2023 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 17–20, 2023, ROME, ITALY

A BIC-BASED MIXTURE MODEL DEFENSE AGAINST DATA POISONING ATTACKS ON CLASSIFIERS

Xi Li, David J. Miller, Zhen Xiang and George Kesidis

the School of EECS, Pennsylvania State University, University Park, PA, 16802

ABSTRACT

Data Poisoning (DP) is an effective attack which degrades a trained classifier's accuracy through covert injection of attack samples into the training set. We propose an *unsupervised* Bayesian Information Criterion (BIC)-based mixture model defense against DP attacks that: 1) addresses the most challenging *embedded* DP scenario wherein, if DP is present, the poisoned samples are an *a priori* unknown subset of the training set, and with no clean validation set available; 2) applies a mixture model to both well-fit potentially multi-modal class distributions and capture poisoned samples within a small subset of the mixture components; 3) jointly identifies poisoned components and samples by minimizing the BIC cost defined over the whole training set. Our experimental results demonstrate the effectiveness of our defense under strong DP attacks, as well as its superiority over other works.

Index Terms— Adversarial learning, Data poisoning attack, Anomaly detection, Mixture model, Bayesian Information Criterion

1. INTRODUCTION

Machine learning systems are vulnerable to maliciously crafted inputs [22]. In this work, we address "error generic" data poisoning attacks (hereafter called DP attacks) [2, 28, 9, 18, 20, 23] against models trained for classification tasks, which aim to degrade the *overall* classification accuracy. DP attacks involve the insertion of "poisoned" samples into the training set of a classifier. To effectively mislead classifier training with few poisoned samples, attackers introduce feature collision[14] by *e.g.*, label-flipping attacks[1], preventing accurate class decision boundary learning. DP attacks have been successfully demonstrated against Support Vector Machines (SVMs) [28], Logistic Regression (LR) models [9], collaborative filtering systems [18], differentially-private learners [20], and neural networks (NN) [23].

A general and challenging attacking setting remains largely unsolved: if DP exists, the poisoned samples are an *unknown* subset embedded among the clean training samples. That is, the defender does not know whether an attack is present, and if so, which samples are poisoned and which class(es) are corrupted. Most studies on defending against such attacks make assumptions on, *e.g.*, the type of classifier [9, 7], the type of DP attack [25], and/or the training data (*e.g.*, possessing a clean validation set[24, 3]). The proposed method does not make any such assumptions.

Poisoned samples are generally *atypical* of the class to which they are labeled. We thus apply mixture modeling [21] to accurately explain the potentially multi-modal data and to capture poisoned samples within a subset of mixture components. We expect that re-assigning atypical samples to other classes should increase the overall data likelihood. Furthermore, removing a poisoned component (and re-assigning its samples to the best-fitting components from other classes) will reduce the model complexity of a mixture. Thus, both the data likelihood and model complexity terms that together constitute the Bayesian Information Criterion (BIC) [26] should improve through such sample re-assignments. That is, we propose to make poisoned sample inferences consistent with minimizing BIC.

In summary, our BIC-based defense is 1) **Novel**: To our knowledge, we are the first to formulate a BIC-based defense for *unsupervised* anomaly detection/DP attack mitigation; 2) **Practical**: We address the challenging embedded DP attack scenario, potentially involving multiple poisoned classes, without requiring a clean validation set; 3) **Effective**: The experimental results on several datasets and various classifier structures demonstrate the effectiveness of our defense under strong DP attacks, as well as its superiority over other works.

2. RELATED WORK

[25] relabels a sample based on the plurality label of its K nearest neighbors (KNN) to enforce label homogeneity. [7] detects outliers by applying singular value decomposition (SVD) to the matrix of gradients, with each row of this matrix the sample-wise gradient, with respect to all the model parameters. A training sample is declared an outlier if the squared magnitude of the projection of the gradient onto the top right singular vector is abnormally large. [14] mitigates the effects of DP by gradient shaping (GS), *i.e.*, constraining the magnitude and orientation of poisoned gradients to make them close

Thanks to the AFOSR, ONR, and Cisco for funding.

to clean gradients. [19] applied a BIC-based defense against DP attacks for binary classification tasks, assuming that the attacker only poisons one of the two classes, with this class known to the defender. [3] employs generative adversarial networks for synthetic clean data generation based on the clean dataset possessed by the defender, on which a mimic model is trained. Samples that have different predictions from the mimic model and the target model are deemed poisoned. [17, 27] employ aggregated base classifier predictions trained on data subsets for DP attack mitigation, minimizing the target model's sensitivity to dataset distortions. [17] partitions the training set into disjoint subsets using a deterministic function (*e.g.*, hash function), while [27] improves certified robustness bounds by combining duplicates of the disjoint subsets.

