International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307

OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/IJSRCSEIT

Improved Data Poison Detection using Multiple Training Models

[1]Ch. Swathi, [2]Rakshitha Tirumalasetty, [3]Karicharla Kavitha, [4]Bejawada Gana Maruthi, [5] Nandeti Teja Sai Singh

[1]Assistant Professor, [2,3,4,5] UG Students

Department of CSE (AI & ML), Sri Vasavi Institute of Engineering and Technology, Nandamuru, Andhra Pradesh, India

ARTICLEINFO

Article History:

Accepted: 10 April 2024

Published: 22 April 2024

Publication Issue

Volume 10, Issue 2

March-April-2024

Page Number

705-712

ABSTRACT

Distributed machine learning (DML) can realize massive dataset training when no single node can work out the accurate results within an acceptable time. However, this will inevitably expose more potential targets to attackers compared with the non-distributed environment. In this paper, we classify DML into basic-DML and semi-DML. In basic-DML, the center server dispatches learning tasks to distributed machines and aggregates their learning results. While in semi-DML, the center server further devotes resources into dataset learning in addition to its duty in basic-DML. We firstly put forward a novel data poison detection scheme for basic-DML, which utilizes a cross-learning mechanism to find out the poisoned data. Then, for semi-DML, we present an improved data poison detection scheme to provide better learning protection with the aid of the central resource. To efficiently utilize the system resources, an optimal resource allocation approach is developed. Simulation results show that the proposed scheme can significantly improve the accuracy of the final model by up to 20% for support vector machine and 60% for logistic regression in the basic-DML scenario.

Keywords : Distributed Machine Learning, Data Poison Detection.

I. INTRODUCTION

Distributed machine learning (DML) has been widely used in distributed systems where no single node can get the intelligent decision from a massive dataset within an acceptable time. In a typical DML system, a central server has a tremendous amount of data at its disposal. It divides the dataset into different parts and disseminates them to distributed workers who perform the training tasks and return their results to the center. Finally, the center integrates these results and outputs the eventual model. Unfortunately, with the number of distributed workers increasing, it is hard to guarantee the security of each worker. This lack of security will increase the danger that attackers poison the dataset and manipulate the training result. Poisoning attack is a typical way to tamper the training data in machine learning. Especially in scenarios that

newly generated datasets should be periodically sent to the distributed workers for updating the decision model, the attacker will have more chances to poison the datasets, leading to a more severe threat in DML.
Unfortunately, with the number of distributed workers increasing, it is hard to guarantee the security of each worker. This lack of security will increase the danger that attackers poison the dataset and manipulate the training result. Poisoning attack is a typical way to tamper the training data in machine learning. Especially in scenarios that newly generated datasets should be periodically sent to the distributed workers for updating the decision model, the attacker will have more chances to poison the datasets, leading to a more severe threat in DML. Hence we are going to use distributed machine learning so it will take less computation time for processing the data and also we are going to detect the data poisoning in data.

II.RELATED WORK

In order to achieve a greener intelligent transport system (ITS), an efficient collaboration between vehicles is required to manage computation task processing with low latency. In this paper, we propose a collaborative edge computing scheme for vehicular Internet-of-things towards a greener ITS. The proposed scheme uses some vehicles as edge nodes, which are responsible for finding task processor nodes on behalf of a task requester node by considering the end-to-end task response time. The proposed scheme employs a two stage approach where the first stage enables an efficient networking and computing architecture by forming vehicle clusters based on the edge architecture, and the second stage optimizes offloading tasks based on the architecture. We use realistic computer simulations to compare the proposed scheme with existing baselines, and show its superiority in terms of task offloading performance.
The combination of cognitive radio (CR) and non-orthogonal multiple access (NOMA) has tremendous potential to achieve high spectral efficiency in the IoT era. In this paper, we focus on the energy-efficient resource allocation of a cognitive multiple-input single-output (MISO) NOMA system with the aid of simultaneous wireless information and power transfer (SWIPT). Specifically, a non-linear energy harvesting (EH) model is adopted to characterize the non-linear energy conversion property. In order to achieve the green design goal, we aim for the minimization of the system power consumption by jointly designing the transmit beamformer and the receive power splitter subject to the information transmission and EH harvesting requirements of second users (SUs), and the maximum tolerable interference constraints at primary users (PUs). However, the formulated optimization problem is nonconvex and hard to tackle. By exploiting the classic semi-definite relaxation (SDR) and successive convex approximation (SCA), we propose a penalty function-based algorithm to solve the nonconvex problem. The convergence of the proposed algorithm is further proved. Finally, simulation results demonstrates that the non-linear EH model is able to strongly reflect the property of practical energy harvester and the performance gain of the proposed algorithm than the baseline scheme.
In the scenario of mobile fog computing (MFC), communication between vehicles and fog layer, which is called vehicle-to-fog (V2F) communication, needs to use bandwidth resources as much as possible with low delay and high tolerance for errors. In order to adapt to these harsh scenarios, there are important technical challenges concerning the combination of network coding (NC) and multipath transmission to construct high-quality V2F communication for cloud-aware MFC. Most NC schemes exhibit poor reliability in burst errors that often occur in high-speed movement scenarios. These can be improved by using interleaving technology. However, most traditional interleaving schemes for multipath transmission are designed based on round robin (RR) or weighted round robin (WRR), in practice, which can waste a lot of bandwidth resources. In order to solve those problems, this paper proposes a novel multipath transmission scheme for

