{n=1}^{N^s} \prod_{k=1}^{K} (r_k^{s(n)})^{z_k^{s(n)}}, \tag{7}$$

where $r_k^{s(n)} \geq 0$ and $\sum_{k=1}^{K} r_k^{s(n)} = 1$ for all s, n. Now Jensen's inequality leads to

$$\begin{aligned} L(\Lambda, \boldsymbol{\mu}, \boldsymbol{\pi}) \geq & L_0(\Lambda, \boldsymbol{\mu}, \boldsymbol{\pi}) \equiv \sum_{n=1}^{N^s} \sum_{s=1}^{S} \sum_{k=1}^{K} r_k^{s(n)} \ln \left(\pi_k^s \mathcal{N}_k^{s(n)} \right) \\ & + \sum_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}^k | \mathbf{0}, (\lambda_0 \Lambda_k)^{-1}) - \frac{\rho}{2} \sum_{k=1}^{K} \|\Lambda_k\|_1 \end{aligned} \tag{8}$$

up to an unimportant constant, where $\mathcal{N}_k^{s(n)}$ is a shorthand notation of $\mathcal{N}(\boldsymbol{x}^{s(n)} | \boldsymbol{\mu}_k, (\Lambda_k)^{-1})$. Apart from the summation over s and the Laplace prior, this is the same as the lower bound of the standard Gaussian mixture [22].

The EM iteration proceeds as follows: Given $\{r_k^{s(n)}\}$ as a numerical value, we find the maximizer of $L_0(\Lambda, \boldsymbol{\mu}, \boldsymbol{\pi})$. Then, given $\Lambda, \boldsymbol{\mu}, \boldsymbol{\pi}$ as a numerical value, L_0 is maximized with respect to $\{r_k^{s(n)}\}$. The former is called the M (maximization) step and the latter called the E ("expectation") step for a historical reason.

In the E step, by taking the derivative with respect to $r_k^{s(n)}$ under the sum-to-one constraint, we readily get

$$r_k^{s(n)} = \frac{\pi_k^s \mathcal{N}_k^{s(n)}}{\sum_{l=1}^{k} \pi_l^s \mathcal{N}_l^{s(n)}}. \tag{9}$$

In the M step, first, by taking the derivative with respect to π_k^s under the sum-to-one constraint, we again have

$$\pi_k^s = \frac{\sum_{n=1}^{N^s} r_k^{s(n)}}{\sum_{n=1}^{N^s} \sum_{l=1}^{K} r_l^{s(n)}}. \tag{10}$$

Second, by taking the derivative with respect to $\boldsymbol{\mu}_k$, we have

$$N_k \equiv \sum_{n=1}^{N^s} \sum_{s=1}^{S} r_k^{s(n)} \tag{11}$$

$$\boldsymbol{\mu}_k = \frac{1}{\lambda_0 + N_k} \sum_{s=1}^{S} \sum_{n=1}^{N^s} r_k^{s(n)} \boldsymbol{x}^{s(n)}. \tag{12}$$

Finally, maximizing L_0 with respect to Λ_k is equivalent to

$$\max_{\Lambda_k} \left\{ \ln \det \Lambda_k - \mathrm{Tr}(\Lambda_k \Sigma_k) - \frac{\rho}{N_k} \|\Lambda_k\|_1 \right\}, \tag{13}$$

where $\det$ denotes the matrix determinant and

$$\Sigma_k \equiv \frac{1}{N_k} \sum_{s=1}^{S} \sum_{n=1}^{N^s} r_k^{s(n)} \boldsymbol{x}^{s(n)} \boldsymbol{x}^{s(n)\top} - \boldsymbol{\mu}_k \boldsymbol{\mu}_k^\top, \tag{14}$$

where $\top$ denotes the transpose. The optimization problem Eq. (13) is called the graphical lasso and an efficient algorithm is available. See [23] for the detail. Thanks to the ℓ_1 regularizer, the solution of Eq. (13) is known to give a sparse matrix. Since the precision matrix has one-to-one correspondence to the Gaussian graphical model [24], the resulting sparse precision matrices give readily interpretable information on the interdependency between the variables. In other words, the learned dictionary will contain major prototypes of dependency patterns in addition to prototype vectors representing the cluster centers.