However, [25, 14, 7, 17, 27] are supervised detection methods, with their performances highly impacted by the (supervised) choice of their hyper-parameters (*e.g.*, K in [25]). Also, [7, 17, 27] are computationally expensive. [25, 14] are computationally cheap, but their efficacies dramatically degrade as the number of poisoned samples increases (*cf*. Sec. 5.2). Besides, in practice, the attacker may label-flip data originating from more than one class, with these classes unknown to the defender. Under this challenging attack scenario, [19] may fail even if only one class is poisoned, as the defender might sanitize the clean class based on the poisoned one (*cf*. Tab. 1 and 2). By contrast, our method is unsupervised, applicable to multi-class settings, and does not require a clean validation set.

3. THREAT MODEL

We consider W-class ($W \geq 2$) classification tasks, where the classifier, denoted $f : \mathbb{R}^d \rightarrow \mathcal{Y} \equiv \{1, \ldots, W\}$, is trained on $\mathcal{D}_{\text{Train}}$ and then tested on $\mathcal{D}_{\text{Test}}$. $\mathcal{D}_{\text{Train}}$ and $\mathcal{D}_{\text{Test}}$, with both assumed i.i.d. from the same distribution. Each feature x_l, $l = 1, \ldots, d$, may be either discrete or continuous-valued. Both feature types will be considered in our experiments.

We assume the attacker: 1) has sufficient knowledge of the classification domain to to procure legitimate samples; 2) poisons these samples by label-flipping; 3) is able to insert poisoned samples into the training set ($\mathcal{D}_{\text{Train}} = \mathcal{D}_{\text{Clean}} \cup \mathcal{D}_{\text{Attack}}$); 4) may poison any subset of classes with different attack strengths; 5) is unaware of any deployed defenses. The attacker's goal is to significantly degrade the classifier's generalization accuracy. We assume the defender: 1) only uses the training set $\mathcal{D}_{\text{Train}}$ manipulated by the attacker, without additional clean (attack-free) samples; 2) is unaware if an attack is present, and if so, which samples are poisoned, or which classes are corrupted. The defender aims to: 1) identify and remove as many poisoned samples as possible and as few clean samples as possible before classifier training/retraining; and 2) achieve classification accuracy close to that for an unpoisoned training scenario.

4. BIC MIXTURE-BASED SANITIZATION

We highlight the intuition behind our proposed strategy: **First**, poisoned samples form different sub-populations from normal samples. Thus, we apply mixture modeling to each class to well-fit potentially multi-modal class distributions [8, 21] and to isolate poisoned samples within a few components[1]. **Second**, the likelihood of the whole dataset would increase if poisoned samples are re-assigned to the true classes, and the complexity of the mixture model would decrease if a component formed by poisoned samples is removed or revised. Therefore, *we aim to identify and re-assign poisoned samples, and remove/revise poisoned components, such that the overall data likelihood increases and the model complexity decreases*. The Bayesian Information Criterion (BIC) [26] expresses a tradeoff between data likelihood fit and model complexity. Accordingly, we develop an *unsupervised* method to mitigate data poisoning via locally optimal minimization of the BIC objective.

In this work, we consider the *complete data* BIC objective function, based on the complete data log-likelihood function [6], wherein each data sample is hard (fully) assigned to the mixture component under which it has the greatest log-likelihood[2]: $\text{BIC}_{\text{cmplt}} = |\theta|k - \sum_{c=1}^{W}\sum_{j=1}^{M_c} L_j^c$, where θ is the set of parameters specifying a density function model for the data, $|\theta|$ is the number of free parameters in this set, k is the cost (penalty) for describing an individual model parameter and is found to be $k = 0.5 * \log(|\mathcal{D}_{\text{train}}|)$ within an approximate Bayesian setting [16]. M_c is the number of mixture components in the density for class c. The complete data log-likelihood for the data from component j in class c is $L_j^c = \sum_{\boldsymbol{x}\in\mathcal{X}_j^c} \log P[\boldsymbol{x}; \Lambda_j^c]$, where $P[\cdot; \Lambda_j^c]$ is the j^{th} mixture component density under class c, Λ_j^c is the set of parameters specifying the component density, and $\boldsymbol{x} \in \mathcal{X}_j^c$ if and only if, for $\boldsymbol{x}$ labeled to class c, $P[\boldsymbol{x}; \Lambda_j^c] \geq P[\boldsymbol{x}; \Lambda_{j'}^c]\ \forall j' \neq j$. Note that $\theta_c = \{\Lambda_j^c\}$ and $\theta = \bigcup_c \theta_c$.