cloud-aware MFC, which is called Bidirectional Selection Scheduling (BSS) scheme. Under the premise of realizing interleaving, since BSS can be used in conjunction with a lot of path scheduling algorithms based on Earliest Delivery Path First (EDPF), it can make better use of bandwidth resources. As a result, BSS has high reliability and bandwidth utilization in harsh scenarios. It can meet the high-quality requirements of cloudaware MFC for transmission.

Federated learning enables training collaborative machine learning models at scale with many participants whilst preserving the privacy of their datasets. Standard federated learning techniques are vulnerable to Byzantine failures, biased local datasets, and poisoning attacks. In this paper we introduce Adaptive Federated Averaging, a novel algorithm for robust federated learning that is designed to detect failures, attacks, and bad updates provided by participants in a collaborative model. We propose a Hidden Markov Model to model and learn the quality of model updates provided by each participant during training. In contrast to existing robust federated learning schemes, we propose a robust aggregation rule that detects and discards bad or malicious local model updates at each training iteration. This includes a mechanism that blocks unwanted participants, which also increases the computational and communication efficiency. Our experimental evaluation on 4 real datasets show that our algorithm is significantly more robust to faulty, noisy and malicious participants, whilst being computationally more efficient than other state-of-the-art robust federated learning methods such as Multi-KRUM and coordinate-wise median.

Internet of Things (IoT) has been seen playing a tremendous change in the Information Technology (IT) environments, and thus its importance has also been realized and played a vital role within Intelligent Home Networks (IHNs). This is because IoT establishes a connection between things and the Internet by utilizing different sensing devices to implement the intelligence to deal with the identification and management of the connected things. IHNs use intelligent systems to perform their daily operations. Meanwhile, these networks ensure comfort, safety, healthcare, automation, energy conservation, and remote management to devices and users. Apart from that, these networks provide assistance in self-healing for faults, power outages, reconfigurations, and more. However, we have realized that more and advanced devices and services continue to be introduced and used in these networks. This has led to competitions of the limited available network resources, services, and bandwidth. In this paper, therefore, we present the design and implementation of a Novel Dynamic Bandwidth Allocation (NoDBA) algorithm to solve the performance bottleneck incurred with IHNs. The proposed algorithm deals with the management of bandwidth and its allocation. In the proposed algorithm, this study integrates two algorithms, namely; Offline Cooperative Algorithm (OCA) and Particle Swarm Optimization (PSO) to improve the Quality of Service (QoS). PSO defines the priority limits for subnets and nodes in the network. Meanwhile, OCA facilitates dynamic bandwidth allocation in the network. The Network Simulator-2 (NS-2) was used to simulate and evaluate the NoDBA and it showed improved results compared to the traditional bandwidth allocation algorithms. The obtained results show an average throughput of 92%, an average delay of 0.8 seconds, and saves energy consumption of 95% compared to Dynamic QoS-aware Bandwidth Allocation (DQBA) and Data-Driven Allocation (DDA).

III.PROPOSED SYSTEM

We are going to classify DML into basic-DML and semi-DML. In basic-DML, the center server dispatches learning tasks to distributed machines and aggregates their learning results. While in semi-DML, the center server further devotes resources into dataset learning in addition to its duty in basic-DML. We firstly put forward a novel data poison detection scheme for basic-DML, which utilizes a cross-learning mechanism

to find out the poisoned data. Then, for semi-DML, we present an improved data poison detection scheme to provide better learning protection with the aid of the central resource. To efficiently utilize the system resources, an optimal resource allocation approach is developed. Simulation results show that the proposed scheme can significantly improve the accuracy of the final model by up to 20% for support vector machine and 60% for logistic regression in the basic-DML scenario.

