

A BIC-based Mixture Model Defense against Data Poisoning Attacks on Classifiers

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

School of EECS, Pennsylvania State University, University Park, PA, 16802, USA
{xzl45, djm25, zux49, gik2}@psu.edu

May 13, 2022

Abstract

Data Poisoning (DP) is an effective attack that causes trained classifiers to misclassify their inputs. DP attacks significantly degrade a classifier’s accuracy by covertly injecting attack samples into the training set. Broadly applicable to different classifier structures, without strong assumptions about the attacker, an *unsupervised* Bayesian Information Criterion (BIC)-based mixture model defense against “error generic” DP attacks is herein proposed 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 both to well-fit potentially multi-modal class distributions and to 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, with the identified poisoned data removed prior to classifier training. Our experimental results, for various classifier structures and benchmark datasets, demonstrate the effectiveness and universality of our defense under strong DP attacks, as well as its superiority over other works.

1 Introduction

Learning-based models have shown impressive performance in various domains, *e.g.*, computer vision and natural language processing. However, machine learning systems are vulnerable to maliciously crafted inputs. Interest in Adversarial Learning (AL) has grown dramatically in recent years, focused on devising attacks against machine learning models and defenses against same. Three important types of AL attacks are [31]: data poisoning (*e.g.*, [6, 46, 28, 24, 1, 16, 27]), test-time evasion (*e.g.*, [3, 37, 25]), and reverse engineering (*e.g.*, [35, 36]). In this work, we address Data Poisoning (DP) attacks against models trained for classification tasks.

DP attacks are training-time attacks which involve the insertion of “poisoned” samples into the training set of a classifier. In this paper, we address the so-called “error generic” DP attacks (hereafter called DP attacks) [4] which aim to degrade the *overall* classification accuracy¹. To effectively mislead classifier training using relatively few poisoned samples, an attacker introduces feature collision[19] to the training samples by *e.g.* flipping the class labels of clean samples. The information extracted from the clean and poisoned samples labeled to the same class contradicts each other and prevents the learning of an accurate class decision boundary. DP attacks have been successfully demonstrated against Support Vector Machines (SVMs) [46], Logistic Regression (LR) models [14], auto-regressive models [1], collaborative filtering systems [24], differentially-private learners [28], and neural networks (NN) [32].

In the most general setting (and also the most challenging one for the defender), considered here, the poisoned samples, *if there is data poisoning*, are an *unknown* subset embedded among the clean training

¹Error specific attacks, particularly backdoor attacks involving specific backdoor patterns and source and target classes, *e.g.*, [6, 27, 16], are not the focus herein.

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. This *embedded data poisoning* attack scenario is of great practical interest, and yet remains largely unsolved. Studies on defending against such attacks either are tailored to a specific type of classifier (*e.g.*, SVM [21], LR [14]), are only suitable for a specific type of DP attack (*e.g.*, [38] only defends against label flipping attacks), or make strong assumptions about the training data (*e.g.*, availability of a clean validation set for use by the defender [34]). The proposed method does not make any such assumptions about the attack, does not require a clean (attack-free) validation set, and can be deployed to protect various types of classifiers.

Poisoned samples are generally *atypical* of the distribution of the class to which they are labeled. We thus apply mixture modeling [13, 29] to accurately explain the potentially multi-modal data and to capture poisoned samples within a subset of mixture components. We make the following observations. If the poisoned samples are typical of another class (different from the class to which they are labeled), we expect that re-distributing them to other classes should increase the overall data likelihood. Furthermore, removing a poisoned component will reduce the model complexity of a mixture. Thus, both the data likelihood and model complexity terms that constitute the Bayesian Information Criterion (BIC) objective function [41] should improve when data poisoning is mitigated. Accordingly, we propose to make poisoned sample inferences consistent with minimizing BIC. We first apply mixture modelling separately to each class, with the number of components chosen to minimize the BIC criterion. Then we assess components for possible poisoning, with a detected component either removed or revised (whichever results in a lower BIC cost). After poisoned samples have been detected and removed, the classifier is trained on the resulting (sanitized) data set.

In summary, our BIC-based mixture model defense is:

- *Novel*: As far as we know, we are the first to formulate a BIC-based defense for *unsupervised* anomaly detection/DP attack mitigation.
- *Practical*: We address the practical and challenging embedded DP attack scenario, and wherein more than one class may be poisoned. Moreover, our defender does not require a clean validation set.
- *Robust*: The experimental results on several benchmark datasets demonstrate the effectiveness of our defense under strong DP attacks, as well as the superiority over other works. In principle, our method can be deployed to defend against any error-generic DP attack.
- *Comprehensive*: Our approach sanitizes the training set (prior to training) and is applicable to a wide variety of classifier structures, including SVMs, LR and NN-based classifiers, as will be seen in the sequel.

The rest of the paper is organized as follows: We first review several existing DP studies in Section 2. Then we define our threat model for DP attacks in Section 3. In Section 4, we propose our BIC-based mixture model defense. Experimental results on binary classification tasks and multi-class classification tasks are presented in Section 5 and Section 6, respectively. Finally, we summarize our work in Section 8.

2 Related Work

An obvious strategy for defending against data poisoning attacks is to conduct “data sanitization” on the training set, *i.e.*, identifying and cleansing the attack samples as training set outliers. Outlier identification is divided into two types – supervised and unsupervised. Some supervised detection methods are in fact a form of supervised learning – the defender trains a binary discriminator based on labeled examples of anomalies and normalities, *e.g.*, [47]. The resulting discriminator can then be applied to identify and sanitize anomalies (possibly poisoned examples) in the classifier’s training set. However, such a learned discriminator may only reliably identify anomalies in a data set that are similar to those

seen during the discriminator’s training – *i.e.*, the discriminator may only be good at identifying *known* anomalies, not *unknown* anomalies. Another issue with this approach is that anomalies (attack instances) may be rarer and more difficult to collect than “normalities”. Thus, the performance of the discriminator may suffer from highly skewed class imbalance (“anomalous” vs. “normal”) in the supervised examples used to train it.

Other supervised detection methods are more akin to *unsupervised* anomaly detection methods, except that they possess hyper-parameters whose setting, to achieve good detection performance, requires either a clean validation set (which is anathema to the embedded DP scenario considered here) or, again, a labeled set of “normalities” and “anomalies”, *e.g.*, [42, 39, 44, 38, 12, 19]. On the other hand, truly unsupervised detection methods do not require labeled examples of what is normal and what is anomalous, and are analogous to unsupervised clustering methods. They model data distributions and flag potential outliers, *e.g.*, [5, 26, 45, 48] and our method proposed herein. The false positive rate (*i.e.*, the fraction of normal samples misidentified as outliers) and false negative rate (*i.e.*, the fraction of true outliers misidentified as normalities) may be relatively high for an unsupervised detector. Thus, [5] proposed to leverage human intelligence to correctly identify outliers from amongst a set of outlier candidates detected by a machine-learning method. However, such an approach is only suitable for domains where humans are skilled at analyzing data – images, speech, text, low-dimensional data domains, and/or high-dimensional ones that can be visualized with fidelity in a low-dimensional space. Moreover, such an approach may be very costly and time-consuming, as there may be a large set of outlier candidates for the human to inspect.

Due to the near-limitless space of possible attacks, in practice there will always be *unknown* attacks, ones which have not been encountered before (and thus, with no labeled examples available for training of a supervised detector). For various types of adversarial attacks (test-time evasion attacks, as well as data poisoning attacks), it has been observed that the performance of supervised detectors may fare poorly on unknown attacks [31]. On the theoretical front, there are works such as [44, 39]. Given a defender that first performs outlier removal followed by margin-based loss minimization, [44] generates an approximate upper bound on the efficacy of any DP attack. They also established a dual method which generates an attack that nearly achieves this upper bound. However, their attack requires full knowledge of the clean training set (prior to its poisoning) and cannot handle non-convex loss functions, which limits its application in practice. [39] proposes a method for producing certificates of robustness for two-layer neural networks. Such certificates are differentiable and can be jointly optimized with the network parameters, providing an adaptive regularizer that encourages robustness against attacks. However, they neither characterize nor discuss the inherent tradeoff between robustness and model bias (*i.e.*, the degradation in the classifier’s accuracy on *clean* data that results from making the classifier robust to attacks).

[38] proposed a label sanitization strategy tailored to label flipping attacks. The poisoned samples are expected to be outliers relative to untainted samples with the same labels. Thus, they relabel a sample based on the plurality label of its K nearest neighbors (KNN) to enforce label homogeneity. However, this defense will fail when the number of poisoned samples is sufficiently large such that some of the neighbors of an attack sample are also attack samples. This defense also relies on the availability of a clean validation set to tune the hyper-parameter K . This choice highly impacts the detector’s performance, as will be seen (*cf.* Section 5 and 6).

[12, 19] proposed to defend DP attacks by analyzing training sample gradients (measured with respect to the loss function used for classifier training). They posited a unified view of effects of DP on learned classifier parameters: (1) the l_2 norm of the gradient from a poisoned sample is larger than that of a clean sample, on average; (2) there is an orientation difference between poisoned and clean sample gradients. [12] detects such effects by singular value decomposition (SVD) applied to the matrix of gradients, with each row of this matrix the gradient, with respect to the model parameters, of one sample’s contribution to the training loss function. They derive an outlier score for each training sample, which is the squared magnitude of the projection of the gradient onto the top right singular vector. For classification tasks, [12] separately constructs the matrix of gradients for the training samples from each class, and computes outlier scores for each class, to improve performance. At each detection step, the top $\frac{\epsilon}{\beta}$ fraction of

samples with highest scores over all classes are removed, where β is the total number of detection steps and ϵ is the fraction of samples to be ultimately removed (after β steps); the classifier is then retrained on the remaining samples. The performance of their detector sensitively depends on the choice of the hyperparameters β and ϵ , as well as on the chosen training loss function. Note that there is no *a priori* knowledge of a good choice for ϵ . Also, their method is only applicable to linear classifiers, *e.g.*, an SVM. Besides, they only report the improvement in the test set error rate – the false positive rate is not mentioned, even though this is an important criterion for assessing a defense against DP attacks. [12] is computationally expensive as it requires performing an SVD for each class, at each detection step, and retraining the classifier after each detection step.