4.1. BIC-based Defense

Recall $\mathcal{Y} = \{1, \ldots, W\}, W \geq 2$, is the set of classes, and let $T = |\mathcal{D}_{\text{Train}}|$ be the total number of training samples. Let $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathcal{Y}$ represent the feature vector and class label for training sample i. Denote Ω and L as the model complexity and data log-likelihood, respectively. M^c is the number of components in class c. Each component, with a set of parameters Λ_j^c, specifies a joint probability mass function (PMF) or probability density function (PDF), depending on whether the data is discrete or continuous-valued.

The model parameters θ_c of the mixture for class c are estimated via the Expectation-Maximization (EM) algorithm

[1]Note that, in practice, poisoned components may own both poisoned and untainted samples, with the poisoning ratio for each component unknown.

[2]Here we assume the component priors $\{\alpha_j^c\}$ are uniform and hence they are absent from the complete data log likelihood. In practice, these terms do not affect detection performance significantly.

[6], applied to the subset of $\mathcal{D}_{\text{train}}$ labeled as class c. The chosen model order M^c is the one that yields the least BIC cost over the set $\{1,\dots,M^c_{\max}\}$ [26], with $M^c_{\max}$ an upper bound on the number of components in class c's mixture[3]. Finally, let $\mathcal{S}=\{(c,j)|c=1,\dots,W, j=1,\dots,M^c\}$ be the set of components across all classes.

Our data sanitization strategy is consistent with BIC minimization, which involves sample re-assignments, component removals/revisions, and parameter updates. To reflect these model changes, we introduce several "indicator" variables:

(1) $(t_i,j_i) = \arg\max_{t\in\{1,\dots,W\},\, j\in\{1,\dots,M^t\}} P[\boldsymbol{x}_i;\Lambda^t_j]$ is the class and component that best-explain sample $\boldsymbol{x}_i$;
(2) r^c_j is set to 1 if component j in class c is poisoned and set to 0 if it is not;
(3) q^c_j is set to 1 if component j in class c needs to be revised and set to 0 if component j in class c needs to be removed. Note that q^c_j is configured only when $r^c_j=1$.

To account for possible data poisoning, the complete data BIC cost to be minimized is

$$\text{BIC}_{\text{cmplt}}(\theta)=\sum_{c=1}^{W}\sum_{j=1}^{M_c}((1-r^c_j(1-q^c_j))k|\Lambda^c_j|+1+\delta(r^c_j,1))$$
$$-\sum_{c=1}^{W}\sum_{j=1}^{M_c}((1-r^c_j)L^c_j(\Lambda^c_j)+r^c_j\sum_{\boldsymbol{x}_i\in\mathcal{X}^c_j}\log P[\boldsymbol{x}_i;\Lambda^{t_i}_{j_i}]). \qquad (1)$$

In (1), the model parameters are $\theta=\{\{\Lambda^c_j\},\{r^c_j\},\{q^c_j\}\}$, where the structural parameters r^c_j and q^c_j each require one bit to specify (hence the '1' and $\delta(r^c_j,1)$ contributions to the model complexity term). By contrast, t_i and j_i are hidden data assignments (as part of the complete data log-likelihood), not model parameters. Note in (1) that if component (c,j) is removed the model complexity decreases by $k|\Lambda^c_j|$.

To minimize (1) in a locally optimal fashion, our approach involves cycling over the mixture components, one at a time, effecting the change (sample reassignments, component removal, or no change) that reduces BIC the most. Accordingly, the new BIC cost can be expressed as the old BIC cost plus the (negative) change resulting from sample re-assignments or component removal/revision, denoted ΔBIC^c_j.

Each feasible joint configuration of the variables for component (c,j) corresponds to one of three cases:

(1) $r^c_j=0$: The component is formed by clean samples, and there is no need to re-distribute its samples or modify the component (*i.e.*, $\Delta\Omega^c_{j,1}=0$, $\Delta L^c_{j,1}=0$). The change in BIC in this case is thus $\Delta\text{BIC}^c_j=0$.