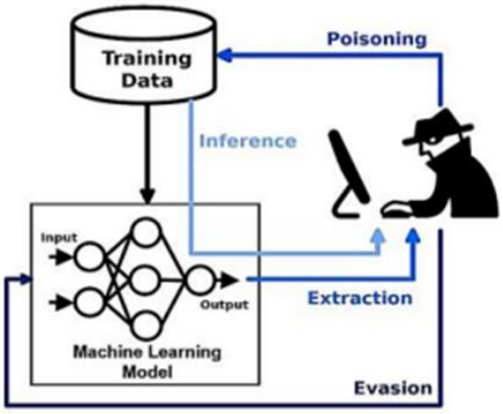


Fig 1: Proposed Flow Architecture

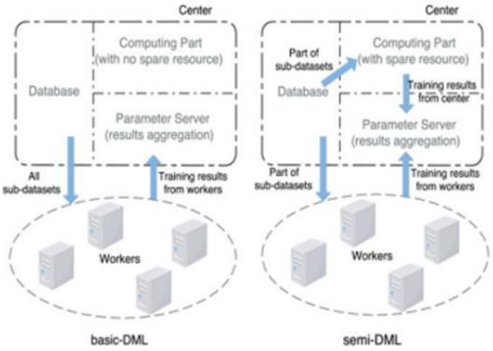


Fig 2: System Architecture

DATA POISONING:

Data poisoning or model poisoning attacks involve polluting a machine learning model's training data. Data poisoning is considered an integrity attack because tampering with the training data impacts the model's ability to output correct predictions.

PREVENT AND DETECT

The difficulties in fixing poisoned models, model developers need to focus on measures that could either block attack attempts or detect malicious inputs before the next training cycle happens things like input validity checking, rate limiting, regression testing, manual moderation and using various statistical techniques to detect anomalies. For example, a small group of accounts, IP addresses, or users shouldn't account for a large portion of the model training data. Restrictions can be placed on how many inputs provided by a unique user are accepted into the training data or with what weight.

DATA POISONING ATTACKS IN DEEP LEARNING VISION SYSTEMS

The practice of using deep learning methods in safety critical vision systems such as autonomous driving has come a long way. As vision systems supported by deep learning methods become

ubiquitous, the possible security threats faced by these systems have come into greater focus. As it is with any artificial intelligence system, these deep neural vision networks are first trained on a data set of interest, once they start performing well, they are deployed to a real-world environment. In thetraining stage, deep learning systems are susceptible to data poisoning attacks.

IV.RESULTS AND DISCUSSION

```
import dask.dataframe as dd
train_data = dd.read_csv('Train_data.csv')
test_data = dd.read_csv('test_data.csv')
train_data.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised
0	0	icmp	20	2	491	0	0	0	0	0	0	0	0
1	0	udp	45	2	146	0	0	0	0	0	0	0	0
2	0	icmp	50	4	0	0	0	0	0	0	0	0	0
3	0	icmp	25	2	232	8153	0	0	0	0	0	1	0
4	0	icmp	25	2	199	420	0	0	0	0	0	1	0

```
[ ] train_data['xAttack'].compute().unique()
```

```
array(['normal', 'dos', 'r2l', 'probe', 'u2r', nan], dtype=object)
```

```
[ ] test_data.head()
```

c_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	xAttack
0.00	0.00	0.65	0.65	0.32	0.32	dos
1.00	0.06	0.00	0.00	0.00	0.00	normal
0.01	0.05	0.01	0.00	0.00	0.00	normal
0.00	0.00	0.00	0.00	0.00	0.00	normal
0.18	0.00	0.00	0.00	0.00	0.00	normal

Fig 3. Results screenshot

```
[ ] train_data.columns
```

```
Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
       'num_access_files', 'num_outbound_cmds', 'is_host_login',
       'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
       'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
       'dst_host_srv_count', 'dst_host_same_srv_rate',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
       'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
       'dst_host_srv_rerror_rate', 'xAttack'],
      dtype='object')
```

check descriptive analysis

