Secure Distributed Execution of Principal Components Analysis (PCA)

Sai V. Mudumba
Graduate Student
Purdue University
smudumba@purdue.edu

Jehan C. Shah
Graduate Student
Purdue University
shah435@purdue.edu

***Abstract*—Increasing number of heterogeneous agents that are distributedly connected due to the nature of autonomous operations increase the need for a distributed approach for solving machine learning algorithms. Hence, the traditional centralized approached fall short of scaling to higher number of operations. This paper presents a distributed PCA implementation in a peer-to-peer network implementation, with a range of use case scenarios comparing the global dataset to centralized PCA and distributed PCA for validation, and also involving malicious nodes and how they impact the results on distributed PCA using Iris dataset. K-means is applied for evaluating the results to the ground-truth labels via rand indexing. Three attacks strategies were designed. In addition, two approaches for secure distributed PCA are proposed to detect and delete the malicious nodes, and their advantages and disadvantages are also elaborated.**

I. Introduction

A. Motivation

As autonomous systems increase in operations, there will be an increase in reliance on Machine Learning, Deep Learning, Reinforcement Learning methodologies to enable these operations. Specifically, the challenge will be to create pipelines for enabling Machine Learning, Deep Learning, Reinforcement Learning algorithms in real-time heterogeneous systems. As the number of heterogeneous systems scale, the current centralized approaches breakdown in terms of efficiency and communication complexity. Heterogeneous systems refers to a diversity of systems with various functionalities (Fig. 1) such as embedded devices, edge devices, aerial drones, servers, automobiles, etc.

These algorithms will be deployed among these distributed systems and must be capable of dealing with both adversarial control and data planes. Adversarial control means that some of the nodes (i.e., the systems) cannot be trusted and has been identified as leaking critical information or violating the integrity of the results. An adversarial data plane means that the algorithm will have to operate with malicious manipulated data sources. As data is analyzed in distributed nodes, any loss or an adversarial attack on it could effect the communication interfaces between the nodes. For example, an adversarial node with the largest node centrality can impact the whole network.

GitHub Code: https://github.com/jehanshah8/distributed-pca/tree/PCA-p2p-network-with-tests

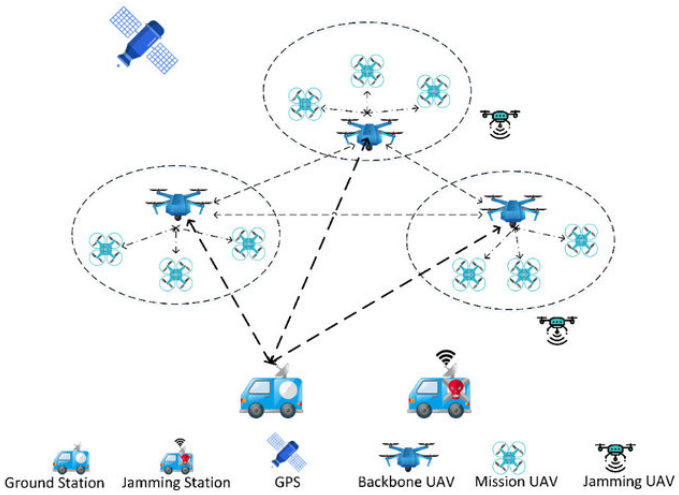

Fig. 1. An Example of a Network of Heterogeneous Systems [1]

B. Problem Formulation

The aim of this paper is to tackle the challenges of implementing a Machine Learning algorithm in a distributed manner that can provide probabilistic guarantees on security and latency, under a set of powerful adversarial attacks. The specific problem statement this paper intends on tackling is as follows:

How to design a distributed Principal Component Analysis (PCA), inject a series of attacks, measure the vulnerabilities to the nodes, and propose defensive techniques to make the distributed PCA more resilient?

C. Contributions of the Paper

As will be discussed in the upcoming sections, multiple literature sources exist on implementing a centralized PCA; however, the implementation of distributed PCA remains scarce. Therefore, the contributions of this paper begin with an rudimentary implementation of the distributed PCA and validating it using the centralized PCA approach. The following additional points detail the key contributions sequentially:

1) Take a distributed Principal Components Analysis (PCA) on a dataset and run K-means clustering using a set of evaluating metrics

2) Incrementally inject consistent attacks to a fraction of the nodes and run K-means clustering to measure performance variations from the former step
3) Identify how vulnerable the overall distributed PCA algorithm is to the adversarial attack, and further identify the critical nodes
4) Propose two defensive techniques to make the distributed PCA algorithm more resilient to adversarial attacks, with the pros and cons detailed

Some of the assumptions used in this paper may be constrained by the scope of the problem definition or network topology definition. For example, as shown in Fig. 2, the distributed vs. decentralized are two different words, with two different meanings. Decentralized means a set of nodes with high network centrality, while other nodes connected to it have low network centrality. On the other hand, distributed means there is no single central owner in the network. In this paper, we assume a decentralized framework in the back-end to collect all the local dataset samples and use distributed network in the front-end when the nodes broadcast information to each other. Network topologies assumptions are discussed in a later section.