C. Protocol for parameter updates

Now that we have derived the algorithm for parameter updates, let us translate the algorithm into the Blockchain language. The key consideration here is privacy preservation of training data: we need to establish a protocol to get the learning going without sharing the raw data with the endorser side. Fortunately, it is possible to rewrite the algorithm to meet the privacy requirement. The proposed protocol alternatingly performs two major procedures of `c_update` ("client update") and `e_update` ("endorser update") until the orderer declares convergence.

On the client side, given the latest dictionary $\{\boldsymbol{\mu}_k, \Lambda_k\}_{k=1}^K$, each client performs `c_update` to get the local parameters

$$\boldsymbol{\theta}_{\mathrm{lo.}}^s \equiv \{N_k^s, \boldsymbol{m}_k^s, \mathsf{C}_k^s, \pi_k^s\}_{k=1}^K \tag{15}$$

as the intermediate statistics. Specifically, the s-th client first computes new $\{r_k^{s(n)}\}_{k=1}^K$ with Eq. (9). Then it updates intermediate statistics as

$$N_k^s \leftarrow \sum_{n=1}^{N^s} r_k^{s(n)}, \tag{16}$$

$$\boldsymbol{m}_k^s \leftarrow \sum_{n=1}^{N^s} r_k^{s(n)} \boldsymbol{x}^{s(n)}, \tag{17}$$

$$\mathsf{C}_k^s \leftarrow \sum_{n=1}^{N^s} r_k^{s(n)} \boldsymbol{x}^{s(n)} \boldsymbol{x}^{s(n)\top}, \tag{18}$$

$$\pi_k^s \leftarrow \frac{N_k^s}{\sum_{l=1}^{K} N_l^s}, \tag{19}$$

for $k = 1, \ldots, K$. Notice that these are *aggregated* statistics that do not convey any sample-wise information. For some additional privacy considerations, see Sec. IV-D.

The updated $\boldsymbol{\theta}_{\mathrm{lo.}}^s$ together with relevant attributes such as the time stamp and the iteration count is shared with the endorsers and the orderer. The orderer checks the attributes to bundle the transactions of the same iteration round. When all the S local parameters become available or a predefined time has passed, a block containing the S local parameters is locked and added to the blockchain. If some of the clients fail to report updated local parameters, the orderer recycles the previous local parameter of the failed clients. In this protocol, one block of the blockchain immutably stores the set of the local parameters: $\boldsymbol{\theta}_{\mathrm{lo.}} = \{\boldsymbol{\theta}_{\mathrm{lo.}}^s\}_{s=1}^S$.

On the endorsers side, given the set of the local parameters, once a new block is approved and locked, `e_update` is performed to update the dictionary. Specifically, the endorsers compute

$$N_k \leftarrow \sum_{s=1}^{S} N_k^s, \tag{20}$$

$$\boldsymbol{\mu}_k \leftarrow \frac{1}{\lambda_0 + N_k} \sum_{s=1}^{S} \boldsymbol{m}_k^s, \tag{21}$$

$$\Sigma_k \leftarrow \frac{1}{N_k} \sum_{s=1}^{S} \mathsf{C}_k^s + \boldsymbol{\mu}_k \boldsymbol{\mu}_k^\top, \tag{22}$$

and solve Eq. (13). Once computation is done, the endorsers broadcast the global parameter (or the dictionary)

$$\boldsymbol{\theta}_{\text{gl.}} = \{\boldsymbol{\mu}_k, \Lambda_k\}_{k=1}^K. \tag{23}$$

One potential issue here is that `e_update` has to wait until a block gets locked. This is indeed a critical point in any Blockchain-based architectures for collaborative learning. By definition, collaborative learning requires information sharing among the clients. Thus, in general, transactions are *stateful*: a transaction made by one client will affect another transaction by another client. One approach to handle this issue is to leverage the idea of speculative execution [25], [26]. See Sec. IV-D for an additional discussion on this issue.