[19] mitigates the effects of DP by gradient shaping (GS), *i.e.*, constraining the magnitude and orientation of poisoned gradients to make them close to clean gradients. For example, one can adopt a differentially-private mechanism based optimizer (*e.g.* differentially-private stochastic gradient descent (DP-SGD)) in training. A DP-SGD optimizer clips gradients according to the hyper-parameter *l2_norm_clip* and then adds noise to the gradients, whose size is controlled by the hyper-parameter *noise_multiplier*. [19] is computationally cheap – it does not require extra computation pre-training/post-training. However, their method only reduces the effect of poisoning, rather than eliminating the poisoned samples. Efficacy of their defense is dramatically degraded as more and more attack samples are injected, as will be seen from our results.

[26] applied a BIC-based defense against DP attacks for binary classification tasks. The fundamental difference between [26] and our work pertains to *untainted data availability*. Their method assumes that the attacker only poisons one of the two classes, with this class known to the defender. Thus, the defender can always take the clean class as reference in helping to identify poisoned samples in the corrupted class (This should be especially helpful for label flipping attacks, where the poisoned samples, labeled to one class, will be typical of the other (clean) class). However, in practice, the attacker is able to poison more than one class, and the defender does not know which class(es) are poisoned. Under this most realistic attack scenario, [26] may fail even if only one class is poisoned, as the defender might sanitize the clean class based on the poisoned one (*cf.* Section 5).

[38, 19, 12] are supervised detection methods, with their performances highly impacted by the choices of hyper-parameters. By contrast, our method is unsupervised. At each optimization step, it separately assesses the hypothesis that each individual mixture component, in each class, is poisoned. Only the component whose trial-sanitization yields the lowest BIC cost is actually sanitized. This process is repeated until the total BIC cost, defined over all classes, converges.

3 Threat Model

In this paper, we consider W -class ($W \geq 2$) classification tasks, where the classifier, denoted

$$f : \mathbb{R}^d \rightarrow \{1, \dots, W\},$$

is trained on $\mathcal{D}_{\text{Train}}$ and then tested on $\mathcal{D}_{\text{Test}}$. $\mathcal{D}_{\text{Train}}$ and $\mathcal{D}_{\text{Test}}$ are assumed i.i.d. from the same (unknown) distribution. Each feature x_l , $l = 1, \dots, d$, may be either discrete or continuous-valued. Both continuous and discrete-valued feature spaces will be considered in our experimental results.

We assume the attacker: 1) has sufficient knowledge of the classification domain to generate or acquire samples that are legitimate instances of the different classes; 2) has access to covertly insert poisoned samples into the training set ($\mathcal{D}_{\text{Train}} = \mathcal{D}_{\text{Clean}} \cup \mathcal{D}_{\text{Attack}}$); 3) May simultaneously poison any subset of the classes, possibly with different attack strengths (*i.e.*, different amounts of poisoned samples) for individual classes; 4) is unaware of any deployed defense. The *goal* of the attacker is to degrade the classifier’s (test set) generalization accuracy as much as possible, *i.e.*,

$$P_c = \frac{1}{|\mathcal{D}_{\text{Test}}|} \sum_{\mathbf{x}_i \in \mathcal{D}_{\text{Test}}} \mathbb{1}(f(\mathbf{x}_i), y_i),$$

where $\mathbb{1}(a, b)$ is an indicator function with value 1 when $a = b$ and value 0 otherwise.

We assume the defender: 1) can only use the training set $\mathcal{D}_{\text{Train}}$ manipulated by the attacker, not any additional samples known to be clean (attack-free) – this is in line with the embedded DP scenario; 2) does not know whether an attack is present, and if so, does not know the subset of attacking samples ($\mathcal{D}_{\text{Attack}}$), nor which class(es) 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 in so doing: 2) maintain classification accuracy as close as possible to that of a classifier trained on a clean (unpoisoned) data set.

4 BIC Mixture-based Sanitization Strategy

We apply a data sanitization strategy on the training set, *i.e.*, identifying and removing poisoned samples as training set outliers prior to classifier training. To accurately describe the possibly poisoned dataset, we apply mixture modeling to each class. Mixture modeling is a sound statistical approach for well-fitting potentially multi-modal data [13, 29] and also gives the potential for concentrating the poisoned samples into just a few components, which assists in accurately identifying and removing them. Note that, in practice, poisoned components may own both poisoned and untainted samples, with the poisoning ratio for each component unknown.

An attacker can confound the learning of class-discriminating features by choosing, for poisoning, samples typical of one class, but then labeling them to other class(es) (*i.e.*, a label flipping attack). *Accordingly, we would expect that such samples are better explained by the mixture model for a class other than the class to which they are labeled.* Thus, we propose to identify poisoned samples in the training set as those with greater *likelihood* under a class different from the one to which they are labeled. We effectively *re-assign* such samples to the class (and mixture component) under which they have the greatest likelihood.

Now, suppose the vast majority of a mixture component’s samples are re-assigned in this way to another class. In this case, there may be insufficient remaining samples to reliably (or even in a well-posed fashion²) estimate the component’s parameters. In such a case, rather than retaining this component, it may be better to *remove* it, with its remaining samples assigned to other components in the class’s mixture model. A principled, theoretically supported criterion for model order selection, suitable for use in deciding between revising and (wholesale) removing a mixture component from a class’s model, is the *Bayesian Information Criterion* (BIC), which expresses an inherent tradeoff between data likelihood fit and model complexity. The BIC objective function for a given data set \mathcal{D} is defined as:

$$\text{BIC} = |\theta|k - L(\mathcal{D}; \theta), \quad (1)$$

where θ is the set of free 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 $L(\mathcal{D}; \theta)$ is the log-likelihood of the data set \mathcal{D} , based on the density function model. In [41], under suitable assumptions, BIC is shown to be a consistent estimator of model order. In [23], within an approximate Bayesian setting, the BIC cost for describing an individual model parameter is derived, and found to be $k = 0.5 * \log(|\mathcal{D}|)$. This model penalty will be used in the following.

The form of the BIC objective seen above is equivalent to the minimum description length (MDL) [40], and is amenable to interpretation as a two-part codelength: i) the first term, which we will denote by Ω in the sequel, is the number of bits needed to describe the parameters of the density model for the data; ii) the second, negative log-likelihood term is the number of bits to describe the data set, given the model.

In this work, we model the training data labeled to each class by a class-specific mixture of density functions (or probability mass functions in the case of discrete data), *i.e.*, for an individual sample \mathbf{x} ,

²For example, for a multivariate Gaussian component, with a full covariance matrix of size $d \times d$, one needs at least d samples to estimate a (full-rank) covariance matrix.

labeled to class c , its density (likelihood) is:

$$P[\mathbf{x}; \theta_c] = \sum_{j=1}^{M_c} \alpha_j^c P[\mathbf{x}; \Lambda_j^c],$$

where α_j^c is the probability mass (prior probability) of mixture component j for class c (i.e., $\sum_{j=1}^{M_c} \alpha_j^c = 1$, $\alpha_j^c \geq 0 \forall j$), $P[\cdot; \Lambda_j^c]$ is the j^{th} mixture component density under class c , Λ_j^c is the set of parameters specifying the component density, and M_c is the number of mixture components in the density for class c . Note that $\theta_c = \{\Lambda_j^c\} \cup \{\alpha_j^c\}$ and $\theta = \bigcup_c \theta_c$.

In the above, a data sample from class c is associated probabilistically with all mixture components from the class's mixture density. Alternatively, in this work, we consider the *complete data* BIC objective function, based on the complete data log-likelihood function [10], wherein each data sample is hard (fully) assigned to the mixture component under which it has the greatest log-likelihood. That is, the complete data log-likelihood for the data from class c is³

$$L_j^c = \sum_{\mathbf{x} \in \mathcal{X}_j^c} \log P[\mathbf{x}; \Lambda_j^c],$$

where $\mathbf{x} \in \mathcal{X}_j^c$ if and only if, for \mathbf{x} labeled to class c , $P[\mathbf{x}; \Lambda_j^c] \geq P[\mathbf{x}; \Lambda_{j'}^c] \forall j' \neq j$.

Likewise the complete data BIC objective is:

$$\text{BIC}_{\text{cmplt}} = |\theta|k - \sum_{c=1}^W \sum_{j=1}^{M_c} L_j^c. \quad (2)$$

In short, the principle behind our complete data BIC-based defense is: *a component is identified as poisoned if removing or revising it and re-assigning its samples reduces the BIC cost; moreover, samples which are redistributed to other class(es) are deemed poisoned*. Thus, our anomaly detection method is *unsupervised* and consistent with solving a BIC minimization problem⁴. In the sequel, we develop an algorithm for mitigating data poisoning via (locally optimal) minimization of the above complete data BIC objective.