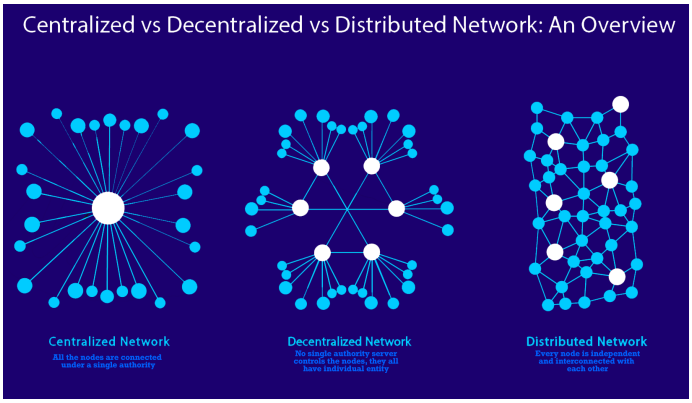

Fig. 2. Centralized vs. Decentralized vs. Distributed [2]

II. Literature Review

A. What is Principal Components Analysis (PCA)?

High dimensional data is becoming ubiquitous. Image resolutions are in the Megapixel size; text is in hundreds of thousands of words; business data includes millions of products; and running analysis on these datasets at their size drastically increases computational costs when training machine learning algorithms. Principal Components Analysis (PCA) is an unsupervised dimensionality reduction technique used for versatile applications [3].

There are many applications that can benefit from using PCA. PCA can be used for anomaly detection [4], [5], from credit card frauds [6] to erroneous measurements in wind tunnel experiments [7]. Training deep learning models on a set of high resolution images will increase the computational time. However, running a PCA on the set of images, compressing the images to a low-rank approximation [8], and passing it into the deep learning model reduces time complexity. In addition, the image compression can also aid in noise reduction via reconstruction (Fig. 3). Furthermore, by reducing the dimensions using PCA, we can also visualize the higher dimensional data onto a 1-D, 2-D, or 3-D graphs [9], [10].

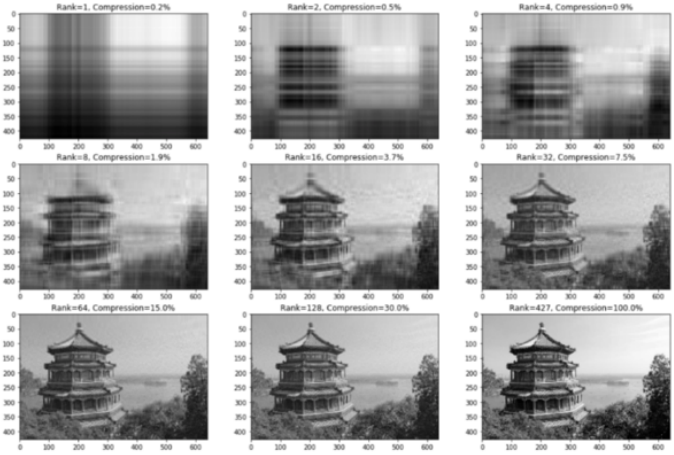

Fig. 3. PCA: Image compression due to dimensionality reduction) [11]

Let's briefly go over what PCA does. PCA finds the best linear projection onto a lower-dimensional space. In other words, PCA finds the line that has the smallest projection error (Fig. 4). It can be formalized as minimizing the linear reconstruction error of the data using only $k \leq d$ dimensions:

$$\min_{Z,W} ||X_c - ZW^T||_F^2 \tag{1}$$

where $X_c = X - \mu_x 1^T \in \mathbb{R}^{nxd}$ which centers the input data
$Z \in \mathbb{R}^{nxk}$ is the latent representation or "scores"
$W^T \in \mathbb{R}^{kxd}$ are the principal components
n is the number of samples
d is the original dimensions
k is the desired dimensionality reduction

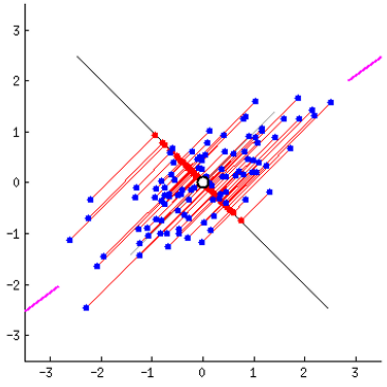

Fig. 4. PCA: finding the line with the smallest projection error (purple line vs red line) [11]

Minimizing reconstruction error is also equivalent to maximizing the variance of the projection (see Fig. 4). Therefore,

the PCA solution is the top k right singular vectors via Singular Value Decomposition (SVD) [12].

If $X_c = USV^T$, then the solution is simply $W* = V_{1:k}$. In other words, the truncated SVD is the best approximation.

The downside of doing this in a centralized node can be easily seen. As the number of dimensions increases and the number of samples increases, the computation of SVD becomes increasingly computationally expensive as SVD essentially is finding the eigenvectors and eigenvalues of the X_c matrix. Hence, there is an interest in allocating the computation across a distributed network of nodes.