Algorithm 1 summarizes `CollabDict`, the proposed collaborative dictionary learning protocol. Once a convergent solution is obtained, each client can build the anomaly score function using Eq. (2). By marginalizing the latent variable $\boldsymbol{z}^s$, we have

$$a^s(\boldsymbol{x}^s) = -\ln \sum_{k=1}^{K} \pi_k^s \mathcal{N}(\boldsymbol{x}^s \mid \boldsymbol{\mu}_k, (\Lambda_k)^{-1}). \tag{24}$$

There are three input parameters in the `CollabDict` protocol. For λ_0, one can set $\lambda_0 = 1$ if there is no specific reason to do otherwise. For K, as discussed previously, careful parameter tuning is not really necessary. It is known that Eq. (10) gives a sparse solution in the sense that $\boldsymbol{\pi}^s$ takes zero in many entries [27], which can be viewed as one example of automatic relevance determination (ARD) [27]. Thus we can start a reasonably large value of K. Finally, ρ is the only parameter that calls for tuning. Typically, ρ is determined so that the performance of anomaly detection is maximized. For example, the harmonic average between the anomalous sample accuracy (true positive) and the normal sample accuracy (true negative) can be used as the performance metric to decide on ρ.

D. Remarks on the `CollabDict` protocol

As briefly noted in Introduction, `CollabDict` meets the three major requirements of Blockchain-based CbM platform. First, it has an automatic mechanism to down-weight irrelevant samples through $r_k^{s(n)}$. This is in sharp contrast to the traceability use case, which assumes all the entries are correct and attempts to establish an immutable and traceable database. Second, it establishes a consensus by averaging the local parameters (see Eqs. (20)-(22)). This can be viewed as a statistical generalization of the proof-of-vote mechanism [28]. Unlike the common notion of proof-of-vote, which is typically introduced in an ad hoc fashion, our protocol is principled — it has been introduced as a consequence of the minimum likelihood principle. Third, since the clients do not share any raw data, the algorithm is safe in terms of data privacy.

However, a careful attention should be paid to data privacy. In this protocol, depending on the value of the global parameters, there is actually a risk that the raw samples get exposed to the endorsers. For example, if the cluster center of a cluster happens to be very close to an outlier, the cluster center can be a very faithful surrogate of the outlier datum. To keep this kind of situations from happening, the clients can add some noise to $\{r_k^{s(n)}\}$ so that the protocol satisfies the condition of differential privacy [29]. One interesting mathematical feature of `CollabDict` is that the clients can quantitatively evaluate the risk of privacy breach by checking $\{r_k^{s(n)}\}$. However, we leave the detailed discussion on this issue to a separated paper.

The other important consideration is how to handle the stateful nature of transactions. The update equations (20)-(22), and (13) need the local parameter from all the clients, which may impact on the total computational time. One simple way to address this issue is to rewrite these equations in an online updating form, similar to the idea of stochastic gradient