4.1 BIC-based Defense

Recall $\{1, \dots, W\}$, $W \geq 2$, is the set of classes, and let $T = |\mathcal{D}_{\text{Train}}|$ be the total number of training samples. Let $(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{1, \dots, W\}$ represent the feature vector and label of 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 , with each specifying, e.g., a joint probability mass function (PMF) or probability density function (PDF) for component j of class c , depending on whether the data is discrete or continuous-valued, respectively.

The model parameters θ_c of the mixture for class c are estimated via the Expectation-Maximization (EM) algorithm [10], 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 (1) over the set $\{1, \dots, M_{\text{max}}^c\}$ [41], with M_{max}^c an upper bound on the number of components in class c 's mixture⁵. Finally, we let $\mathcal{S} = \{(c, j) | c = 1, \dots, W, j = 1, \dots, M^c\}$ be the set of components across all classes.

As just mentioned, we first estimate a mixture model for each class and then identify poisoned components and samples by minimizing the complete data BIC cost (2) defined in the last section. This BIC

³Here we assume the component priors are uniform and hence they are absent from the complete data log likelihood. In practice, these terms do not affect detection performance significantly.

⁴Apart from poisoned samples, our method might also remove any outliers, if it is BIC-efficacious to do so.

⁵ M_{max}^c is in fact not a hyperparameter, as one can observe the changes of BIC to adjust the range of model orders. For example, if M_{max}^c yields the least BIC, one can increase M_{max}^c and repeat model selection until $M^c \neq M_{\text{max}}^c$.

minimization involves sample re-distribution, component removal/revision, and parameter updates. To reflect these model changes, we introduce several types of “indicator” variables:

- The class t_i and component under this class j_i that best-explain sample \mathbf{x}_i are

$$(t_i, j_i) = \arg \max_{t=\{1,\dots,W\}, j=\{1,\dots,M^c\}} P[\mathbf{x}_i; \Lambda_j^t],$$

- $r_j^c = \begin{cases} 1 & \text{component } j \text{ in class } c \text{ is poisoned} \\ 0 & \text{else} \end{cases}$
- $q_j^c = \begin{cases} 1 & \text{component } j \text{ in class } c \text{ needs to be revised} \\ 0 & \text{component } j \text{ in class } c \text{ needs to be removed} \end{cases}$

Note that q_j^c is configured only when $r_j^c = 1$.

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

$$\begin{aligned} \text{BIC}_{\text{cmplt}}(\theta) &= \sum_{c=1}^W \sum_{j=1}^{M_c} ((1 - r_j^c(1 - q_j^c))k|\Lambda_j^c| + 1 + \delta(r_j^c, 1)) \\ &- \sum_{c=1}^W \sum_{j=1}^{M_c} ((1 - r_j^c)L_j^c(\Lambda_j^c) + r_j^c \sum_{\mathbf{x}_i \in \mathcal{X}_j^c} \log P[\mathbf{x}_i; \Lambda_{j_i}^{t_i}]). \end{aligned} \quad (3)$$

In (3), the model parameters are $\theta = \{\{\Lambda_j^c\}, \{r_j^c\}, \{q_j^c\}\}$, where the structural parameters r_j^c and q_j^c each require one bit to specify (hence the ‘1’ and $\delta(r_j^c, 1)$ 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, and complete data BIC) not model parameters.

To minimize (3) in a locally optimal fashion, our approach will involve cycling over the mixture components, one at a time, effecting changes to the mixture models that reduce this objective. The new BIC cost, in light of changes to component j from class c , can be expressed as the “old” BIC cost plus the (negative) change resulting from sample re-assignments or component removal, denoted ΔBIC_j^c .

Each feasible joint configuration of the variables for component j in class c corresponds to one of three cases:

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

$$\Delta \text{BIC}_j^c = \Delta \Omega_{j,1}^c + \Delta L_{j,1}^c = 0.$$

2) $r_j^c = 1, q_j^c = 0$: Component j is poisoned, and we are choosing to remove it from the mixture, changing the model complexity term by

$$\Delta \Omega_{j,2}^c = -|\Lambda_j^c| \frac{1}{2} \log T,$$

where $|\Lambda_j^c|$ is the number of parameters in component j from class c . The component’s samples are re-distributed consistent with maximizing the log-likelihood: Each sample $\mathbf{x}_i \in \mathcal{X}_j^c$ is re-assigned to component j_i of class t_i , where

$$(t_i, j_i) = \arg \max_{(t,j') \in \mathcal{S} \setminus \{(c,j)\}} \log P[\mathbf{x}_i; \Lambda_{j'}^{t'}].$$

Let

$$\mathcal{Q} = \{(t_i, j_i) | \forall i, \mathbf{x}_i \in \mathcal{X}_j^c\}$$

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 $\widehat{\mathcal{X}}_{j'}^w$ by maximum likelihood estimation (MLE):

$$\Lambda_{j'}^{w, \text{new}} = \arg \max_{\Lambda} \sum_{\mathbf{x}_i \in \widehat{\mathcal{X}}_{j'}^w} \log P[\mathbf{x}_i; \Lambda],$$

where

$$\widehat{\mathcal{X}}_{j'}^w = \mathcal{X}_{j'}^w \cup \{\mathbf{x}_i \in \mathcal{X}_j^c | t_i = w, j_i = j'\}.$$

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

$$\begin{aligned} \Delta L_{j,2}^c = & - \sum_{(w,j') \in \mathcal{Q}} \sum_{\mathbf{x}_i \in \widehat{\mathcal{X}}_{j'}^w} \log P[\mathbf{x}_i; \Lambda_{j'}^{w, \text{new}}] \\ & + \sum_{(w,j') \in \mathcal{Q}} \sum_{\mathbf{x}_i \in \mathcal{X}_{j'}^w} \log P[\mathbf{x}_i; \Lambda_{j'}^w] + \sum_{\mathbf{x}_i \in \mathcal{X}_j^c} \log P[\mathbf{x}_i; \Lambda_j^c]. \end{aligned}$$

The change in BIC in this case is

$$\Delta \text{BIC}_j^c = \Delta \Omega_{j,2}^c + \Delta L_{j,2}^c.$$

3) $r_j^c = 1, q_j^c = 1$: 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, since the code length is untouched (*i.e.*, $\Delta \Omega_{j,3}^c = 0$). The parameters Λ_j^c are re-estimated by MLE on the surviving samples:

$$\Lambda_j^{c, \text{new}} = \arg \max_{\Lambda} \sum_{\mathbf{x}_i \in \widehat{\mathcal{X}}_j^c} \log P[\mathbf{x}_i; \Lambda],$$

where

$$\widehat{\mathcal{X}}_j^c = \{\mathbf{x}_i \in \mathcal{X}_j^c | t_i = c\}.$$

Samples that are best represented by class $w \neq c$ (*i.e.*, $t_i = w, w \neq c$) 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

$$\begin{aligned} \Delta L_{j,3}^c = & - \sum_{(w,j') \in \mathcal{Q}'} \sum_{\mathbf{x}_i \in \widehat{\mathcal{X}}_{j'}^w} \log P[\mathbf{x}_i; \Lambda_{j'}^{w, \text{new}}] \\ & + \sum_{(w,j') \in \mathcal{Q}'} \sum_{\mathbf{x}_i \in \mathcal{X}_{j'}^w} \log P[\mathbf{x}_i; \Lambda_{j'}^w], \end{aligned}$$

where $\widehat{\mathcal{X}}_{j'}^w$ and $\Lambda_{j'}^{w, \text{new}} \forall (w, j') \in \mathcal{Q}' \setminus \{(c, j)\}$ are defined in the same way as in case 2. The BIC change in this case is

$$\Delta \text{BIC}_j^c = \Delta L_{j,3}^c.$$

To minimize the complete data BIC objective, for each component in class $c \in \{1, \dots, W\}$, we should choose the configuration of the parameters that reduces BIC the most. However, 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 (*e.g.*, by exhaustively evaluating over the huge combinatorial space of component configurations). Instead, at each optimization

step, we separately *trial*-update each component’s configuration, and then only permanently update for 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.

The null hypothesis of our detection inference is that the training set is not poisoned (and is generated according to the existing mixture model). If there is data poisoning, the training set is hypothesized to be generated by an alternative model (with some components removed and/or modified). Thus, we perform the following hypothesis testing: after BIC minimization, if $r_j^c = 0$ holds for all components in all classes, and $t_i = y_i$ holds for all samples, then no components and samples are inferred to be poisoned, and we accept the null hypothesis. Otherwise, we reject the null hypothesis, and the training set is deemed poisoned. The samples that were re-assigned to other classes via the BIC minimization are deemed poisoned, and are removed from the training set.

4.2 Implementation

Consistent with the above description, we apply an iterative, locally optimal approach to minimize the total BIC cost and optimize its parameters, as shown in Algorithm 1. As we discussed before, at each algorithm step, we only sanitize the component whose removal/revision decreases the total BIC cost the most. That is, we first evaluate the reduction in the total BIC cost ΔBIC_j^c caused by trial removal/revision of each component j in each class c . Then, we sanitize the component j^* in class c^* which decreases the total BIC cost the most, *i.e.*,

$$(c^*, j^*) = \arg \min_{c \in \{1, \dots, W\}, j=1, \dots, M^c} \Delta\text{BIC}_j^c,$$

where

$$\Delta\text{BIC}_j^c = \min_{m=1,2,3} \{\Delta\Omega_{j,m}^c + \Delta L_{j,m}^c\}.$$

The above procedure is repeated until the total BIC cost converges, *i.e.* until no trial component updates further reduce the BIC cost. 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

$$\hat{\mathcal{D}}_{\text{Train}} = \{\mathbf{x}_i \in \mathcal{D}_{\text{Train}} | t_i = y_i\}.$$

Note that: 1) the same component may be re-optimized multiple times during the course of this algorithm; 2) But removal of a component is permanent, *i.e.* once removed, a component cannot be reinstated.