B. What is Distributed PCA?

The distributed PCA approach works similar to the centralized PCA, but it divides the datasets locally to each node, and each node solves a local PCA. This paper implements distributed PCA from Algorithm pseudocode provided in Liang et. al. [13]. The pseudocode of the Algorithm is shown in Fig. 5. In Round 1, the algorithm does local PCA on each node, and each node announces the singular values and vectors of the t largest components. In Round 2, the local data is then projected onto these t global principal components and concatenated. This distributed PCA implementation is chosen from the other distributed PCA implementations ([14], [15]) because Liang et. al. [13] claims to provide any theoretical guarantees and relate distributed PCA to K-means clustering.

Algorithm 1 Distributed PCA

1: **Input:** local data sets $\{P_i, 1 \leq i \leq n\}$, projected dimension t.
2: ▷ `Local PCA`
3: **Round 1:** on each node $v_i \in V$ do
4: Compute local SVD: $P_i = U_i D_i (E_i)^T$.
5: Let $D_i^{(t)}$ be the matrix that contains the first t diagonal entries of D_i and is 0 otherwise.
6: Let $P_i^{(t)} = U_i D_i^{(t)} (E_i)^T$. Let $E_i^{(t)}$ be the matrix that contains the first t columns of E_i.
7: ANNOUNCE: $D_i^{(t)}, E_i^{(t)}$.
8: ▷ `Global PCA`
9: **Round 2:** on each node $v_i \in V$ do
10: Use $D_i^{(t)}, E_i^{(t)}$ to compute $S_i^{(t)} = (P_i^{(t)})^T P_i^{(t)}$. Set $S^{(t)} = \sum_{i=1}^{n} S_i^{(t)}$.
11: Compute the eigenvectors for the estimated covariance matrix: $S^{(t)} = E \Lambda E^T$.
12: Let $E^{(t)}$ be the matrix that contains the first t columns of E.
13: Project $P_i^{(t)}$ on $E^{(t)}$, resulting in $\hat{P}_i = P_i^{(t)} E^{(t)} (E^{(t)})^T$.
14: **Output:** $\hat{P}^T = [\hat{P}_1^T, \ldots, \hat{P}_n^T]$.

Fig. 5. Distributed PCA Pseudocode [13]

Let's walk through a simple example of how the pseudo code executes:

1) Fig. 6 shows a 5-node, fully-connected network topology, with each node having a local dataset of size n_i x d, where $i = 1, 2, 3, 4, 5$ and d is the number of dimensions (Line 1 in Fig. 5 pseudocode)
2) Each node does local PCA with its local dataset and broadcasts its low-rank singular values and right singular vectors. A closer look using node 1 is shown in Fig. 7 (Lines 2 - 7 in Fig. 5 pseudocode)
3) Once all nodes broadcast their singular values and vectors to each other node in their connectivity, each node has all the other nodes' singular values and vectors, as shown in Fig. 8 (Lines 9 - 10 Fig. 5 pseudocode)
4) Each node estimates the covariance matrix, $S^{(t)}$ and estimates the global covariance matrix by summing all the locally estimated covariance matrices by $S^{(t)} = S_1^{(t)} + S_2^{(t)} + ... + S_5^{(t)}$, as shown in Fig. 9 (Lines 9 - 10 Fig. 5 pseudocode)
5) After the total covariance matrix is estimated, the low-rank approximate of P is found at each node and is called $\hat{P}$, as shown in Fig. 10. These approximations will differ among the nodes if there is a malicious node (Lines 11 - 14 Fig. 5 pseudocode)

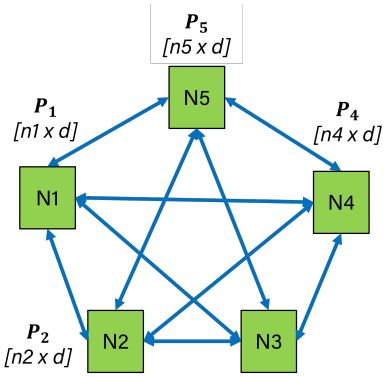

Fig. 6. 5-Node Fully-Connected Network Topology, where $n1, n2, ..., n5$ are the number of samples in the local datasets, and d is the number of dimensions

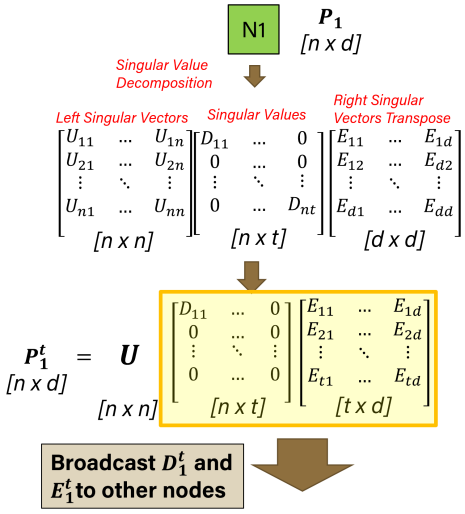

Fig. 7. Node 1 does local PCA on its local datasets, finds the left singular vectors (nxn), singular values (nxt), and the right singular vector transpose (dxd), and broadcasts E_1^t and D_1^t

III. Simulation Setup

Now that the distributed PCA pseudocode is summarized in the previous section, let's build a simulation. The following sections walk through the network topology used in this study, the design of attacks and their reasoinigs, datasets overview, and the evaluation metrics for the results.