Algorithm 1 Collaborative dictionary learning

procedure COLLABDICT
  Initialize dictionary $\{(\boldsymbol{\mu}_k, \Lambda_k) \mid k = 1, \ldots, K\}$
  repeat
    for $s \leftarrow 1, S$ **do**
      Perform `c_update`$(\{\boldsymbol{\mu}_k, \Lambda_k\}_{k=1}^K)$
    end for
    for $k \leftarrow 1, K$ **do**
      Perform `e_update`$(\{N_k^s, \boldsymbol{m}_k^s, \mathsf{C}_k^s\}_{s=1}^S)$
    end for
  until convergence
end procedure

procedure C_UPDATE$(\{\boldsymbol{\mu}_k, \Lambda_k\}_{k=1}^K)$
  for $k \leftarrow 1, K$ **do**
    Perform Eqs. (9), (16)-(19)
  end for
return $\{N_k^s, \boldsymbol{m}_k^s, \mathsf{C}_k^s, \pi_k^s\}_{k=1}^K$
end procedure

procedure E_UPDATE$(\{N_k^s, \boldsymbol{m}_k^s, \mathsf{C}_k^s\}_{s=1}^S)$
  for $s \leftarrow 1, S$ **do**
    Perform Eqs. (20)-(22), (13).
  end for
return $\{\boldsymbol{\mu}_k, \Lambda_k\}_{k=1}^K$
end procedure

descent [30] and speculative execution of Smart Contract [25]. This is indeed possible with a little modification of the learning algorithm. We also discuss this extension separately elsewhere.

V. Experiments

From the implementation point of view, there are two interesting questions about this work. The first question is if the proposed collaborative density estimation approach is useful in realistic scenarios. The second question is if the learning procedure really works on Blockchain. Due to limitations of existing Blockchain implementations, especially those related to the difficulty in handling non-categorical real numbers with random noise, we will focus on the first question in this section.

A. Methods compared

The proposed anomaly detection framework features the sparse Gaussian graphical model to achieve practical interpretability. We compare `CollabDict` with two alternative algorithms that can learn sparse and thus interpretable dependency structures in the multi-task learning setting: the group graphical lasso (`group`) and fused graphical lasso (`fused`) algorithms [31]. Similarly to the proposed method, these methods can compute client-wise precision matrices $\{\Lambda^s\}$ but they allow only the single precision matrix (*i.e.* $K = 1$). Therefore, the anomaly score is defined as

$$-\ln \mathcal{N}(\boldsymbol{x}^s \mid \hat{\boldsymbol{x}}^s, (\Lambda^s)^{-1}) \tag{25}$$

for each client s, where $\hat{\boldsymbol{x}}^s$ is the sample mean over $\mathcal{D}^s$, which will be the zero vector for centered data sets.

B. Data cleansing

We applied `CollabDict` to a real-world data set collected from physical sensors installed on industrial vehicles. The data set includes about 100 files of 42-variate time-series data ($M = 42$) with about 150 time points ($N^s \sim 150$) under normal operating conditions. We randomly picked the first ten files and think of them as the clients ($S = 10$). For initialization, we split each data set into three equal-length blocks ($K = 3 \times 10$) and standardized each variable by adding 2σ with σ being the standard deviation. Since samples have been taken under normal operating conditions, an anomaly detection algorithm should give very small anomaly scores for all the samples. However, due to random noise and measurement errors, some samples may be highly contaminated and should be removed from the data set. Data cleansing is the task to identify such outlying samples. Intuitively the goal is to pick up samples that are located farthest from any of the clusters.

Upon convergence, `CollabDict` gave only five clusters, out of which three were dominant. Close inspection shows that those clusters correspond to high and low amplitude fluctuations. This makes sense because most of the time-series incur high fluctuations in the beginning and in the end due to the transient behaviors of engine start and stop.

One important feature of Gaussian-based probabilistic models is its capability of computing the conditional distributions for each variable. Specifically, given $\mathcal{N}(\boldsymbol{x}^s \mid \boldsymbol{\mu}_k, (\Lambda_k)^{-1})$, the conditional distribution of the i-th variable x_i^s given the other variables, denoted by $\boldsymbol{x}_{-i}^s$, is written by

$$p_k(x_i^s \mid \boldsymbol{x}_{-i}^s) = \mathcal{N}(x_i^s \mid m_{k,i}, \sigma_{k,i}^2) \tag{26}$$

$$m_{k,i} = [\boldsymbol{\mu}_k]_i - \frac{1}{[\Lambda_k]_{i,i}} \sum_{j \neq i} [\Lambda_k]_{i,j} [\boldsymbol{x}^s - \boldsymbol{\mu}_k]_j \tag{27}$$

$$\sigma_{k,i}^2 = \frac{1}{[\Lambda_k]_{i,i}} \tag{28}$$

where $[\cdot]_i$ and $[\cdot]_{i,i}$ denoted the i-th and (i,i) entry of the object inside the square bracket. See, *e.g.* Appendix B of [22] for derivation.