5 Experiments on Binary Classification Tasks

5.1 Experiment Setup

Dataset and mixture model: For binary ($W = 2$) classification, we use the TREC 2005 spam corpus (TREC05) [8]. TREC05 contains 39,999 real ham and 52,790 spam emails which are labeled based on the sender/receiver relationship. For training, we randomly select 9000 ham emails and 9000 spam emails⁶. For testing, we randomly select 3000 ham emails and 3000 spam emails. The remaining samples are used for poisoning. The dictionary, following case normalization, stop word removal, stemming and low-frequency word filtering, has about 30,000 unique words.

We apply Parsimonious Mixture Modeling (PMM) [15] on both datasets. PMMs allow parameter sharing across multiple components, which greatly reduces the number of model parameters compared with standard (unstructured) mixtures, and which allows BIC to choose good model orders in high feature dimensions, rather than grossly underestimating the model order (number of mixture components).

⁶There are 8651 ham emails and 8835 spam emails remaining in the training set after pre-processing. Some emails are removed since there are no tokens left after *e.g.*, stop word removal and low-frequency word filtering.

Algorithm 1: BIC-Based Defense Against DP Attacks

Input : $\mathcal{D}_{\text{Train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \{\{\Lambda_j^c\}_{j=1}^{M^c}\}_{c \in \{1, \dots, W\}}$
Output: $\hat{\mathcal{D}}_{\text{Train}}$
 $r_j^c = 0, q_j^c = 0, \forall c, j$;
 $t_i = y_i, \forall i$;
 $\Delta \text{BIC}_j^c = 0, \forall j, c$;
do
 for $c \in \{1, \dots, W\}$ **do**
 for each component j **in class** c **do**
 compute BIC reduction from j for the three cases: $\{\Delta \Omega_{j,m}^c + \Delta L_{j,m}^c\}, m = 1, 2, 3$;
 configure $\{t_i \mid \mathbf{x}_i \in \mathcal{X}_j^c, r_j^c, q_j^c\}$ to $\min\{\Delta \Omega_{j,m}^c + \Delta L_{j,m}^c\}_{m=1}^3$;
 $\Delta \text{BIC}_j^c = \min\{\Delta \Omega_{j,m}^c + \Delta L_{j,m}^c\}_{m=1}^3$;
 $(c^*, j^*) = \arg \min_{c \in \{1, \dots, W\}, j \in \{1, \dots, M^c\}} \Delta \text{BIC}_j^c$;
 if $r_{j^*}^{c^*} = 1$ **then**
 For $\mathbf{x}_i \in \mathcal{X}_{j^*}^{c^*}$, if $t_i = w, w \neq c^*$, re-distribute it to component $m = \arg \max_{m'} \log P[\mathbf{x}_i; \Lambda_{m'}^w]$
 in class w 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 $\mathbf{x}_i \in \mathcal{X}_{j^*}^{c^*}$ to component $m = \arg \max_{m'} P[\mathbf{x}_i; \Lambda_{m'}^{c^*}]$ and update
 component m 's parameters;
 else
 update component j^* 's parameters on $\mathcal{X}_{j^*}^{c^*}$;
 while $\sum_{c,j} \Delta \text{BIC}_j^c < 0$;
 $\hat{\mathcal{D}}_{\text{Train}} = \{\mathbf{x}_i \in \mathcal{D}_{\text{Train}} \mid t_i = y_i\}$;

Figure 1 shows that, for class “soc.religion.christian” of the clean 20-Newsgroups dataset, PMM chooses a reasonable model order (14) that minimizes the training set BIC cost and also has good generalization (*i.e.*, test set log-likelihood). PMMs’ superiority over standard unstructured mixture models for high dimensional datasets such as text was demonstrated in [15]. Initially we chose $M_{\max}^c = 25, \forall c$. If the chosen model order $M^c = M_{\max}^c$, then M_{\max}^c is increased and the model is retrained. For both datasets, the training and test samples are represented using a bag-of-words. Each PMM component is a multinomial joint probability mass function.

DP attack and target classifiers: To simulate a reasonable and potent embedded data poisoning attack, we used real samples as attacking samples. That is, we inject samples from class c into class $w \neq c$ as poisoned samples, with these samples intentionally mislabeled to class w . The attack strength, *i.e.*, the number of attacking samples injected, may not be the same for each class. The poisoned samples are randomly selected (from the sample pool left over from the training and test sets).

We launched 12 poisoning attacks on TREC05. For half of the attacks, we only poisoned one class (*e.g.*, spam), with attack strength from 1000 to 6000 samples. For the other half of the attacks, we simultaneously poisoned the ham and spam training sets with various attack strengths (*cf.* Table1).

We chose linear SVM [9], LR [33], and bi-directional one-layer long short-term memory (LSTM) [18] recurrent neural networks with 128 hidden units as the target classifiers, since they are effective in the classification of text data.

Evaluation criteria: The performance of our defense (BIC-D) was measured by: 1) improvement in test classification accuracy after data 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

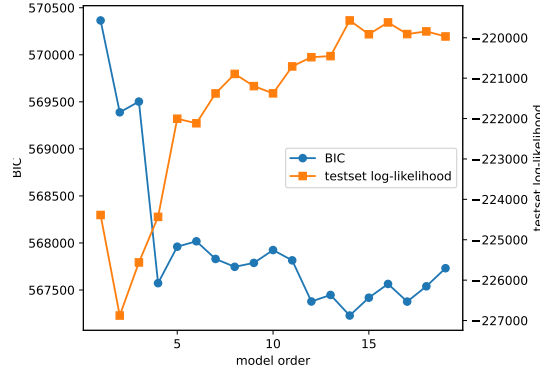


Figure 1: Training set BIC cost and test set log-likelihood as a function of the model order of the PMM on class “soc.religion.christian” of clean 20-Newsgroups dataset.

detected. We also applied the BIC-based defense with clean data samples (BIC-C-D) [26], a KNN-based defense (KNN-D) [38], and a GS-based defense (GS-D) [19] on the same poisoned training sets and compared with them in terms of the above evaluation criteria. We failed to apply the SVD-based defense [12] on TREC05, since it is too expensive to perform SVD on the matrix of gradients, whose size is around (20000 x 60000).

Hyperparameter setting: 1) For the KNN-based defense, we set the number of neighbors at $K = 10$, which is suggested by [38]. 2) For the GS-based defense, we applied a DP-SGD optimizer with clip-norm of 2.0 and noise multiplier of 0.1 for training – these values were used in [19].

5.2 Experimental Results

# Poisoned Ham # Poisoned Spam	0, 0	0, 1000	0, 2000	0, 3000	0, 4000	0, 5000	0, 6000	1000, 1000	1000, 2000	2000, 1000	2000, 2000	2000, 4000	4000, 2000
SVM													
Poisoned	0.9522	0.8867	0.8461	0.8215	0.7932	0.7731	0.7495	0.8339	0.7924	0.7833	0.7488	0.7142	0.7114
BIC-D	0.9684	0.9611	0.9530	0.9425	0.9411	0.9394	0.9329	0.9454	0.9284	0.9429	0.9143	0.8998	0.8731
KNN-D	0.9001	0.8974	0.8828	0.8660	0.8358	0.7958	0.7751	0.9049	0.8917	0.8880	0.8793	0.8421	0.8367
GS-D	0.9645	0.9372	0.9225	0.9023	0.8131	0.7042	0.6314	0.9129	0.8807	0.8738	0.8568	0.8159	0.7711
BIC-C-D	0.9579	0.9434	0.9124	0.8519	0.6882	0.6039	0.5697	0.9217	0.9088	0.9061	0.8288	0.6385	0.7153
LR													
Poisoned	0.9616	0.9175	0.8828	0.8443	0.8172	0.7803	0.7488	0.8843	0.8501	0.8481	0.8183	0.7591	0.7438
BIC-D	0.9699	0.9660	0.9559	0.9519	0.9461	0.9394	0.9368	0.9511	0.9402	0.9507	0.9315	0.9126	0.8799
KNN-D	0.9099	0.9073	0.8955	0.8831	0.8529	0.8069	0.7776	0.9169	0.9039	0.9044	0.8968	0.8646	0.8656
GS-D	0.9598	0.9384	0.9184	0.8606	0.8158	0.7099	0.6655	0.9258	0.9137	0.8948	0.8797	0.8091	0.7847
BIC-C-D	0.9622	0.9554	0.9247	0.8633	0.6909	0.6192	0.5801	0.9375	0.9222	0.9152	0.8358	0.6427	0.7158
LSTM													
Poisoned	0.9632	0.9363	0.9111	0.8852	0.8668	0.8159	0.8028	0.8788	0.8681	0.8691	0.8521	0.7738	0.7985
BIC-D	0.9701	0.9682	0.9619	0.9588	0.9513	0.9465	0.9424	0.9584	0.9476	0.9551	0.9431	0.9223	0.8991
KNN-D	0.9313	0.9281	0.9183	0.8941	0.8744	0.8449	0.8009	0.9317	0.9131	0.9013	0.9125	0.8905	0.8821
GS-D	0.8339	0.8205	0.8123	0.7792	0.7347	0.7153	0.6824	0.8383	0.8176	0.8198	0.8208	0.7718	0.7949
BIC-C-D	0.9629	0.9607	0.9217	0.8712	0.6915	0.6149	0.5906	0.9359	0.9232	0.9277	0.8404	0.6514	0.7368

Table 1: Test set classification accuracy of victim classifiers as a function of attack strength on poisoned and sanitized TREC05 datasets.