(2) $r^c_j=1, q^c_j=0$: Component j is poisoned, and we are choosing to remove it from the mixture, changing the model complexity term by $\Delta\Omega^c_{j,2}=-|\Lambda^c_j|\frac{1}{2}\log T$. Each sample $\boldsymbol{x}_i\in\mathcal{X}^c_j$ is re-assigned to component j_i of class t_i, where $(t_i,j_i)=\arg\max_{(t,j')\in\mathcal{S}\setminus\{(c,j)\}}\log P[\boldsymbol{x}_i;\Lambda^t_{j'}]$. Let $\mathcal{Q}=\{(t_i,j_i)|\forall i,\ \boldsymbol{x}_i\in\mathcal{X}^c_j\}$ be the set of components which receive the re-assigned samples. For each component $(w,j')\in\mathcal{Q}$, we re-estimate its parameters on $\widehat{\mathcal{X}^w_{j'}}$ by maximum likelihood estimation (MLE): $\Lambda^{w,\text{new}}_{j'}=\arg\max_\Lambda\sum_{\boldsymbol{x}_i\in\widehat{\mathcal{X}^w_{j'}}}\log P[\boldsymbol{x}_i;\Lambda]$, where $\widehat{\mathcal{X}^w_{j'}}=\mathcal{X}^w_{j'}\cup\{\boldsymbol{x}_i\in\mathcal{X}^c_j|t_i=w, j_i=j'\}$. This optimization has a closed form, globally optimal solution for the component density model forms considered in this paper. The total data log-likelihood changes by

$$\Delta L^c_{j,2}=\sum_{(w,j')\in\mathcal{Q}}\sum_{\boldsymbol{x}_i\in\widehat{\mathcal{X}^w_{j'}}}\log P[\boldsymbol{x}_i;\Lambda^{w,\text{new}}_{j'}]$$
$$-\sum_{(w,j')\in\mathcal{Q}}\sum_{\boldsymbol{x}_i\in\mathcal{X}^w_{j'}}\log P[\boldsymbol{x}_i;\Lambda^w_{j'}]-\sum_{\boldsymbol{x}_i\in\mathcal{X}^c_j}\log P[\boldsymbol{x}_i;\Lambda^c_j].$$

The change in BIC in this case is $\Delta\text{BIC}^c_j=\Delta\Omega^c_{j,2}-\Delta L^c_{j,2}$.

(3) $r^c_j=1, q^c_j=1$: Similar to case (2) but instead of removing it, we re-estimate the parameters of component j by its surviving samples (*i.e.*, samples with $t_i=c$). Revising a component does not change the model complexity cost, *i.e.*, $\Delta\Omega^c_{j,3}=0$. The parameters Λ^c_j are re-estimated by MLE on the surviving samples: $\Lambda^{c,new}_j=\arg\max_\Lambda\sum_{\boldsymbol{x}_i\in\widehat{\mathcal{X}^c_j}}\log P[\boldsymbol{x}_i;\Lambda]$, where $\widehat{\mathcal{X}^c_j}=\{\boldsymbol{x}_i\in\mathcal{X}^c_j|t_i=c\}$. Samples that are best represented by class $t_i=w\neq c$ are re-distributed to their fittest components in class w, but the remaining samples (*i.e.*, $t_i=c$) are explained by the updated component j. Let $\mathcal{Q}'=\{(w,j')\in\mathcal{Q}|w\neq c\}\cup\{(c,j)\}$ be the set of components to be updated. The total data log-likelihood changes by:

$$\Delta L^c_{j,3}=\sum_{(w,j')\in\mathcal{Q}'}\sum_{\boldsymbol{x}_i\in\widehat{\mathcal{X}^w_{j'}}}\log P[\boldsymbol{x}_i;\Lambda^{w,\text{new}}_{j'}]$$
$$-\sum_{(w,j')\in\mathcal{Q}'}\sum_{\boldsymbol{x}_i\in\mathcal{X}^w_{j'}}\log P[\boldsymbol{x}_i;\Lambda^w_{j'}],$$

where $\widehat{\mathcal{X}^w_{j'}}$ and $\Lambda^{w,\text{new}}_{j'}$ $\forall(w,j')\in\mathcal{Q}'\setminus\{(c,j)\}$ are defined in the same way as in case 2. The BIC change in this case is $\Delta\text{BIC}^c_j=-\Delta L^c_{j,3}$.