Using this formula, we computed the variable-wise anomaly score for each sample for the task of data cleansing, which is to find outliers in the training data set itself as explained above. Figure 2 shows a typical result, where `CollabDict` gives a much smoother signal than `group` and `fused`. Notice that `group` and `fused` have large signals in the beginning and the last periods. This corresponds to the high-fluctuation components in the multivariate system and cannot be captured by single-modal models. For the purpose of data cleansing, the robustness of this kind is quite useful since high fluctuations upon engine start and stop should not be thought of as a malfunction.

To quantify the goodness of the smoother trend in the anomaly score, Fig. 3 shows accuracy-coverage plots (a surrogate of the ROC curve in this setting), where the coverage is defined by how much percentage of samples is covered by a threshold, and the accuracy in this case means the ratio of the samples below the threshold. The curve is drawn by changing the threshold. The AUC values of this plot are summarized in Table I. As shown, `CollabDict` clearly achieves the best AUC. Different variables and tasks have a different value of AUC, but they share the same trend.

C. Anomaly detection

The data set used in the previous section also includes 20 faulty data files, in which two sensors produced faulty signals due to a physical problem. Although the original data is not in the collaborative learning setting, we use the dictionary with five components learned in the previous subsection to build the anomaly score function. We again used the formula Eq. (26) in Eq. (24) to get the variable-wise score function.

The goal of this anomaly detection task is to precisely pinpoint the faulty variables from the others in terms of the anomaly score. We employed a two-sample test setting. We randomly picked a normal and a faulty dataset and computed

TABLE I
AUC values.

	`CollabDict`	`group`	`fused`
cleansing (Fig. 3)	**0.933**	0.863	0.856
detection (Fig. 4)	**0.950**	0.887	0.897

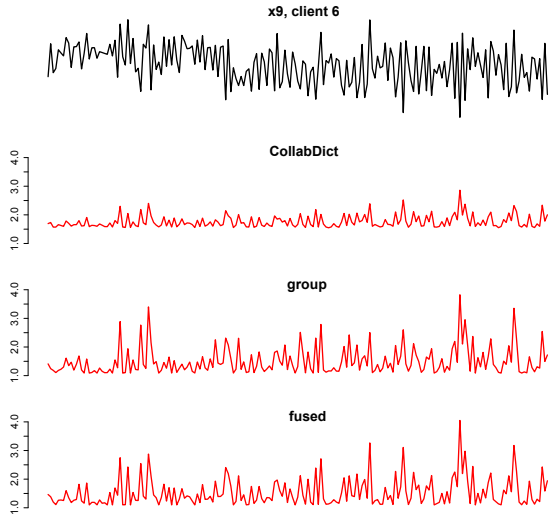

Fig. 2. Variable-wise anomaly scores for x_9 of task 6. The top row shows the raw signal, while the bottom three are anomaly scores computed by the specified methods.