The results are listed in Table 1 and 2. Table 1 shows the performance of victim classifiers as a function of attack strength on poisoned and sanitized TREC05 datasets. We first trained the target classifiers on

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

Table 2: TPRs and FPRs of three defenses on the TREC05 dataset under all attack cases.

# Poisoned Ham, # Poisoned Spam	0,0	0, 1000	0, 2000	0, 3000	0, 4000	0, 5000	0, 6000	1000, 1000	1000, 2000	2000, 1000	2000, 2000	2000, 4000	4000, 2000
# Components	(21,18)	(29,16)	(22,18)	(25,17)	(19,20)	(24,20)	(24,31)	(49,27)	(25,15)	(37,29)	(48,28)	(40,29)	(36,28)
# Revised Components	(1,5)	(0,6)	(6,11)	(5,10)	(1,16)	(2,9)	(7,11)	(19,18)	(11,7)	(17,12)	(9,7)	(14,11)	(14,13)
# Removed Components	(0,1)	(5,3)	(2,6)	(1,2)	(2,4)	(3,4)	(4,11)	(7,4)	(4,2)	(4,6)	(12,5)	(10,5)	(8,11)

Table 3: The number of components, number of revised components, and number of removed components of each class under all attack cases on the TREC05 dataset.

the clean dataset (Attack (0,0) in Table 1), yielding clean test accuracies (baselines) for SVM, LR, and LSTM of 0.9522, 0.9616, and 0.9632, respectively. The test accuracies of the classifiers poisoned by 12 DP attacks (described in Section 5.1) are shown as poisoned SVM/LR/LSTM in Table 1. As the total attack strength (the sum of the attacking ham and spam samples) is strengthened to 6000, the classification accuracies of SVM/LR drop below 0.75 and LSTM drops to 0.8. Thus, embedding real ham into the spam set and real spam into the ham set is indeed a significant poisoning attack on SVM/LR/LSTMs.

Then, we applied the four defenses on the corrupted training sets and retrained the victim classifiers on the sanitized datasets. The corresponding test accuracies are shown as BIC-D, KNN-D, GS-D, and BIC-C-D in Table 1. Since BIC-C-D [26] unrealistically assumes the defender knows which class is clean, we alternately apply BIC-C-D on ham and spam until the total BIC cost over the two classes converges. The order of sanitization was fixed – it was always initiated from class ham. As expected, the test accuracies of the classifier with BIC-C-D drop rapidly when the total attacking strength exceeds 4000. The test accuracies of classifiers using KNN/GS-based defenses also decline gradually as the attack strength increases, while our BIC-based defense performs well and stably. In all cases, our defense surpasses the other three defenses in classification accuracy (marked in bold). When the attack is strengthened to 5000 ham emails in the spam training set, the KNN-based defense exhibits little improvement in test accuracy over that of the poisoned classifier. The classifiers equipped with the GS-based defense perform even worse than the poisoned classifiers. However, our defense significantly improves test accuracies even under strong attacks, restoring the test accuracies close to the clean baselines. For LSTM with the GS-based defense, its accuracy is even worse than the poisoned LSTM in all cases. The performance of the GS-based defense is affected by the choice of its hyper-parameters – clip norm and noise multiplier. We directly used the hyper-parameter settings from [19], which are tuned for LR, not LSTM. See [2, 30] for tuning the hyperparameters of a DP-SGD optimizer. Note that for the embedded data poisoning scenario considered here, there is no clean validation set available for (principled) tuning of hyperparameters.

Table 2 shows the TPRs and FPRs of the BIC/KNN-based defenses (since the GS-based defense does

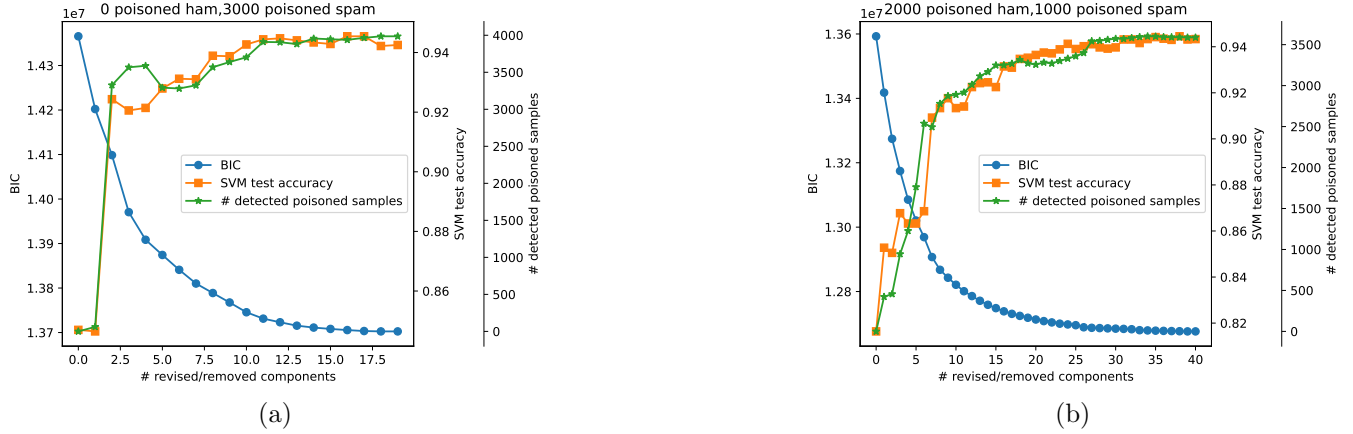


Figure 2: BIC cost, SVM test accuracy, and the number of detected poisoned samples as a function of the number of component change steps for attacks against TREC05 using (a) 0 poisoned ham samples, 3000 poisoned spam samples (b) 2000 poisoned ham samples, 1000 poisoned spam samples.

not identify the poisoned samples). Compared with KNN-D and BIC-C-D, our defense has relatively low FPRs for all cases. Besides, our defender has higher TPRs than the KNN-based defense when only the spam set is poisoned and has comparable TPRs when both of the classes are poisoned. When the attacking strength exceeds 4000, BIC-C-D fails to detect poisoned samples and falsely removes a large number of clean samples. In short, compared with the other three defenses, our defense does not require any additional (clean) validation set or elaborate hyperparameter tuning, and it is more robust, especially when just one of the classes is corrupted.

When there is no attack, our defense still sanitizes several components in both classes and falsely detects 789 clean training samples as poisoned samples. The average log likelihoods of these removed samples under ham and spam are -879.77 and -852.88 , respectively (in other words, the likelihoods are similar). Also, for an SVM trained on a “perfectly sanitized” dataset (*i.e.*, clean dataset, without these samples), the classification accuracy on the set of these removed samples is only 0.5855. These falsely detected samples are close to the decision boundary and are well-explained by both classes. Given the similar likelihoods, it is BIC-efficacious to remove/revise the components formed by these samples; thus our defense inevitably produces some false positives. Besides, removing these ambiguous samples in fact slightly increases the test accuracy.

In Figure 2, we show how the total BIC cost, SVM test accuracy, and the number of detected poisoned samples change with the number of components removed/revise by our defense under the attack with 3000 poisoned samples injected into TREC05. (Note that in practice, we only remove the detected poisoned samples once the detection procedure terminates. But to see the algorithm’s progress, via Figure 2, we trained an SVM on the training set without the detected samples at each detection step.) As we emphasized in Section 4, our method guarantees strict descent in the BIC objective. But we cannot guarantee that the test accuracy or the number of detected poisoned samples is non-decreasing. Samples that were previously deemed poisoned might be restored clean at subsequent detection steps (re-assigned to the class to which they are labeled) and vice versa, which explains the slight fluctuation in the curves of SVM test accuracy and the number of detected poisoned samples. Overall, though, the strong trend of the two curves is an increase in detection accuracy and true positive detections with increasing detection steps. Specifically, the two curves increase sharply in the early stages and converge in the final stages, as the BIC cost is further decreased.

We show the number of components, number of revised components and number of removed components for both ham and spam under all attack cases in Table 3. In general, the total number of removed and revised components increases as the attack is strengthened. For most of the cases, our method prefers revising a poisoned component rather than removing it. A possible reason is that, given a large number of features (30000 for TREC05), it is difficult to cluster the clean samples and the poisoned samples into

separate groups [43]. Most of the poisoned components are formed by both clean and poisoned samples, and it is apparently most BIC-efficacious to revise them.

6 Experiments on Multi-class Classification Tasks

6.1 Experiment Setup

Dataset and mixture model: For multi-class ($W > 2$) classification, we used the 20-Newsgroups dataset (20NG) [22], MNIST [11], CIFAR10 [20], and STL10 [7]. 20NG collects news documents across 20 different news groups. Each group corresponds to a different topic and contains around 600 training samples and 400 test samples. MNIST is a dataset of 28x28 gray-scale images. It contains 5000 training images and 1000 test images per class. CIFAR10 consists of 32x32 color images, with 5000 training images and 1000 test images per class. STL10 is composed of 96x96 color images, with 500 training images and 800 test images⁷. The experiments on each dataset involved 5 classes. For 20NG, we chose classes “rec.sport.baseball”, “soc.religion.christian”, “comp.graphics”, “rec.autos”, and “misc.forsale”. For MNIST, CIFAR10, and STL10, we chose the first 5 classes.