A. Network Topology

A 7-Node network is defined. This network can have up to 6 malicious nodes. Formally, this is a peer-to-peer network,

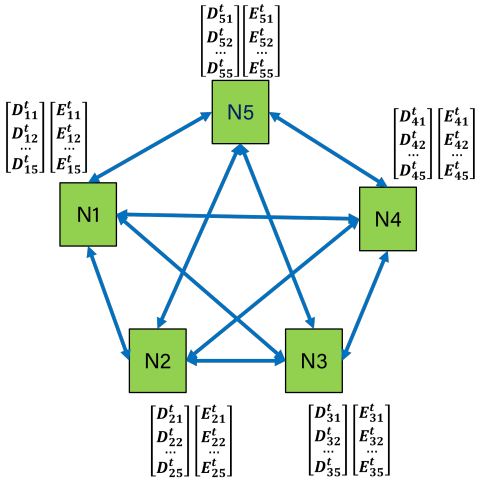

Fig. 8. All the nodes have each other's broadcasted singular values (D_{ij}^t) and singular vectors (E_{ij}^t), where ij means node i received from node j

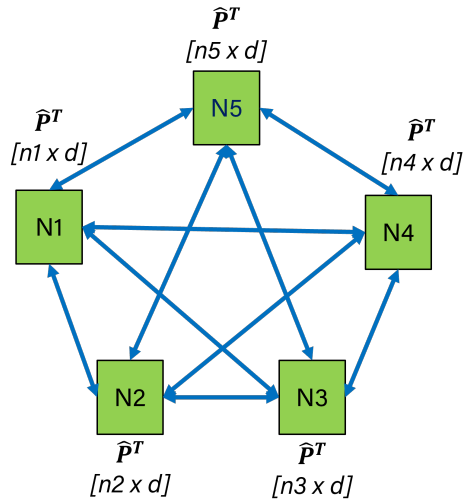

Fig. 10. Estimating low-rank approximation of P called $\hat{P}$, so each node has its own version of the approximation of the global dataset P (Note: these approximations will differ among the nodes if there is a malicious node)

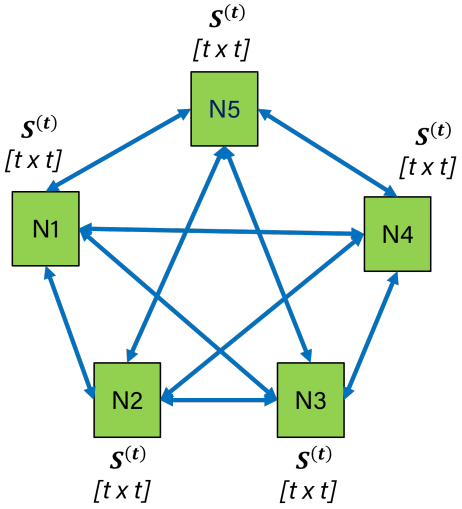

Fig. 9. Each node estimates the covariance matrix, $S^{(t)}$ and estimates the global covariance matrix by summing all the locally estimated covariance matrices by $S^{(t)} = S_1^{(t)} + S_2^{(t)} + ... + S_5^{(t)}$

as opposed to a central computer that processes information. Network topologies are discussed in detail in Verbraeken et. al. [16]. The following are the assumptions:

- Network is fully-connected; each node broadcasts to every other node. In reality, however, this is computationally expensive on its own to communicate with all the nodes, and in some instances, it is not feasible to communicate with all the nodes. For example, if the nodes are flying aerial drones and the drones have limited radius for vehicle-to-vehicle communication, then the node can only communicate with those that are within its communication radius. Acknowledging this important limitation, we set our initial node topology to fully connected nodes, where the number of nodes are kept small. Future node topologies will include different levels of centrality and distributedness.
- Each node assumes that its local dataset is a good approximation of the global dataset. This assumption is required so that we can use the evaluation metrics that will be defined in the next subsection. In addition, under this assumption, non-malicious nodes will have similar principal components (if each node is a good approximation of the global dataset, then each node is a good approximation to each other node). Similarly, if there is a malicious node in the pack, it will stand out from the other nodes.
- Attacker has information on the node topology (i.e., the number of nodes, and the type of information that is being broadcasted). This assumption is a must since the cyberattacks we expect would not happen if the attacker has no information about the network topology and connectedness.

B. Design of Attacks

The high-level goal of cyberattack formulation is that a malicious node attacker broadcasts invalid principal components (i.e., singular vectors) of its local dataset. In Fig. 5, the weak point is Line 7, where each node is making an announcement to each other node. In this paper, 3 design of attacks are formulated and are discussed below. However, in general, the design of attacks is not limited to only these.

1) ***Design of Attack 1: Broadcast Least 't' Principal Components*** *:* Malicious nodes broadcast the least 't' principal components of the local datasets that explain the variance to all other nodes. The most 't' principal components explain most of the variance in the datasets. Hence, by picking the least principal components, we remove the cruicial information that explains which direction (i.e., the eigen-vector) the data is represented (see Fig. 4 for a clear understanding).