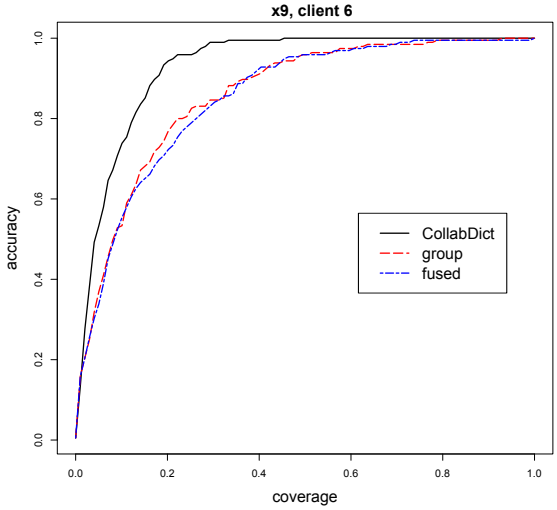

Fig. 3. Comparing coverage-accuracy curves. See Table I for corresponding AUC values.

the variable-wise anomaly score averaged over the samples. If the anomaly score was computed right, the faulty two sensors should stand out in terms of anomaly score. We repeated this 200 times for different normal-faulty pairs and computed the negative sample accuracy (true negative) and positive sample accuracy (true positive).

Figure 4 show the result. The corresponding AUC values are shown in Table I. The results clearly show the better performance of `CollabDict`. As discussed in the previous subsection, the data contains different operational conditions especially due to transient behaviors of engine start and stop. By collecting various patterns from different clients (or

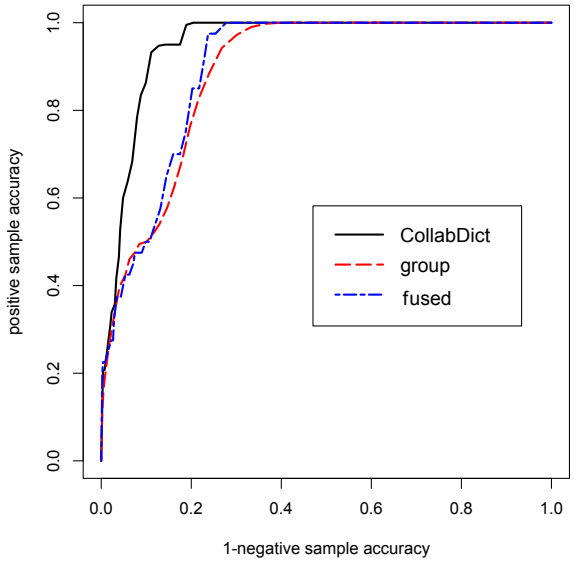

Fig. 4. Comparison of anomaly detection performance.

multiple data sets) and keeping them in the dictionary, the resulting model has better coverage than the single pattern models.

VI. Concluding Remarks

We have proposed a framework for collaborative anomaly detection on Blockchain. Taking condition-based management of industrial assets as a practical example, we successfully extended the notion of Smart Contract, which has implicitly assumed deterministic discrete quantities, to noisy real-valued sensor data. We showed that the three major technical challenges of validation, consensus building, and data security are naturally addressed within a machine learning algorithm. Specifically, the weighting factor introduced by the EM framework serves to invalidate contaminated and thus less informative samples. Consensus building is achieved as a natural consequence of statistical aggregation, which can also be viewed as a statistical generalization of the proof-of-vote mechanism [28]. Since the proposed `CollabDict` protocol requests the client to share only aggregated statistics while keeping the raw data private, it can easily be made satisfy the condition of differential privacy by combining with an appropriate randomization scheme. We formalized the task of collaborative anomaly detection as collaborative dictionary learning based on a multi-task Gaussian mixture and demonstrated that it can capture the multi-modality of the real world.

Regarding industrial applications of Blockchain, data traceability scenarios have been the central topic in the Blockchain community so far. Essentially Blockchain is a technology to share information among trustless parties. We argue that Blockchain should be redefined as a platform for *collaborative learning* rather than just a traceable, immutable, and

decentralized data management system. The key contribution of this paper is the first proposal of a practical protocol that achieves collaborative learning on Blockchain. We pointed out that handling the statefulness of transactions as a result of collaborative learning is one of the biggest technical challenges, in addition to the three requirements of validation, consensus building, and data security. Extending the proposed framework to guarantee differential privacy and to relax the statefulness of transactions will be an interesting future research topic.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[3] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, 2017.