To simulate reasonable and potent embedded DP attacks, we used real samples as poisoning samples. For 20NG, we split the test set: 220 samples per class are used for testing and 160 samples per class are used for poisoning. For MNIST, we split the training set: 2000 images per class are used for training and 800 images per class are used for poisoning⁸. For CIFAR10, we split the training set: for each class, 4000 images are used for training and 800 images are preserved for poisoning. For STL10, we split the test set: for each class, 700 images are used for testing and 100 images are used for poisoning. For each dataset, the samples used for poisoning are injected into the training set, as described further below.

For each dataset, we trained a mixture model for each of its classes. We applied PMMs on 20NG. After pre-processing, the dictionary contains around 10000 unique words, and the training and test samples are represented using a bag-of-words. Each PMM component is a multinomial joint probability mass function. For MNIST, we applied Gaussian mixture models (GMMs). We first flattened the training images as 784-dimension vectors, normalized the intensity values to $[0, 1]$, and centered the feature vectors. Then we trained GMMs on the pre-processed feature vectors. To reduce the model complexity, we assumed the features are independent conditioned on the mixture component of origin (*i.e.* a diagonal covariance matrix for each Gaussian component). For CIFAR10 and STL10, as the raw images are not very suitable for clustering, we trained GMMs on the feature vectors extracted from an internal layer of the victim NN-based classifier. In the experiments, we used the 512-dimensional penultimate layer features, which were again assumed independent, conditioned on the mixture component of origin. The features were again normalized and centered before GMM learning.

DP attack and target classifiers: For the multi-class classification task, we launched 5 DP attacks on all datasets. In attack $i = 1, \dots, 5$, we used samples of the first i classes for poisoning. Samples from a given class c are used to poison all other classes. That is,

For attack $i = 1, \dots, 5$:

For class $c = 1, \dots, i$:

Evenly distribute the poisoning samples of class c to the training sets of classes $w \neq c$ and label these samples as w .

We chose linear SVM and LR as the target classifiers for 20NG and MNIST, as they are not effective for classifying complicated images such as CIFAR10 and STL10. For the NN based classifier, we chose a bi-directional one-layer LSTM recurrent neural network with 128 hidden units for 20NG, ResNet-18 [17] for MNIST and CIFAR10, and ResNet-34 for STL10⁹.

⁷The STL10 dataset is an image recognition dataset with 100000 unlabeled images for developing unsupervised feature learning, deep learning, self-taught learning algorithms. Here we only use the labeled set.

⁸For MNIST we only used half of the training samples.

⁹Since the number of labeled training samples of STL10 is not sufficient for training a complicated NN from scratch, we fine-tuned a pre-trained ResNet-34 on STL10.

Evaluation criteria: The performance of our defense (BIC-D) was assessed using the same metrics as in Section 5: 1) improvement in test classification accuracy, relative to that of the poisoned classifier, of the classifier trained following data sanitization; 2) TPR; and 3) FPR. We also applied the KNN-based defense (KNN-D) [38]¹⁰, GS-based defense (GS-D) [19], and the SVD-based defense (SVD-D) [12], using the same poisoned training sets. As [26] was only proposed for binary classification (and assumes the poisoned class is known), we did not apply this method on the multi-class classification task. We also did not apply the SVD-based defense on the LSTM classifier, since it is only applicable to linear classifiers. For ResNet models, we applied the SVD-based defense on the output layer, since it is actually a linear classifier built on features extracted based on the previous layers.

Hyper-parameter setting: 1) For the KNN-based defense, we set the number of neighbors at $K = 10$, which is suggested by [38]. 2) For the GS-based defense, we applied a DP-SGD optimizer with clip-norm of 2.0 and noise multiplier of 0.1 for training, which were suggested in [19]. 3) For the SVD-based defense, we set the number of detection steps at $\beta = 2$, which is the same as in [12]. To show the best performance for that method, we set ϵ to the real poisoning ratio if the training set is poisoned, and set it to 0.01 if there is no poisoning.

6.2 Experimental Results

The results are listed in Table 4-11. Table 4, 6, 8, and 10 show the test accuracy of victim classifiers as a function of attack strength on poisoned and sanitized 20NG, MNIST, CIFAR10, and STL10 datasets, respectively. We first trained the target classifiers on the attack-free datasets to get baseline test accuracies (*i.e.*, column 0 of poisoned classifiers). Then we trained the classifiers on training sets poisoned by the 5 DP attacks described in Section 6.1, with the resulting test accuracies in columns 1-5 of “Poisoned”, respectively. For 20NG and MNIST, as the number of classes used for poisoning is increased to 5, the classification accuracies of SVM and LR drop by over 30% (absolute percentage drop). The test accuracy of the NN-based classifier drops by nearly 30% on 20NG and 20% on MNIST. The test accuracies of ResNets drop by over 10% on CIFAR10 and nearly 10% on STL10. Thus, the attacks designed in Section 6.1 are indeed pretty effective poisoning attacks against all target classifiers, on all datasets.

Then we applied our defense and KNN/GS/SVD-based defenses on the poisoned training sets and retrained the target classifiers on the sanitized datasets. The corresponding test accuracies on 20NG, MNIST, CIFAR10, and STL10 are shown as BIC-D, KNN-D, GS-D, and SVD-D in Table 4, 6, 8, and 10, respectively. For 20NG, CIFAR10, and STL10, our defense outperforms the other three defenses in classification accuracy (marked in bold) in all attacking cases, excluding attack 0 (attack-free) against CIFAR10 and STL10. When there is no poisoning on CIFAR10 and STL10, SVD-D performs the best since we set ϵ as a small number. However, in practice, it will be difficult to find an appropriate value for ϵ in the absence of a clean validation set. For 20NG, although KNN/GS/SVD-based defenses improve the test accuracies for most cases, the test accuracies drop by 10%-25% as the attack strength increases, compared with the clean baseline. Our BIC-based defense performs well and stably – the test accuracy drops by 5% at most. For CIFAR10, the ResNet-18 with KNN/GS-based defense performs even worse than the poisoned classifier in all attacking cases. The SVD-based defense only improves the test accuracies by 4% at most. By contrast, the test accuracy of the classifier with our method drops by only 2% under the strongest attack, compared with the clean baseline. For STL10, the ResNet-34 with KNN-based defense performs worse than the poisoned classifier in all attacking cases. The GS-based defense mitigates the negative effect of DP, but the test accuracy still drops by 6% compared with the clean baseline. The SVD-based defense has little effect in improving test accuracy, compared with that of the poisoned classifier. On the other hand, the test accuracies of the classifier with our method drop by 1.31% at most, compared with the clean baseline.

For MNIST, our method still performs well and beats GS/SVD-based defenses, while the KNN-based defense with $K = 10$ (KNN-10-D in Table 6) gives slightly better results than ours. $K = 10$ was suggested

¹⁰Although the method was proposed for binary classification tasks, it is straightforward to apply it for multi-class classification tasks.

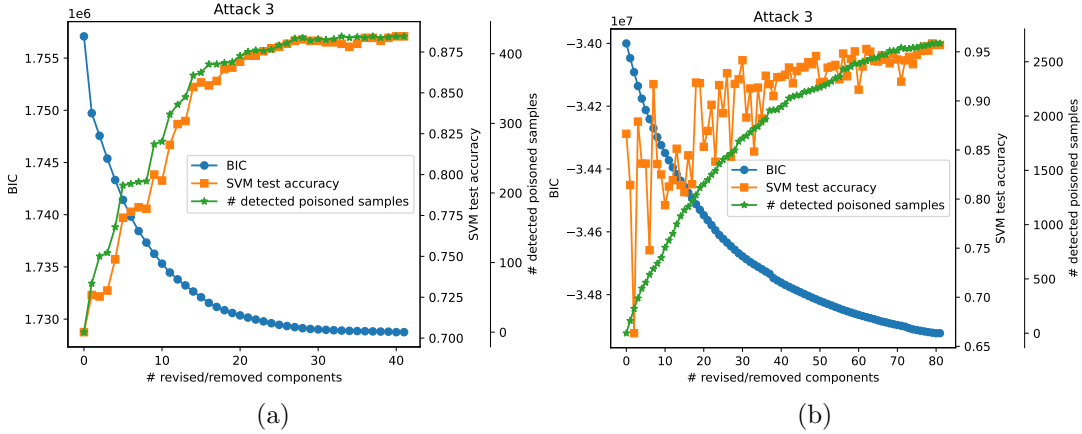


Figure 3: BIC cost, SVM test accuracy, and the number of detected poisoned samples as a function of the number of visited components (detection stages) under attack 3 against (a) 20NG dataset (b) MNIST dataset.

by [38] and was chosen according to the performance of the algorithm evaluated on a (clean) validation dataset of MNIST. However, for the embedded DP attack scenario considered here, the defender does not have access to a trusted validation set. Also, as we discussed in Section 2, the choice of K tremendously impacts the detector’s performance. Thus, we also evaluated the performance of the KNN-based detector with $K = 3$, with the resulting test accuracies shown as KNN-3-D in Table 6. As we can see, the test accuracy drops significantly under attack 5, compared with KNN-10-D and our method. On the attack-free training set, the SVD-based defense performs best on SVM and LR, and the KNN-based defense performs best on ResNet-18, since the hyper-parameters of both methods are well-chosen.