4.2. Implementation

The optimal configuration for any component j depends on the configurations for other components. It is thus intractable to define an algorithm guaranteed to find a globally optimal configuration over all components. Instead, at each optimization step, we separately *trial*-update each component's configuration, and then only permanently update the component that yields the greatest reduction in BIC. This is repeated until there are no further changes. This optimization approach is non-increasing in the BIC objective and results in a locally optimal solution. Finally, all samples with $t_i\neq y_i$ (*i.e.*, the detected poisoned samples) are removed from the training set, and we have the sanitized training set $\widehat{\mathcal{D}}_{\text{Train}}=\{\boldsymbol{x}_i\in\mathcal{D}_{\text{Train}}|t_i=y_i\}$, which will be used to learn the classifier. See the corresponding Algorithm 1 pseudocode.

[3] $M^c_{\max}$ is not a hyper-parameter, as one can observe the changes of BIC to adjust the range of model orders. For example, if $M^c_{\max}$ yields the least BIC, one can increase $M^c_{\max}$ and repeat model selection until $M^c\neq M^c_{\max}$.

Algorithm 1: BIC-Based DP Attack Defense

```
Input  : $\mathcal{D}_{\text{Train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, $\{\Lambda_j^c\}_{j=1,\dots,M^c,\ c\in\{1,\dots,W\}}$
Output: $\widehat{\mathcal{D}}_{\text{Train}}$
$r_j^c = 0, q_j^c = 0, \forall c, j$ ;
$t_i = y_i, \forall i$;
$\Delta\text{BIC}_j^c = 0, \forall j, c$;
do
    for $c \in \{1, \dots, W\}$ do
        for each component $j$ in class $c$ do
            compute BIC reduction from $j$
              $\Delta\text{BIC}_j^c = \min\{\Delta\Omega_{j,m}^c + \Delta L_{j,m}^c\}_{m=1}^3$;
            configure $\{t_i | \forall \boldsymbol{x}_i \in \mathcal{X}_j^c\}$, $r_j^c$, $q_j^c$ consistent with
              $\Delta\text{BIC}_j^c$;
    $(c^*, j^*) = \arg\min_{c\in\{1,\dots,W\}, j=1,\dots,M^c} \Delta\text{BIC}_j^c$;
    if $r_{j^*}^{c^*} = 1$ then
        For $\boldsymbol{x}_i \in \mathcal{X}_{j^*}^{c^*}$, if $t_i \neq c^*$, re-distribute $\boldsymbol{x}_i$ to component
          $m = \arg\max_{m'} \log P[\boldsymbol{x}_i; \Lambda_{m'}^{t_i}]$ in class $t_i$ and then
          update component $m$'s parameters via MLE;
        if $q_{j^*}^{c^*} = 0$ then
            remove component $j^*$ from $\{\Lambda_j^{c^*}\}_{j=1,\dots,M^{c^*}}$ ;
            re-distribute each $\boldsymbol{x}_i \in \mathcal{X}_{j^*}^{c^*}$ to component
              $m = \arg\max_{m'} P[\boldsymbol{x}_i; \Lambda_{m'}^{c^*}]$ and update component
              $m$'s parameters;
        else
            update component $j^*$'s parameters on $\mathcal{X}_{j^*}^{c^*}$ ;
while $\sum_{c,j} \Delta\text{BIC}_j^c < 0$;
$\widehat{\mathcal{D}}_{\text{Train}} = \{\boldsymbol{x}_i \in \mathcal{D}_{\text{Train}} | t_i = y_i\}$;
```

5. EXPERIMENTS

5.1. Experiment Setup

Dataset and mixture model: For binary ($W = 2$) classification, we use the TREC 2005 spam corpus (TREC05) [4], with 9000 ham and spam emails for training and 3000 each for testing. The remaining samples are used for poisoning. For multi-class ($W > 2$) classification, we use the first 5 classes of the CIFAR10 [15] dataset. In each class, 4000 images are for training, 1000 for testing, and 800 for poisoning.

For TREC05, we represent each email using a bag-of-words and apply Parsimonious Mixture Modeling (PMM) [11] on each class. Each PMM component is a multinomial joint PMF. For CIFAR10, we trained Gaussian mixture models (GMMs) on the 512-dimensional feature vectors extracted from the penultimate layer of a pre-trained NN classifier. To reduce the model complexity, we assumed the features are independent conditioned on the mixture component of origin.

DP attack and target classifiers: We launched label flipping data poisoning attacks (LPDP) [1] on both datasets. For half of the binary classification attacks, we only poisoned the spam set, randomly selecting ham samples and mislabeling them as spam. For the remaining attacks, we simultaneously poisoned both ham and spam with varying attack strengths (**AS**, denoted as (number of poisoned ham, number of poisoned spam)), as seen in Tab. 1 and 2. For multi-class classification, we launched 5 DP attacks. For each attack $i = 1, \dots, 5$, we took samples from classes $c = 1, \dots, i$ and randomly mislabeled them to any of the other classes.