*2) **Design of Attack 2: Broadcast Perpendicular Principal Components**:* Malicious nodes broadcast perpendicular principal component vectors and an inverse of the singular values of the local datasets to all the other nodes. The idea behind this design is that given a singular vector that represents most of the variance of the data, a malicious node falsely broadcasts a perpendicular vector instead to trick the other nodes.

*3) **Design of Attack 3: Broadcast Random Principal Components**:* Like the above two attacks, this one broadcasts a random principal component vector than the true vector. The purpose of this attack is that there is no purpose. The broadcasting of incorrect principal components is based on a random policy, which is probably more realistic given its stochastic nature, while the above two attacks are relative to the true principal component.

C. Datasets Overview

The dataset chosen for this initial study is the Iris dataset. It is a well-established dataset with many resources available in the machine learning community, and also computationally light. Another potential dataset we considered was the MNIST digits dataset, however, it is quite large, having 784 dimensions, and 60,000+ samples. This paper aims to provide a stable, working results, and hence, MNIST was not considered for the scope of this paper.

The Iris dataset consists of 3 classes of Iris flowers: Versicolor, Setosa, Virginica. There are 4 dimensions: sepal length and width, petal length and width. Overall, it includes 149 samples. When K-means is performed, the clustering estimates a low-rank matrix to one of the 3 classes, and compared to the ground-truth label, which is available in the dataset.

The PCA performed on this dataset will reduce the dimensions from 4 to 2. Hence, we will find the 2 dimensions that capture the most variance.

Fig. 11. Iris Dataset: 149 samples, 4 dimensions (sepal length and width, petal length and width), 3 classes (Iris Setosa, Iris Versicolor, Iris Virginica) [17], [18]

D. Evaluation Metrics

To evaluate the results of the distributed PCA, we propose 3 metrics of evaluations. The following list gives a brief description of the metrics:

1) Cosine Similarity (intermediate calculation): this metric measures how similar two vectors are. This is useful to evaluate whether the assumption made in Network Topology is true (regarding nodes assuming local data is good approximation of the global data). On the other hand, if two vectors are dissimilar, then either one of them is malicious. We do not know which one is malicious just by looking at those two. In summary, Cosine Similarity measures similarity of the principal component vectors it receives in Fig. 8. A 0 means two vectors are dissimilar; a 1 means two vectors are similar. The ideal case is where two nodes have similar principal components, and thus, a high Cosine Similarity metric. Although this is an intermediate calculation, this metric plays an important role when implementing secure distributed PCA.
2) Total Explained Variance: Sum of the singular values squared divided by the local dataset size gives the total explained variance. A mathematical expression is as follows:
$$(\sum_{i=1}^{t} d_i^2)/(n-1) \tag{2}$$
3) K-means Clustering using Rand Index: we use 3-means for this dataset since there are 3 classes. 3-means finds 3 centers in the graph to cluster the points to any of the 3 classes of Iris flowers. Given that the malicious attacks on the nodes can change the global PCA, doing 3-means provides an insight into how much the clustering has changed to the ground-truth labels. Rand Index measures the similarities between two data clusters.

IV. Results and Discussion

In this section, let's look at the results of total explained variance and rand index from 3-means clustering of the 3 attacks proposed in this study across the original global data (og data), the centralized PCA estimates (c pca), distributed PCA when 0 nodes are malicious, distributed PCA when 1 node is malicious, all the way to distributed PCA when 6 nodes are malicious, on a 7-node network.

A. Total Explained Variance

Fig. 12 shows the total explained variance of the PCA on the Iris dataset for 3 types of attacks across the various cyberattack use case scenarios mentioned above. The takeaway is that the centralized PCA approximates the original dataset closely. Furthermore, distributed PCA with 0 malicious nodes also approximates the explained variance of the original dataset and the centralized PCA. The variations happen when we introduce increasing malicious nodes, starting with 1 malicious node. The total explained variance by the network decreases drastically for the randomized and reverse order attacks, while the variance by perpendicular attack also goes down at a slower rate.

B. Rand-Index for K-means

Rand index shows % of labels that are misclustered relative to the ground truth labels, which reduces with increasing number of malicious nodes. The rand index ranges from 0 to 1, with 0 being the lowest accuracy and 1 being highly accurate. The rand index for 3-means compares the clusters and sees

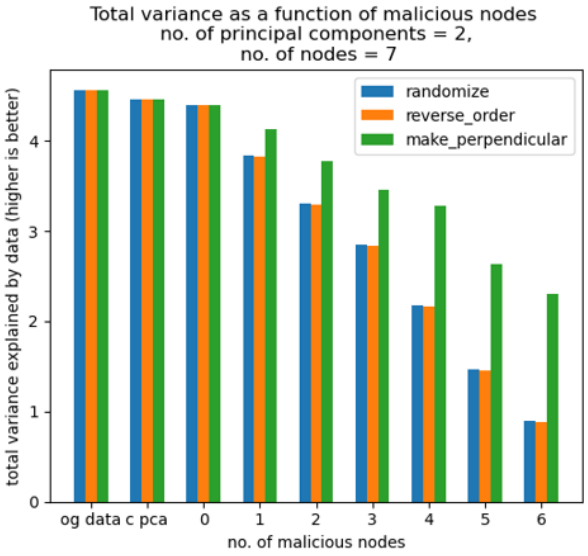

Fig. 12. PCA on Iris Dataset: 3 Types of Attacks: Randomize, Reverse, Perpendicular; how increase in the number of malicious nodes in the 7-Node, fully-connected network topology changes total expected variance by the data