Table 5, 7, 9, and 11 show the TPRs and FPRs of the BIC/KNN/SVD-based defenses (since the GS-based defense does not identify the poisoned samples) on 20NG, MNIST, CIFAR10, and STL10 datasets, respectively. Since the performance of the SVD-based defense depends on the classifier architecture and training loss function (it evaluates the gradients of same), we respectively show its TPR/FPR on SVM, LR, and ResNet as SVD-D-S, SVD-D-L, and SVD-D-R. We reiterate here that both KNN and SVD-based defenses evaluated here are supervised methods, *i.e.* with appropriately chosen hyper-parameters. For 20NG, CIFAR10, and STL10, compared with the other two defenses, our defense has relatively high TPRs and low FPRs for all cases. Almost no clean samples are falsely reported by our defense, while a large number of poisoned samples are correctly identified. The KNN-based defense falsely detects lots of clean samples in all attack cases, even when there is no poisoning. By contrast, the SVD-based defense only detects a small amount of poisoned samples, especially when the attack strength is weak. For MNIST, the SVD-based detector has lower TPRs and FPRs than ours on SVM and LR, and does not perform well on ResNet. The KNN-based detector with $K = 10$ has higher TPRs and lower FPRs than our method under most attacking cases. Again, the performance of the KNN-based detector is affected by the choice of K , and $K = 10$ was chosen based on the detector’s performance evaluated on a clean validation set for MNIST. With $K = 3$, the detector is less “aggressive” – it reports fewer detected poisoned images and has much lower TPRs.

Similar to the results in Section 5, our method also falsely removes a few clean samples from the attack-free datasets (attack 0). These samples are well-explained by more than one class, and it is BIC-efficacious to re-distribute these samples. Removing these samples slightly *increases* the test accuracy, compared with the clean classifier baselines. In summary, our defense is an unsupervised method and significantly improves test accuracies for all classifiers on all datasets even under strong attacks, restoring the test accuracies close to the clean baselines.

In Figure 3a and 3b, we respectively show how the total BIC cost, SVM test accuracy, and the number of detected poisoned samples change with the number of components removed/revised by our defense method under attack 3 against 20NG and MNIST. (Note that in practice, we only remove the

Attack	0	1	2	3	4	5
SVM						
Poisoned	0.9172	0.8681	0.7936	0.7036	0.6072	0.5281
BIC-D	0.9254	0.9127	0.9027	0.8845	0.8818	0.8736
KNN-D	0.9009	0.8891	0.8672	0.8236	0.7736	0.7391
GS-D	0.9127	0.8918	0.8691	0.8391	0.8073	0.7645
SVD-D	0.9181	0.8663	0.8272	0.7745	0.7254	0.6600
LR						
Poisoned	0.9309	0.8781	0.8291	0.7536	0.6718	0.5909
BIC-D	0.9354	0.9218	0.9172	0.8954	0.8872	0.8809
KNN-D	0.8909	0.8791	0.8681	0.8309	0.7963	0.7700
GS-D	0.9181	0.8873	0.8855	0.8536	0.8373	0.7973
SVD-D	0.9309	0.9081	0.8690	0.8618	0.8463	0.8400
LSTM						
Poisoned	0.8063	0.7427	0.7163	0.6736	0.6055	0.5336
BIC-D	0.8073	0.8064	0.8018	0.7800	0.7627	0.7481
KNN-D	0.7454	0.7409	0.7200	0.7136	0.7000	0.6664
GS-D	0.2636	0.2555	0.2400	0.2282	0.2545	0.2536

Table 4: Test set classification accuracy of victim classifiers on poisoned and sanitized 20NG datasets, under different attacks.

Attack	0	1	2	3	4	5
True Positive Rates (TPRs)						
BIC-D	-	0.8325	0.8203	0.7958	0.7843	0.7762
KNN-D	-	0.7687	0.7412	0.7296	0.7281	0.7104
SVD-D-S	-	0.2500	0.4156	0.4708	0.4828	0.4900
SVD-D-L	-	0.5937	0.6218	0.7020	0.7562	0.7650
False Positive Rates (FPRs)						
BIC-D	0.0084	0.0145	0.0135	0.0145	0.0168	0.0155
KNN-D	0.1993	0.2001	0.1991	0.1997	0.2017	0.2091
SVD-D-S	0.01	0.0405	0.0632	0.0858	0.1118	0.1378
SVD-D-L	0.01	0.0219	0.0408	0.0483	0.0527	0.0581

Table 5: TPRs and FPRs of two defenses on the 20NG dataset under all attack cases.

detected poisoned samples after the detection procedure terminates. Solely for visualization purposes, we trained an SVM on the training set without the detected samples at each detection step.) As we emphasized in Section 4 and similar to the results in Section 5, our method strictly descends in the BIC objective. But we cannot guarantee that the test accuracy or the number of detected poisoned samples is non-decreasing. For 20NG, both SVM test accuracy and the number of detected poisoned samples are almost strictly non-decreasing, with rapid increases during the first few detection steps. For MNIST, the SVM test accuracy fluctuates heavily as the number of visited components increases and finally converges at around 0.95, while the number of detected poisoned samples is nearly monotonically increasing.

We show the number of components, number of revised components, and number of removed components of each class under all attack cases on all datasets in Table 12. For all datasets, the total number

Attack	0	1	2	3	4	5
SVM						
Poisoned	0.9621	0.8791	0.8717	0.8665	0.7773	0.5711
BIC-D	0.9536	0.9519	0.9551	0.9569	0.9537	0.9441
KNN-10-D	0.9560	0.95830	0.95640	0.95910	0.96180	0.9544
KNN-3-D	0.9616	0.9416	0.9367	0.9019	0.8772	0.8464
GS-D	0.9508	0.8846	0.8007	0.7867	0.7048	0.6213
SVD-D	0.96240	0.9537	0.9497	0.9415	0.9377	0.9239
LR						
Poisoned	0.9606	0.8927	0.8521	0.8005	0.6592	0.6390
BIC-D	0.9628	0.9569	0.9508	0.9583	0.9552	0.9455
KNN-10-D	0.9636	0.95840	0.95600	0.96110	0.95650	0.9534
KNN-3-D	0.9604	0.9450	0.9359	0.8927	0.8814	0.8484
GS-D	0.9545	0.9241	0.8004	0.7186	0.6079	0.5754
SVD-D	0.96590	0.9536	0.9452	0.9377	0.9353	0.9392
ResNet-18						
Poisoned	0.9976	0.9548	0.8986	0.8735	0.8597	0.8266
BIC-D	0.9986	0.9918	0.9951	0.9908	0.9911	0.9869
KNN-10-D	0.99880	0.99880	0.99760	0.99640	0.99610	0.9953
KNN-3-D	0.9974	0.9935	0.9706	0.9688	0.9622	0.9568
GS-D	0.9968	0.9787	0.9311	0.8846	0.8246	0.8001
SVD-D	0.9986	0.9920	0.9644	0.9322	0.9073	0.8433

Table 6: Test set classification accuracy of victim classifiers on poisoned and sanitized MNIST datasets, under different attacks.

of removed and revised components is nearly strictly increasing as the attack is strengthened. For CIFAR10 and STL10, we applied our defense method on the features extracted from the penultimate layer of the NN-based classifier. The feature vectors of poisoned samples are well separated from those of clean samples; thus our method prefers removing poisoned components rather than revising them. By contrast, our detector prefers revising poisoned components rather than removing them for 20NG. Similar to the results for TREC05 in Section 5, given a high feature dimensionality, most of the poisoned components are formed by both clean and poisoned samples, and it is apparently BIC-efficacious to revise them.

7 Computational Complexity of the BIC-based Defense

For attack 1 poisoning the MNIST dataset, we report the computation time required to implement different defenses and that required to train the DNN classifier (deep learning). All the experiments are conducted on a compute platform consisting of Intel i9-10900K CPU, NVIDIA GeForce 3080 GPU and 32GB memory. Table 13 shows the time used for training a ResNet-18 DNN classifier and deploying BIC/KNN/GS/SVD-based defenses. We train the ResNet-18 for 50 epochs. The initial learning rate of 0.001 is divided by 10 every 20 epochs. For the BIC/KNN/SVD-based methods, the computational complexity is the time spent on anomaly detection and removal. Since the GS-based defense mitigates DP attacks during DNN training, its reported computational time is the training time. As we can see,

Attack	0	1	2	3	4	5
True Positive Rates (TPRs)						
BIC-D	-	0.9562	0.9556	0.9429	0.9568	0.9315
KNN-10-D	-	0.9950	0.9918	0.9867	0.9834	0.9827
KNN-3-D	-	0.8662	0.8106	0.7833	0.7543	0.7397
SVD-D-S	-	0.8600	0.8418	0.8591	0.8821	0.8832
SVD-D-L	-	0.8812	0.8668	0.8875	0.9009	0.9082
SVD-D-R	-	0.7725	0.6550	0.5287	0.4581	0.4352
False Positive Rates (FPRs)						
BIC-D	0.0465	0.0723	0.0503	0.0406	0.0561	0.0531
KNN-10-D	0.0182	0.0175	0.0166	0.0169	0.0186	0.0194
KNN-3-D	0.0088	0.0109	0.0191	0.0394	0.0533	0.0695
SVD-D-S	0.0100	0.0112	0.0253	0.0338	0.0377	0.0467
SVD-D-L	0.0100	0.0095	0.0213	0.0270	0.0317	0.0367
SVD-D-R	0.0100	0.0182	0.0552	0.1131	0.1734	0.2259

Table 7: TPRs and FPRs of two defenses on MNIST under all attack cases.

Attack	0	1	2	3	4	5
Poisoned	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

Table 8: Test set classification accuracy of ResNet-18 on poisoned and sanitized CIFAR10 datasets, under different attacks.

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 9: TPRs and FPRs of two defenses on CIFAR10 under all attack cases.

