Chapter 1
Using Deep Learning to Generate Relational HoneyData

Nazmiye Ceren Abay, Cuneyt Gurcan Akcora, Yan Zhou, Murat Kantarcioglu, and Bhavani Thuraisingham

Abstract Although there has been a plethora of work in generating deceptive applications, generating deceptive data that can easily fool attackers received very little attention. In this book chapter, we discuss our secure deceptive data generation framework that makes it hard for an attacker to distinguish between the real versus deceptive data. Especially, we discuss how to generate such deceptive data using deep learning and differential privacy techniques. In addition, we discuss our formal evaluation framework.

Keywords Cyber deception · Differential privacy · Deep learning · Decoy deployment

1.1 Introduction

Deception techniques have been recently deployed in cybersecurity for achieving many important goals ranging from understanding the attacker intent to tricking attackers into spending resources and time on fake targets. For example, honeypots (e.g., [29]) have been proposed to provide deceptive targets (i.e., servers) for attackers. Similarly, Honeyfiles (e.g., [31]) have been proposed to lure attackers to spend time in searching files and potentially disclose their intent. Still, to our knowledge, none of the previous work tries to create deceptive "data" (i.e., HoneyData) to fool potential attackers. Unfortunately, lack of realistic deceptive data may make it easier for an attacker to detect deception. For example, without good HoneyData, it may be easier to spot a fake database hosted on a honeypot.

The original version of this chapter was revised: Chapter authors have been added. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-02110-8_12

N. C. Abay · C. G. Akcora · Y. Zhou · M. Kantarcioglu (✉) · B. Thuraisingham
The University of Texas at Dallas, Richardson, TX, USA
e-mail: muratk@utdallas.edu

E. Al-Shaer et al. (eds.), *Autonomous Cyber Deception*,
https://doi.org/10.1007/978-3-030-02110-8_1

Recent work looked into generating privacy-preserving synthetic relational data using differential privacy (e.g., [2]). The main purpose of this line of work is to preserve individual privacy while providing data utility. Therefore, it is not clear whether they could be applicable for generating good HoneyData. In the context of cyber deception, it is important that the HoneyData is indistinguishable from real data so that it can easily fool the attacker.

Creating deceptive data (i.e., HoneyData) has many challenges. For different settings, we may need different types of HoneyData. For example, to deceive an attacker and feed false information, deceptive technical plans (e.g., technical drawings of an airplane) could be generated. On the other hand, to make HoneyFiles more believable, fake text data could be added to such files. Since addressing all these different types of data requires different techniques, in this work, we focus on generating deceptive HoneyData that is relational data. The main differentiating factor for relational data is that the number of columns and the types of the columns in a given dataset are known in advance. Still, generating realistic relational HoneyData while not disclosing sensitive information is a significant challenge.

We need to answer questions, such as: (1) how to automatically generate relational HoneyData? and (2) how to measure whether the generated relational HoneyData is deceptive enough? In this work, we try to answer these questions by leveraging existing work in differentially private synthetic data generation and explore its effectiveness for generating relational HoneyData.

As a part of this work, we propose an important measure for understanding the effectiveness of HoneyData. Basically, given the available information, a potential attacker may not build an effective machine learning model to distinguish between real vs HoneyData. We evaluate the effectiveness of relational HoneyData on real datasets, and show under what conditions differentially private deep learning techniques could be used to generate relational HoneyData.

Remainder of this book chapter is organized as follows: Sect. 1.2 details related work. Section 1.3 gives preliminaries for our data generation technique in Deep learning and privacy preserving, and Sect. 1.4 explains our methodology. We report our experimental results in Sect. 1.5 and conclude with Sect. 1.6.

1.2 Related Work

Cyber deception mechanisms have been heavily studied to enhance the computer security. However, most of the existing techniques are not focused on generating deceptive data. Here, we review the existing cyber deception techniques with their limitations and strengths.

Honeypots are a prominent cyber deception mechanism to investigate and analyze the unauthorized intrusions [29]. Honeypots are designed as trap-based isolated systems that appear vulnerable to attackers. Legitimate users are not supposed to interact with them and any interaction with honeypots is considered an illicit attempt. While interacting with intruders, honeypots gather information of

them to disclose intruders' behavior for forensic analysis. Although honeypots are a notable cyber deception technique, they have limitations. Since honeypots are *fake environments*, they might fail to simulate the real services. As attackers become more sophisticated, they ensure their safety by using more advanced systems to distinguish "fake" and real system to avoid honeypots [18]. Moreover, honeypots might create irredeemable risks for the real user environment when the attacker can use honeypots as a bridge to the real user environment [4].

In addition to Honeypots, decoy injection mechanisms evolved to integrate real systems in aiding defensive computer deception. These mechanisms serve as a *decoy* to intruders to mitigate unauthorized threats by distracting attackers from a target that has sensitive information.