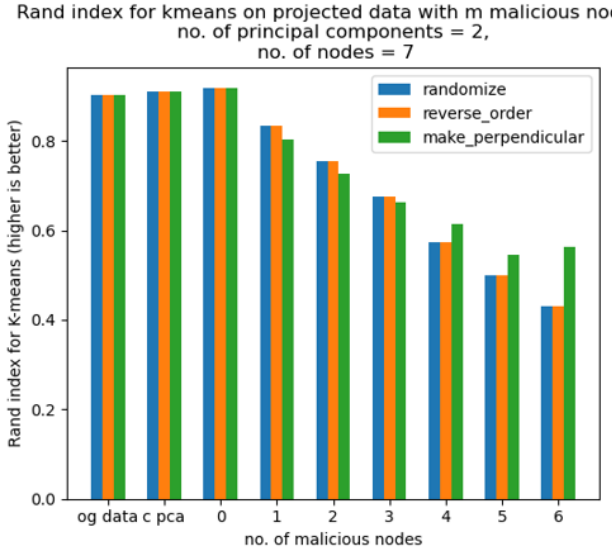

Fig. 13. PCA on Iris Dataset:3 Types of Attacks: Randomize, Reverse, Perpendicular; how increase in the number of malicious nodes in the 7-Node, fully-connected network topology varies the Rand-index for K-means

how much variation is there with respect to the ground-truth labels. Similar to Fig. 12, Fig. 13 shows the accuracy of the clustering. It follows a similar trend to total explained variance, where the introduction of malicious nodes decreases the accuracy of the clustering. In this case all three attacks decrease the rand index.

Now, let's look at what the projected datasets look like across original global data (og data), the centralized PCA estimates (c pca), distributed PCA when 0 nodes are malicious, distributed PCA when 1 node is malicious, all the way to distributed PCA when 6 nodes are malicious, on a 7-node network.

C. Projected Data: Randomized Attack

Fig. 14 shows the datasets across the 9 use case scenarios for randomized attack. It can be noticed that the original data, centralized PCA, distributed PCA with 0 malicious nodes are validated and look similar, which supports the results seen in Fig. 12-13. As the number of malicious nodes increase, there is less variance being captured due to malicious nodes injecting randomized principal component vectors.

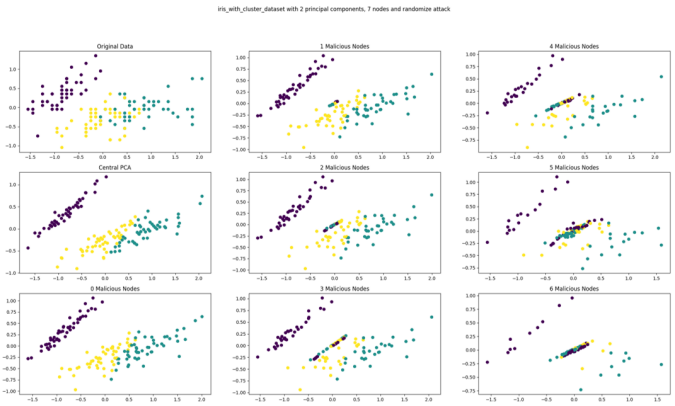

Fig. 14. PCA on Iris Dataset: Randomize Attack Projected Data with various Malicious Nodes Compared with the Original Data Projection

D. Projected Data: Reversed Attack

Fig. 15 shows the projected data for reversed order attack. For the initial use case scenarios, note the similarities with Fig. 14. As the number of malicious nodes increase, the variance captured in the direction decreases. When there are 6 malicious nodes in the network, the direction of the projected data is flipped by 90 degrees.

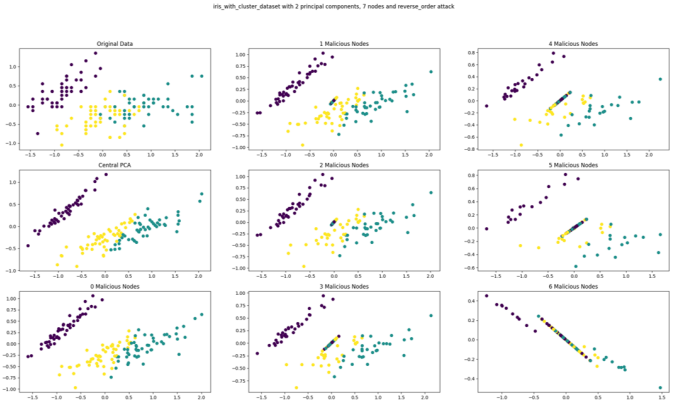

Fig. 15. PCA on Iris Dataset: Reverse Attack Projected Data with various Malicious Nodes Compared with the Original Data Projection

E. Projected Data: Perpendicular Attack

Fig. 16 shows the projected data for the perpendicular attack. For the initial use cases, again, note the similarities with Fig. 14-15. For more than 1 malicious nodes, the direction of the projected data is flipped.