We chose linear SVM [5] and bi-directional one-layer long short-term memory (LSTM) [13] recurrent neural networks with 128 hidden units as the target classifiers for TREC05. We chose ResNet-18 [12] for CIFAR10.

Evaluation criteria: We evaluated our defense (**BIC-D**) based on: 1) improvement in test accuracy post-sanitization; 2) true positive rate (TPR) — the fraction of poisoned samples that are detected; and 3) false positive rate (FPR) – the fraction of non-poisoned samples falsely detected.

Hyperparameter setting: On TREC05, we also applied the KNN-based defense (**KNN-D**) [25], GS-based defense (**GS-D**) [14], and BIC-based defense with clean data (**BIC-C-D**) [19]. The SVD-based defense (**SVD-D**) was infeasible due to high computational cost. For multi-class classification, BIC-C-D was not applied, since it was proposed for binary classification. For ResNet models, we applied SVD-D on the output layer. We used hyper-parameter values suggested in the original papers for these methods. To show the best performance for SVD-D, we set the number of suspicious samples to be removed to the actual poisoning rate for poisoned sets, and to 0.01 for non-poisoned sets. We also applied **DPA** [17] on CIFAR-10. To improve performance against LPDP, DPA first trains the base classifiers (ResNets) on the whole training set by semi-supervised learning, where the DNN is trained to predict the rotation angles of the images instead of their categories [10]. Then each base classifier is fine-tuned on its dedicated partition through supervised learning. To reduce computational cost, we set the number of partitions to 5.

5.2. Experimental Results

AS	Poisoned	KNN-D	GS-D	BIC-C-D	BIC-D
0,0	0.9522	0.9001	0.9645	0.9579	**0.9684**
0, 1000	0.8867	0.8974	0.9372	0.9434	**0.9611**
0, 2000	0.8461	0.8828	0.9225	0.9124	**0.9530**
0, 3000	0.8215	0.8660	0.9023	0.8519	**0.9425**
0, 4000	0.7932	0.8358	0.8131	0.6882	**0.9411**
0, 5000	0.7731	0.7958	0.7042	0.6039	**0.9394**
0, 6000	0.7495	0.7751	0.6314	0.5697	**0.9329**
1000, 1000	0.8339	0.9049	0.9129	0.9217	**0.9454**
1000, 2000	0.7924	0.8917	0.8807	0.9088	**0.9284**
2000, 1000	0.7833	0.8880	0.8738	0.9061	**0.9429**
2000, 2000	0.7488	0.8793	0.8568	0.8288	**0.9143**
2000, 4000	0.7142	0.8421	0.8159	0.6385	**0.8998**
4000, 2000	0.7114	0.8367	0.7711	0.7153	**0.8731**

Table 1: Test accuracy of SVM classifiers as a function of attack strength on TREC05.

Results on binary classifiers: Tab. 1 and 2 show clean baselines (Attack (0,0)) and poisoned classifiers' accuracies. As total AS is strengthened to 6000, the classification accuracies of SVM and LSTM drop notably, demonstrating the effectiveness of the attacks. Then, we applied the four defenses on the corrupted datasets and trained classifiers on

AS	Poisoned	KNN-D	GS-D	BIC-C-D	BIC-D
0,0	0.9632	0.9313	0.8339	0.9629	**0.9701**
0, 1000	0.9363	0.9281	0.8205	0.9607	**0.9682**
0, 2000	0.9111	0.9183	0.8123	0.9217	**0.9619**
0, 3000	0.8852	0.8941	0.7792	0.8712	**0.9588**
0, 4000	0.8668	0.8744	0.7347	0.6915	**0.9513**
0, 5000	0.8159	0.8449	0.7153	0.6149	**0.9465**
0, 6000	0.8028	0.8009	0.6824	0.5906	**0.9424**
1000, 1000	0.8788	0.9317	0.8383	0.9359	**0.9584**
1000, 2000	0.8681	0.9131	0.8176	0.9232	**0.9476**
2000, 1000	0.8691	0.9013	0.8198	0.9277	**0.9551**
2000, 2000	0.8521	0.9125	0.8208	0.8404	**0.9431**
2000, 4000	0.7738	0.8905	0.7718	0.6514	**0.9223**
4000, 2000	0.7985	0.8821	0.7949	0.7368	**0.8991**

Table 2: Test accuracy of LSTM classifiers as a function of attack strength on TREC05.