```
[ ] train_data.compute().describe()
```

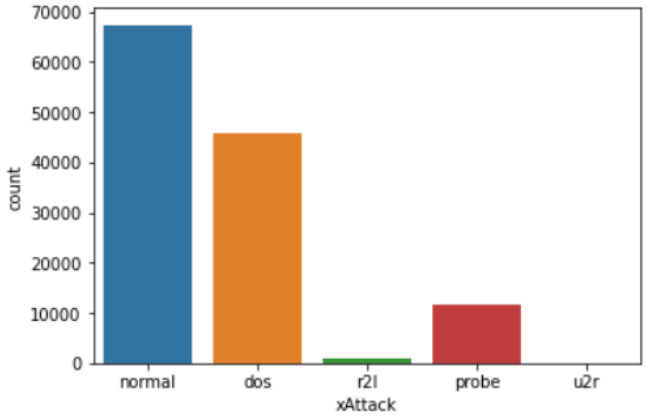
	duration	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_fail
count	115967.000000	115967.000000	115967.000000	1.159670e+05	1.159670e+05	115967.000000	115967.000000	115967.000000	115967.000000	115
mean	285.859684	32.085869	2.574612	3.676942e+04	2.124395e+04	0.000207	0.022972	0.000121	0.202308	
std	2591.260845	16.469378	1.141824	4.583699e+06	4.191118e+06	0.014385	0.255306	0.014973	2.133965	
min	0.000000	1.000000	1.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	20.000000	2.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	25.000000	2.000000	4.400000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	

Fig 4. Results screenshot

```
[ ] # train set dimension
    print('Train set dimension: {} rows, {} columns'.format(train_data.compute().shape[0], train_data.compute().shape[1]))

    Train set dimension: 125973 rows, 42 columns

[ ] # train set dimension
    print('Test set dimension: {} rows, {} columns'.format(test_data.compute().shape[0], test_data.compute().shape[1]))

    Test set dimension: 10000 rows, 42 columns

[ ] # 'num_outbound_cmds' field has all 0 values. Hence, it will be removed from both train and test dataset since it is a redundant field.
    train_data.compute().drop(['num_outbound_cmds'], axis=1, inplace=True)
    test_data.compute().drop(['num_outbound_cmds'], axis=1, inplace=True)
```

Fig 5. Results screenshot

```
[ ] <matplotlib.axes._subplots.AxesSubplot at 0x7f89f83feeb8>
```




```
[ ] sns.countplot(x = 'protocol_type', hue = 'xAttack', data = training_data)

    <matplotlib.axes._subplots.AxesSubplot at 0x7f89f7e852e8>
```

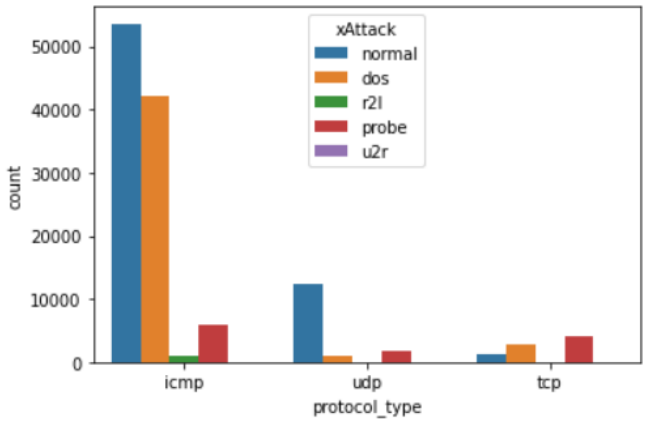


Fig 6. Results screenshot

```
[1] from dask_ml.preprocessing import LabelEncoder
```

```
[3] import dask.dataframe as dd
    train_data = dd.read_csv('Train_data.csv')
    test_data = dd.read_csv('test_data.csv')
    train_data.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised
0	0	icmp	20	2	491	0	0	0	0	0	0	0	0
1	0	udp	45	2	146	0	0	0	0	0	0	0	0
2	0	icmp	50	4	0	0	0	0	0	0	0	0	0
3	0	icmp	25	2	232	8153	0	0	0	0	0	1	0
4	0	icmp	25	2	199	420	0	0	0	0	0	1	0

```
[4] # 'num_outbound_cmds' field has all 0 values. Hence, it will be removed from both train and test dataset since it is a redundant field.
    train_data.compute().drop(['num_outbound_cmds'], axis=1, inplace=True)
    test_data.compute().drop(['num_outbound_cmds'], axis=1, inplace=True)
```

```
[5] labelencoder = LabelEncoder()
    train_data['protocol_type'] = labelencoder.fit_transform(train_data['protocol_type'])
    train_data['xAttack'] = labelencoder.fit_transform(train_data['xAttack'])
```

```
[6] test_data['protocol_type'] = labelencoder.fit_transform(test_data['protocol_type'])
    test_data['xAttack'] = labelencoder.fit_transform(test_data['xAttack'])
```