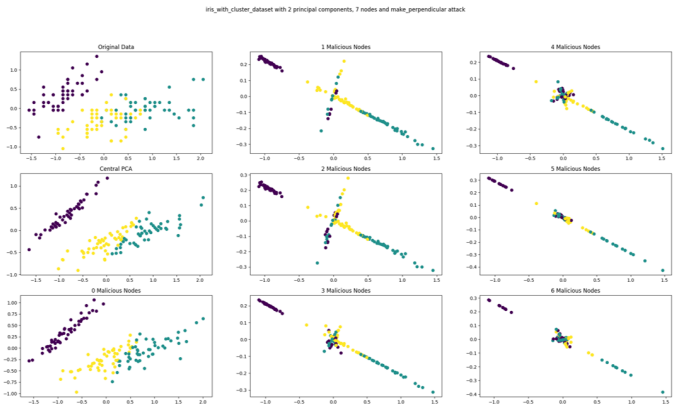

Fig. 16. PCA on Iris Dataset: Perpendicular Attack Projected Data with various Malicious Nodes Compared with the Original Data Projection

V. Secure Distributed PCA

Now that we have seen how malicious nodes impact the total explained variance and the K-means clustering analysis, let's look at some secure distributed PCA approaches proposed in this paper.

A. Approach 1: Independently Ignore Suspicious Nodes

The first approach is to independently ignore any suspicious nodes. To do this, Cosine Similarity metric is utilized. The goal is to calculate Cosine Similarity of the singular vectors received at a node from other nodes ($E_{11}^t, ..., E_{17}^t$ for node 1). In the Round 2 of the algorithm proposed in Fig. 5 (Lines 9 - 10), a node will not consider another nodes' information in estimation of the covariance matrix if the Cosine Similarity is less than some percent relative to the median cosine similarity estimates. Essentially, any node that appears to have low Cosine Similarity to the others relative to the median is discarded. After this elimination, the low-rank approximation of the global dataset is found (i.e., $\hat{P}$). The advantages of this approach are that it has low latency and no extra communication overheads. The disadvantage is that due to the cutoff policy, even non-malicious nodes with relatively low Cosine Similarities will be discarded, which might be seen as discarding valuable data. Fig. 17 shows a high-level overview of this approach using a 5-Note network, where the red node (N5) is the malicious node, and N1, N2, N3, N4 find that its Cosine Similarity metric is relatively low and decide independently to discard its information in estimating $\hat{P}$.

B. Approach 2: Reach Consensus on a Suspicious Set

The main disadvantage of Approach 1 was that good nodes might be discarded if they have relatively poor Cosine Similarity metric. This approach aims to reach consensus with

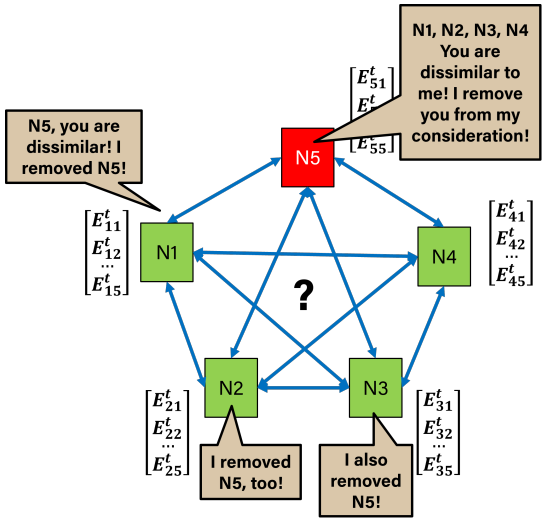

Fig. 17. PCA on Iris Dataset:3 Types of Attacks: Randomize, Reverse, Perpendicular; how increase in the number of malicious nodes in the 7-Node, fully-connected network topology varies the Rand-index for K-means

the other nodes via a suspicious set. Each node creates a suspicious set of nodes based on Cosine Similarity of singular vectors and broadcasts this to all the other nodes. If a majority of the nodes reach an agreement on the malicious nodes, then that node is discarded. The advantage is that non-malicious nodes are not eliminated. However, the disadvantages are the extra communications overhead and high latency due to time to reach an agreement. An additional disadvantage is when there are a majority of malicious nodes in a network that decide to discard the few good nodes. In this scenario, the tables turn the other way and lead to malicious nodes having an upper hand. Fig. 18 shows a high-level overview of this approach using a 5-Note network, where the red node (N5) is the malicious node, and N1, N2, N3, N4 reach a consensus to eliminate data received from N5 to estimate $\hat{P}$.