AS	KNN-D	BIC-C-D	BIC-D
0,0	-/0.0745	-/0.0505	-/**0.0177**
0, 1000	0.8393/0.0826	0.8846/0.0724	**0.8898/0.0249**
0, 2000	0.8154/0.0936	0.8340/**0.0775**	**0.9044**/0.0841
0, 3000	0.7856/0.1095	0.7303/0.0881	**0.9036/0.0877**
0, 4000	0.7342/0.1377	0.3644/0.3209	**0.8689/0.0553**
0, 5000	0.6478/0.1798	0.1951/0.3621	**0.9014/0.0885**
0, 6000	0.5761/0.2122	0.1122/0.3868	**0.8865/0.0652**
1000, 1000	**0.8996**/0.0888	0.8628/0.0598	0.8633/**0.0499**
1000, 2000	0.8518/0.1057	0.8420/0.0681	**0.8678/0.0629**
2000, 1000	**0.9082**/0.1012	0.8284/0.0630	0.8874/**0.0586**
2000, 2000	**0.8842**/0.1099	0.7446/0.2128	0.8351/**0.0737**
2000, 4000	**0.8362**/0.1339	0.2102/0.2958	0.8113/**0.0809**
4000, 2000	**0.8261**/0.1452	0.4390/0.2698	0.8142/**0.1131**

Table 3: TPR/FPR of three defenses on TREC05.

the sanitized datasets[4]. The performance of BIC-C-D drops rapidly with AS over 4000. The test accuracies of KNN-D/GS-D decline gradually as AS increases (GS-D performs even worse than the poisoned classifiers under strong attacks and for LSTMs[5]), while our BIC-D performs stably and outperforms the others in all cases (marked in bold). Tab. 3 displays TPRs and FPRs for BIC-D, BIC-C-D, and KNN-D[6]. Our defense exhibits lower FPRs and higher or comparable TPRs than KNN-D and BIC-C-D across all attacks. When the AS exceeds 4000, BIC-C-D fails to detect poisoned samples and falsely removes clean samples.

Results on multi-class classifiers: Tab. 4 shows the clean baseline (Attack 0) and poisoned classifier's accuracies. As the number of poisoned classes increases, the test accuracy drops by over 10%, demonstrating the effectiveness of the attacks. Similar to the results for binary classifiers, our defense outperforms the other three defenses in classification accuracy (marked in bold) in all attacking cases, excluding attack 0 (attack-free). When there is no poisoning SVD-D performs the best, as we set a small number of suspicious samples for removal, which is difficult to set accurately in practice without a clean validation set. When DP is present, KNN-D/GS-D perform even worse than the poisoned classifier. SVD-D only improves the test accuracies by 4% at most. Due to fine-tuning base classifiers on small partitions which are individually inadequate for learning a precise DNN, DPA performs worse than the single poisoned classifier under weak attacks. However, it offers a slight improvement in the test accuracy (3% compared with the single poisoned model) under attack 4 and 5, as ensemble models tend to be more robust and better at handling noise than single models. By contrast, the test accuracy for our method drops by only 2% under the strongest attack, compared with the clean baseline. Tab. 5 shows the TPRs and FPRs of the defenses. Compared with the other two defenses, our defense has relatively high TPRs and low FPRs for all cases. KNN-D falsely detects lots of clean samples in all attack cases, even when there is no poisoning, and SVD-D only detects a small amount of poisoned samples.

[4]For BIC-C-D [19], we alternately apply BIC-C-D on ham and spam until the total BIC cost over the two classes converges.

[5]We used the settings from [14], which were tuned for a logistic regression classifier.

[6]GS-D does not identify the poisoned samples.

Defense	0	1	2	3	4	5
No defense	0.8634	0.8502	0.8302	0.7974	0.7634	0.7430
BIC-D	0.8638	**0.8616**	**0.8528**	**0.8452**	**0.8446**	**0.8416**
KNN-D	0.7150	0.7154	0.6688	0.6758	0.6602	0.6752
GS-D	0.8272	0.8074	0.7866	0.7288	0.7036	0.6852
SVD-D	**0.8668**	0.8584	0.8466	0.8164	0.8046	0.7812
DPA	0.8044	0.8028	0.7971	0.7958	0.7852	0.7782

Table 4: Test accuracy of ResNet-18 as a function of attack strength on CIFAR-10.