Fig 7. Results screenshot

```
[8] from dask_ml.preprocessing import StandardScaler
    std_scaler = StandardScaler()
    train_transform = std_scaler.fit_transform(X_train)
    test_transform = std_scaler.fit_transform(X_test)
```

```
[9] from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score
```

```
[10] lr = LogisticRegression()
     lr.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:72: DataConversionWarning: A column-vector y was passed when a 1d array was ex
  return f(**kwargs)
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:764: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression()
```

```
[11] y_pred = lr.predict(X_test)
     print(f'Accuracy with Logistic regression is {accuracy_score(Y_test,y_pred)}')
```

```
Accuracy with Logistic regression is 0.6328
```

Fig 8. Results screenshot

Fig 9. Results screenshot

V. CONCLUSION

Data poison detection schemes in both basic-DML and semi-DML scenarios. Furthermore, we presented an improved data poison detection scheme in the semi-DML scenario. Simulation results show that in the basic-DML scenario, the proposed scheme can increase the model accuracy by up to 20% for support vector machine and 60% for logistic regression, respectively.

VI. FUTURE WORK

In the future, the data poison detection scheme can be extended to a more dynamic pattern to fit the changing application environment and attacking intensity. Besides, since the multi-training of sub-datasets would increase the resource consumption of the system, the trade-off between security and resource cost is another topic that needs to be studied further.

II. REFERENCES

[1]. Jason Brownlee, “What is Deep Learning?” August 16, 2019, https://machinelearningmastery.com/what-is-deep-learning/

[2]. Mathworks, “What Is Deep Learning?” https://www.mathworks.com/discovery/deeplearning.html

[3]. Computer Science, University of Maryland, “Poison Frogs! Targeted Poisoning Attacks on Neural Networks,” https://www.cs.umd.edu/~tomg/projects/poison/

[4]. Keith D. Foote, “A Brief History of Deep Learning,” Feburary 7, 2017, https://www.dataversity.net/brief-history-deep-learning/

[5]. Reportlinker, “Global Deeping Learning Industry,” July 2020, https://www.reportlinker.com/p05798338/Global-Deep-Learning-Industry.html?utmsource=GNW

[6]. Larry Hardesty, “Explained: Neural networks,” April 14, 2017, https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

[7]. Jeff Dean, “Large-Scale Deep Learning for Intelligent Computer Systems,”https://static.googleusercontent.com/

media/research.google.com/en//people/jeff/BayLearn2015.pdf

[8]. DeepAI, "Feature Extraction," https://deepai.org/machine-learning-glos sary-and-terms/featureextraction

[9]. Artem Oppermann, "Artificial Intelligence vs. Machine Learning vs. Deep Learning," October 29, 2019, https://towardsdatascience.com/artif icial-intelligence-vs-machine-learning-vs-deeplearning-2210ba8cc4ac

[10]. Alexander Polyakov, "How to attack Machine Learning (Evasion, Poisoning, Inference, Trojans, Backdoors)," August 6, 2019, https://toward sdatascience.com/how-to-attack-machine-learningevasion-poisoninginference-trojans-backdoors-a7cb5832595c

[11]. Ilja Moisejevs, "Poisoning attacks on Machine Learning," July 14, 2019, https://towardsdatascience.com/poisoning-attacks-on-machine-learning -1ff247c254db

[12]. Daniel Lowd, Christopher Meek, "Good Word Attacks on Statistical Spam Filters," Semantic Scholar, https://www.semanticscholar.org/paper/Good-Word-Attacks-on-Statistical-Spam-FiltersLowd-Meek/16358a75a3a6561d042e6874d128d82f5b0bd4b3

[13]. Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, Tom Goldstein, "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," in Proceedings of 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, Canad, https://papers.nips.cc/paper/7849-po ´ ison-frogs-targeted-clean-labelpoisoning-attacks-on-neural-networks.p df

[14]. Luis Munoz-Gonz ˜ alez, Battista Biggio, Ambra Demontis, Andrea Pau- ´ dice, Vasin Wongrassamee, Emil C. Lupu, Fabio Roli, "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization," August 29, 2017, https://arxiv.org/abs/1708.08689

[15]. Keras, https://keras.io/

[16]. TensorFlow, https://www.tensorflow.org/