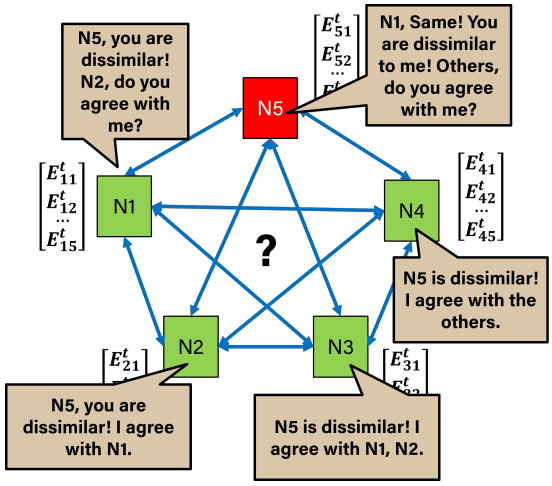

Fig. 18. PCA on Iris Dataset:3 Types of Attacks: Randomize, Reverse, Perpendicular; how increase in the number of malicious nodes in the 7-Node, fully-connected network topology varies the Rand-index for K-means

VI. Conclusion

Increasing number of heterogeneous agents that are distributedly connected due to the nature of autonomous operations increase the need for a distributed approach for solving machine learning algorithms. Hence, the traditional centralized approached fall short of scaling to higher number of operations. This paper presents a distributed PCA implementation in a peer-to-peer network implementation, with a range of use case scenarios comparing the global dataset to centralized PCA and distributed PCA for validation, and also involving malicious nodes and how they impact the results on distributed PCA on Iris dataset when K-means is applied and compared to the ground-truth labels via rand indexing. Three attacks were designed, which include changing the principal components in a random way, making the principal components perpendicular to the estimated direction, and picking the least 't' principal components that do not explain the total variance of the data. In addition, two approaches for secure distributed PCA are proposed to detect and delete the malicious nodes, and their advantages and disadvantages are also elaborated.

VII. Future Work

The future work will involve using a more realistic network topology (e.g., power law networks). As we already mentioned in the paper, there are limitations to the network topology used in this paper, but was well suited for having a working simulation. The future work will also expand on the secure distributed PCA approaches proposed by implementing them and showing that they do indeed work by repeating the figures displayed in the Results section. The future work will also increase the intelligence of attacks by incorporating an adaptive adversary who knows what defense mechanism is being used. We will also look more broadly into other distributed PCA implementations that were mentioned in the paper, but not simulated, and study how vulnerable these implementations are to the other similar distributed PCA algorithms. The key is whichever distributed PCA fares well against these adversary attacks takes the mantle, while the other distributed PCA versions will quickly become obsolete. With this in mind, the authors of this paper keep an evolving mindset to look more broadly than what was shown in this paper.

Acknowledgment

The authors would like to thank Prof. Saurabh Bagchi for providing us with this problem statement, along with the motivation, and for guiding us through the questions and doubts via couple of Office Hours, even one on Saturday. We appreciate his time and feedback provided, and will be actively scoping the readiness of this paper into a conference paper based on his recommendation.

References

[1] URL: https://www.researchgate.net/publication/332906620_A_Dyna-Q-Based_Solution_for_UAV_Networks_Against_Smart_Jamming_Attacks.

[2] *Centralized vs decentralized vs distributed*. Jan. 2019. URL: https://blockchainengineer.com/centralized-vs-decentralized-vs-distributed-network/.

[3] David Reich, Alkes L Price, and Nick Patterson. *Principal component analysis of Genetic Data*. URL: https://www.nature.com/articles/ng0508-491.

[4] James McCaffrey10/21/2021. *Anomaly detection using principal component analysis (PCA)*. URL: https://visualstudiomagazine.com/articles/2021/10/20/anomaly-detection-pca.aspx.

[5] Likebupt. *PCA-based Anomaly Detection: Component Reference - Azure Machine Learning*. URL: https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/pca-based-anomaly-detection.

[6] *PCA-based Anomaly Detection*. Mar. 2022. URL: https://www.atmosera.com/blog/pca-based-anomaly-detection/.

[7] Aaron Defreitas et al. *Anomaly detection in wind tunnel experiments by Principal Component Analysis*. Jan. 2022. URL: https://arc.aiaa.org/doi/10.2514/1.J060349.

[8] Rukshan Pramoditha. *Image compression using principal component analysis (PCA)*. Apr. 2021. URL: https://towardsdatascience.com/image-compression-using-principal-component-analysis-pca-253f26740a9f.

[9] Adrian Tam. *Principal component analysis for visualization*. Oct. 2021. URL: https://machinelearningmastery.com/principal-component-analysis-for-visualization/.

[10] *Principal Components Analysis Visualization*. URL: https://plotly.com/python/pca-visualization/.

[11] URL: https://www.davidinouye.com/course/ece57000-fall-2021/lectures/pca.pdf.

[12] *Singular Value Decomposition (SVD) Tutorial*. URL: https://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm.

[13] URL: https://www.cs.cmu.edu/~ninamf/papers/distributedPCAandCoresets.pdf.

[14] *Principal Component Analysis for Dimension Reduction in Massive Distributed Data Sets*. URL: https://technicalreports.ornl.gov/cppr/y2001/pres/116016.pdf.

[15] Zheng-Jian Bai, Raymond H. Chan, and Franklin T. Luk. "Principal component analysis for distributed data sets with updating". In: *Lecture Notes in Computer Science* (2005), pp. 471–483. DOI: 10.1007/11573937_51.

[16] Joost Verbraeken et al. *A survey on distributed machine learning*. Dec. 2019. URL: https://arxiv.org/abs/1912.09789.

[17] URL: https://archive.ics.uci.edu/ml/datasets/iris.

[18] URL: http://www.lac.inpe.br/~rafael.santos/Docs/CAP394/WholeStory-Iris.html.