Attack	0	1	2	3	4	5
Poisoned	0.9028	0.8902	0.8722	0.8548	0.8328	0.8165
BIC-D	0.9028	0.90800.90850.90200.90680.8897				
KNN-D	0.8000	0.7825	0.7440	0.7391	0.7217	0.7065
GS-D	0.9008	0.8972	0.8882	0.8631	0.8508	0.8405
SVD-D	0.90680.8954	0.8714	0.8622	0.8282	0.8111	

Table 10: Test set classification accuracy of ResNet-34 on poisoned and sanitized STL10 datasets, under different attacks.

Attack	0	1	2	3	4	5
True Positive Rates (TPRs)						
BIC-D	-	0.9100	0.88500.87000.85750.8040			
KNN-D	-	0.97000.8500	0.8133	0.8100	0.8040	
SVD-D	-	0.3800	0.2800	0.3066	0.3050	0.2220
False Positive Rates (FPRs)						
BIC-D	0	0.00120.00080.00080.00120.0028				
KNN-D	0.5028	0.4964	0.4936	0.4976	0.4944	0.4856
SVD-D	0.0100	0.0248	0.0576	0.0832	0.1112	0.1556

Table 11: TPRs and FPRs of two defenses on STL10 dataset under all attack cases.

Attack	0	1	2	3	4
20NG					
# components	(8,14,9,12,8)	(12,14,11,15,12)	(12,12,14,16,11)	(20,12,12,16,12)	(20,13,16,16,11)
# revised components	(1,1,4,0,3)	(2,5,7,6,8)	(6,6,10,5,6)	(7,5,7,7,7)	(9,6,13,5,9)
# removed components	(0,0,0,0,0)	(0,1,1,1,1)	(0,1,1,3,0)	(0,3,1,3,1)	(1,2,1,2,2)
MINIST					
# components	(28,28,27,27,28)	(28,31,30,24,33)	(44,31,39,37,43)	(38,41,39,38,44)	(31,33,33,38,47)
# revised components	(6,4,15,10,5)	(4,2,12,15,7)	(3,4,14,14,5)	(3,7,7,3,4)	(6,4,12,11,4)
# removed components	(0,0,2,1,0)	(0,7,5,3,4)	(5,8,10,8,11)	(6,17,9,11,13)	(8,16,9,12,18)
CIFAR10					
# components	(13,14,12,15,11)	(14,15,21,17,20)	(15,13,19,19,23)	(19,13,18,17,19)	(21,16,17,14,22)
# revised components	(1,2,4,3,1)	(0,0,1,0,1)	(0,0,0,1,1)	(0,0,0,0,1)	(0,0,0,0,0)
# removed components	(0,0,1,1,0)	(3,2,3,2,2)	(5,2,5,4,4)	(6,3,8,5,4)	(7,5,6,6,5)
STL10					
# components	(8,10,11,8,10)	(9,9,16,9,12)	(10,8,16,12,12)	(15,13,17,17,11)	(15,15,16,16,11)
# revised components	(0,0,0,0,0)	(0,1,0,0,2)	(1,0,1,1,0)	(1,0,0,1,4)	(2,1,1,1,0)
# removed components	(0,0,0,0,0)	(0,1,2,2,1)	(2,1,4,3,3)	(3,3,3,5,3)	(4,5,4,6,4)

Table 12: The number of components, number of revised components, and number of removed components of each class under all attack cases on 20NG, MINIST, CIFAR10, and STL10 datasets.

KNN-based defense is extremely fast – it detects and removes anomalies in about 2 seconds. Our BIC-based defense only sanitizes one component in one class at each detection step, and thus takes longer than the KNN-based defense. However, the execution time of our BIC-based defense is comparable to the DNN training time. As expected, SVD-based defense is computationally expensive as it performs SVD and re-trains the DNN at each detection step. GS-based defense greatly increases the training time due to the DP-SGD optimizer applied during DNN training.

ResNet-18 training	BIC-D	KNN-D	GS-D	SVD-D-R
497.67	427.32	2.08	1808.68	1122.42

Table 13: Time (in seconds) used for training a ResNet-18 and deploying BIC/KNN/GS/SVD-based defenses on the MNIST dataset poisoned by attack 1.

8 Conclusion

We proposed an *unsupervised* BIC-based mixture model defense against DP attacks on classifiers, where poisoned samples are an unknown subset (potentially) simultaneously embedded in the training sets of multiple classes. Our defense applies mixture modeling to accurately explain the poisoned dataset and concentrate poisoned samples into several mixture components. Also, it jointly identifies poisoned components and poisoned samples within them by minimizing the BIC cost, with the identified poisoned samples purged from the training set prior to classifier training. We launched our defense and three other defenses against DP attacks targeting SVM, LR, and NN-based classifiers for the TREC05, 20NG, MNIST, CIFAR10, and STL10 domains. Experiments demonstrate the effectiveness and robustness of our defense under strong attacks, as well as the superiority over the other defenses.

Acknowledgements

This research was supported in part by grants from AFOSR and ONR and by a gift from Cisco.

References

- [1] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *Proc. AAAI*, 2016.
- [2] Eugene Bagdasaryan, Omid Poursaeed, and Vitaly Shmatikov. Differential privacy has disparate impact on model accuracy. In *Proc. Neural Information Processing Systems*, 2019.
- [3] Battista Biggio, Iginio Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Proc. European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2013.
- [4] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proc. ACM CCS*, 2018.
- [5] Chengliang Chai, Lei Cao, Guoliang Li, Jian Li, Yuyu Luo, and Samuel Madden. Human-in-the-loop outlier detection. In *Proc. Int’l Conference on Management of Data*, 2020.
- [6] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR*, abs/1712.05526, 2017.
- [7] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR 15*, 2011.
- [8] Gordon V. Cormack and Thomas R. Lynam. Trec 2005 spam public corpora. <https://plg.uwaterloo.ca/~gvcormack/trecspamtrack05>, 2005.
- [9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

- [11] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [12] Ilias Diakonikolas, Gautam Kamath, Daniel Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. Sever: A robust meta-algorithm for stochastic optimization. In *Proc. ICML*, 2019.
- [13] RO Duda, PE Hart, and DG Stork. *Pattern Classification, Second Edition*. Wiley, 1999.
- [14] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. Robust logistic regression and classification. In *Proc. Neural Information Processing Systems*, 2014.
- [15] Michael W. Graham and David J. Miller. Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. *IEEE Trans. Signal Process.*, 54(4):1289–1303, 2006.
- [16] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [19] Sanghyun Hong, Varun Chandrasekaran, Yigitcan Kaya, Tudor Dumitras, and Nicolas Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *CoRR*, abs/2002.11497, 2020.
- [20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>, 2009.
- [21] Ricky Laishram and Vir Virander Phoha. Curie: A method for protecting SVM classifier from poisoning attack. *CoRR*, abs/1606.01584, 2016.
- [22] Ken Lang. Newsweeper: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.
- [23] Aaron D. Lanterman. Schwarz, Wallace, and Rissanen: Intertwining Themes in Theories of Model Selection. *International Statistical Review*, 69(2):185–212, 2001.
- [24] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Proc. Neural Information Processing Systems*, pages 1885–1893, Dec. 2016.
- [25] Jintang Li, Tao Xie, Chen Liang, Fenfang Xie, Xiangnan He, and Zibin Zheng. Adversarial attack on large scale graph. *IEEE Transactions on Knowledge and Data Engineering*, 2021.
- [26] Xi Li, David J. Miller, Zhen Xiang, and George Kesidis. A scalable mixture model based defense against data poisoning attacks on classifiers. In *Proc. Third Int’l Conference on Dynamic Data Driven Applications Systems (DDDAS)*, 2020.
- [27] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2018.

- [28] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *Proceedings IJCAI*, Aug. 2019.
- [29] G McLachlan and D Peel. *Finite mixture models*. Wiley, 2004.
- [30] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *Proc. ICLR*, 2018.
- [31] David J. Miller, Zhen Xiang, and George Kesidis. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 108(3):402–433, 2020.
- [32] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proc. ACM AISec*, 2017.
- [33] John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.
- [34] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. Misleading learners: Co-opting your spam filter. In *Proc. Machine Learning in Cyber Trust: Security, Privacy, and Reliability*, 2009.
- [35] Seong Joon Oh, Max Augustin, Mario Fritz, and Bernt Schiele. Towards reverse-engineering black-box neural networks. In *Proc. ICLR*, 2018.
- [36] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proc. ACM AsiaCCS*, 2017.
- [37] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proc. IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.
- [38] Andrea Paudice, Luis Muñoz-González, and Emil C. Lupu. Label sanitization against label flipping poisoning attacks. In *Proc. ECML PKDD Workshops*, 2018.
- [39] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *Proc. ICLR*, 2018.
- [40] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5), September 1978.
- [41] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464, 1978.
- [42] Shaoxu Song, Fei Gao, Ruihong Huang, and Yihan Wang. On saving outliers for better clustering over noisy data. In *Proc. International Conference on Management of Data*, June 2021.
- [43] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*, pages 273–309, 2004.
- [44] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 3517–3529, 2017.
- [45] Wentai Wu, Ligang He, Weiwei Lin, Yi Su, Yuhua Cui, Carsten Maple, and Stephen A. Jarvis. Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

- [46] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.
- [47] Yugen Yi, Wei Zhou, Yanjiao Shi, and Jiangyan Dai. Speedup two-class supervised outlier detection. *IEEE Access*, 6:63923–63933, 2018.
- [48] Yuxin Zhang, Yiqiang Chen, Jindong Wang, and Zhiwen Pan. Unsupervised deep anomaly detection for multi-sensor time-series signals. *IEEE Transactions on Knowledge and Data Engineering*, 2021.