[4] M. Muzammal, Q. Qu, and B. Nasrulin, "Renovating blockchain with distributed databases: An open source system," *Future Generation Computer Systems*, vol. 90, pp. 105–117, 2019.

[5] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, p. 30.

[6] F. Tian, "An agri-food supply chain traceability system for china based on RFID & blockchain technology," in *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*. IEEE, 2016, pp. 1–6.

[7] L. Matsakis, "Praying to Satoshi at the Blockchain Art Expo," Retrieved from https://www.wired.com/story/ethereal-summit-blockchain-art/, May 2018.

[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[9] J. W. Sheppard, M. A. Kaufman, and T. J. Wilmer, "Ieee standards for prognostics and health management," *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 9, pp. 34–41, 2009.

[10] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," in *Proc. of the 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2017, pp. 1–5.

[11] J. Cohn, P. Finn, S. Nair, and S. Panikkar, "Device democracy – saving the future of the internet of things," Retrieved from https://www-935.ibm.com/services/us/gbs/thoughtleadership/internetofthings/, Sep 2014.

[12] A. Bahga and V. K. Madisetti, "Blockchain platform for industrial internet of things," *Journal of Software Engineering and Applications*, vol. 9, no. 10, p. 533, 2016.

[13] S. Huh, S. Cho, and S. Kim, "Managing iot devices using blockchain platform," in *Proceedings of the 19th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2017, pp. 464–467.

[14] Y. Yuan and F.-Y. Wang, "Towards blockchain-based intelligent transportation systems," in *Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 2663–2668.

[15] B. Nasrulin, M. Muzammal, and Q. Qu, "Chainmob: Mobility analytics on blockchain," in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2018, pp. 292–293.

[16] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017, pp. 618–623.

[17] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv preprint arXiv:1707.08114*, 2017.

[18] Y. Xiao, B. Liu, S. Y. Philip, and Z. Hao, "A robust one-class transfer learning method with uncertain data," *Knowledge and Information Systems*, vol. 44, no. 2, pp. 407–438, 2015.

[19] X. He, G. Mourot, D. Maquin, J. Ragot, P. Beauseroy, A. Smolarz, and E. Grall-Maës, "Multi-task learning with one-class svm," *Neurocomputing*, vol. 133, pp. 416–426, 2014.

[20] T. Idé, D. T. Phan, and J. Kalagnanam, "Multi-task multi-modal models for collective anomaly detection," in *Proc. of the 17th IEEE Intl. Conf. on Data Mining (ICDM 17)*, 2017, pp. 177–186.

[21] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," *Data Mining and Knowledge Discovery*, vol. 8, no. 3, pp. 275–300, 2004.

[22] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[23] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[24] S. L. Lauritzen, *Graphical Models*. Oxford, 1996.

[25] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, "Adding concurrency to smart contracts," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*. ACM, 2017, pp. 303–312.

[26] C. Cachin, S. Schubert, and M. Vukolic, "Non-determinism in byzantine fault-tolerant replication," in *Proceedings of the 20th International Conference on Principles of Distributed Systems (OPODIS)*, 2016, pp. 24:1–24:16.

[27] A. Corduneanu and C. M. Bishop, "Variational Bayesian model selection for mixture distributions," in *Artificial intelligence and Statistics*, vol. 2001, 2001, pp. 27–34.

[28] K. Li, H. Li, H. Hou, K. Li, and Y. Chen, "Proof of vote: A high-performance consensus protocol based on vote mechanism & consortium blockchain," in *Proceedings of 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE, 2017, pp. 466–473.

[29] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[30] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT)*. Springer, 2010, pp. 177–186.

[31] P. Danaher, P. Wang, and D. M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 76, no. 2, pp. 373–397, 2014.