Attack	0	1	2	3	4	5
			True Positive Rates (TPRs)			
BIC-D	-	**0.9275**	**0.9263**	**0.9133**	**0.9378**	**0.9290**
KNN-D	-	0.9025	0.8050	0.8112	0.7922	0.8010
SVD-D	-	0.3650	0.2662	0.3587	0.4171	0.3655
			False Positive Rates (FPRs)			
BIC-D	0.0267	0.0494	0.0717	0.0881	0.1405	0.1626
KNN-D	0.4596	0.4628	0.4514	0.4533	0.4545	0.4484
SVD-D	**0.0100**	**0.0254**	**0.0587**	**0.0769**	**0.0932**	**0.1269**

Table 5: TPR and FPR of three defenses on CIFAR-10.

For both binary and multi-class classification, our method falsely removes a few clean samples from the attack-free datasets. These samples are well-explained by more than one class[7], and it is BIC-efficacious to re-distribute these samples. Removing these samples also slightly *increases* test accuracy.

6. CONCLUSION

We proposed an *unsupervised* BIC-based mixture model defense against DP attacks on classifiers, where the poisoned samples (which may originate from more than one class), if present, are an unknown subset of the data set. Our defense utilizes mixture modeling to isolate suspicious samples and solves outlier detection by minimizing BIC. We launched our

[7]For example, the clean samples falsely removed from TREC05 have similar average log likelihoods under ham (-879.77) and spam (-852.88). An SVM on a "perfectly sanitized" dataset (*i.e.*, the clean dataset without these samples) classifies these samples with only 0.5855 accuracy.

defense and three other defenses against DP attacks targeting SVM and NN-based classifiers for TREC05 and CIFAR10. Experiments demonstrate the effectiveness and robustness of our defense under strong attacks, as well as superiority over the other defenses.

7. REFERENCES

[1] B. Biggio, B. Nelson, and P. Laskov. Support vector machines under adversarial label noise. In *ACML*, 2011.

[2] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proc. ACM CCS*, 2018.

[3] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu. De-pois: An attack-agnostic defense against data poisoning attacks. *IEEE Trans. Inf. Forensics Secur.*, 16:3412–3425, 2021.

[4] G. V. Cormack and T. R. Lynam. Trec 2005 spam public corpora. `https://plg.uwaterloo.ca/~gvcormac/trecspamtrack05`, 2005.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 1995.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc.*, 1977.

[7] I. Diakonikolas, G. Kamath, D. Kane, J. Li, J. Steinhardt, and A. Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *Proc. ICML*, 2019.

[8] R. Duda, P. Hart, and D. Stork. *Pattern Classification, Second Edition*. 1999.

[9] J. Feng, H. Xu, S. Mannor, and S. Yan. Robust logistic regression and classification. In *Proc. NeurIPS*, 2014.

[10] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018.

[11] M. W. Graham and D. J. Miller. Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. *IEEE Trans. Signal Process.*, 2006.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[14] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitras, and N. Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv:2002.11497*, 2020.

[15] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`, 2009.

[16] A. D. Lanterman. Schwarz, Wallace, and Rissanen: Intertwining Themes in Theories of Model Selection. *International Statistical Review*, 2001.

[17] A. Levine and S. Feizi. Deep partition aggregation: Provable defenses against general poisoning attacks. In *Proc. ICLR*, 2021.

[18] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Proc. NeurIPS*, Dec. 2016.

[19] X. Li, D. J. Miller, Z. Xiang, and G. Kesidis. A scalable mixture model based defense against data poisoning attacks on classifiers. In *Proc. DDDAS*, 2020.

[20] Y. Ma, X. Zhu, and J. Hsu. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *Proc. IJCAI*, 2019.

[21] G. McLachlan and D. Peel. *Finite mixture models*. Wiley, 2004.

[22] D. J. Miller, Z. Xiang, and G. Kesidis. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 2020.

[23] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proc. ACM AISec*, 2017.

[24] B. Nelson, M. Barreno, F. Jack Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Misleading learners: Co-opting your spam filter. In *Proc. Machine Learning in Cyber Trust: Security, Privacy, and Reliability*, 2009.

[25] A. Paudice, L. Muñoz-González, and E. C. Lupu. Label sanitization against label flipping poisoning attacks. In *Proc. ECML PKDD Workshops*, 2018.

[26] G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 1978.

[27] W. Wang, A. Levine, and S. Feizi. Improved certified defenses against data poisoning with (deterministic) finite aggregation. In *Proc. ICML*, 2022.

[28] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 2015.