Yuill et al. [31] present an intrusion detection system that installs *decoy files* on file servers with enticing names to capture the attention of attackers. These decoy files are constantly monitored and when accessed by any intruder, the system will trigger an alarm to notify system administrator. However, in some cases, decoy files fail to influence the perception of attackers since published data (e.g., password file stolen from LinkedIn[1]) provide attackers insight to distinguish between real and fabricated data. They can enhance their technique and re-attack again. To circumvent attacker insight, Juels et al. [19] propose *Honeywords* to defend hashed password databases by generating "fake passwords" that seem real to attackers. In their work, they preserve N-1 "fake passwords" referred as honeywords for each legitimate user password in the database. If any of the honeywords is submitted for logging into databases, attack has been detected and system administrator is notified that database has been hacked. Although honeywords are useful to detect the unauthorized intruders, in some cases it may deteriorate system performance because each submitted password is compared with all previously generated honeywords which slows down the authentication process for legitimate users. Also, generating and preserving the honeywords increases the storage requirement N times. Still, this approach is only applicable for password setting.

Our approach proposes decoy data generation to fool attackers without degrading system performance. Although decoy files are used in a cyber defensive system to entice attackers, they may reveal sensitive information if care is not taken during data generation. To preserve individual privacy, the decoy files require sanitization of sensitive information. Dwork [10] proposes a data privacy model as ε-*differential privacy* to ensure the protection of private data from leakage by perturbing the data with random noise based on ε. Differential privacy has been implemented in a number of data analysis tasks, including regression models [9, 33], classification models [26, 30], and privacy-preserving data publishing [3, 6, 32]. In some cases, it is required to combine differentially private algorithms to formulate complex privacy solutions. To track the total privacy loss while executing these repetitive mechanisms, Abadi et al. [1] propose the advanced composition theorem known as

[1] https://www.cnet.com/news/linkedin-confirms-passwords-were-compromised/.

the *moment accountant* and verify that it has the best overall privacy bound in the literature. In this work, we also employ the moment accountant to bound privacy of the proposed technique to generate decoy files.

To balance both utility and user privacy, Rubin [25] introduces repetitive perturbation of the original data as a substitute to the original data. However, data generation may suffer from curse of dimensionality when the data has more than dozen attributes. To overcome the curse of dimensionality, Zhang et al. [32] present PrivBayes as a private generative model that decomposes high-dimensional data into low-dimensional marginals by constructing a Bayesian network. Afterwards, noise is injected into previously constructed low-dimensional marginals to ensure differential privacy and the synthetic data is inferred from these sanitized marginals. Acs et al. [3] model another generative approach to produce synthetic samples. First, the original data is partitioned into k clusters with private kernel k-means. Then, each previously clustered data is inputted to private generative neural networks to create synthetic data.

Park et al. [23] propose DpEm as a private version of the iterative expectation maximization algorithm. They combine differential privacy and expectation maximization algorithm to cluster datasets. Here, we use this approach to discover patterns in latent space. We observed an improvement in the performance of this technique when used with partitioning the original dataset into unique data label groups. Here, we use this modified version in our experiments [23] as DpEm^{+} and compare its results in the experiments section.

Similar to the clustering approach, Abay et al. [2] propose a new generative deep learning method that produces synthetic data from a dataset while preserving the utility of the original dataset. In [2], the original data is partitioned into groups, and then the private auto-encoder (a type of deep learning model) is employed for each group. Auto-encoder learns the latent structure of each group and uses expectation maximization algorithm to simulate them. In this work, we employ the same data generation model but we explore whether these techniques are applicable in the context of generating relational honeydata.

1.3 Background

This section provides a summary of deep learning and the principles of the differential privacy. Deep learning is utilized for the proposed approach. Differential privacy is applied to deep learning model to construct the private generative model to prevent the disclosure of sensitive data while generating honeydata.

1.3.1 Deep Learning

Deep learning is a representation learning-based machine learning technique that has been applied to image recognition and natural language processing where they have resulted in remarkable advances. The power of deep learning is based on

learning hierarchical concepts that allows the model to build complex concepts from the simpler ones [16]. Deep learning can be employed for addressing supervised, semi-supervised, or unsupervised tasks. Here, we employ unsupervised deep learning to form a generative neural network that samples honeydata.

Most deep learning models create complex networks that are formed with multilayer architectures. This multilayer network is a parametrized function that aims to fit any given input. To get optimal parameters to generalize input structure, our aim is to minimize the mismatching of error in the input, defined as loss function $\mathscr{L}(\theta)$, where θ is the set of network parameters. For each step in the optimization process, θ is updated with its gradient as follows:

$$\theta_{t+1} = \theta_t - \alpha \left(\frac{1}{|n|} \sum_{x_i \in D} \nabla_\theta \, \mathscr{L}(\theta; x_i) \right), \tag{1.1}$$

where D is the dataset with n records $x_i \in \mathbb{R}^d$. Deep learning models have complex networks usually formed with multilayer architectures that hinder the optimization. To circumvent this obstacle, Stochastic gradient descent (SGD) is used in the optimization [28].

1.3.2 Differential Privacy

Differential privacy is a mathematical formula that ensures privacy even if adversary has background knowledge [10]. Differential privacy adds random noise to the aggregated statistics to hinder impersonation attacks.

Theorem 1.1 *Mechanism $\mathscr{M}$ is a randomized real-valued function that satisfies (ε, δ) -differential privacy for some $\varepsilon > 0$ and $\delta > 0$ if for any adjacent datasets d, d', and for any subset of the output $S \subseteq Range(\mathscr{M})$ it holds that*

$$Pr[\mathscr{M}(d) \in S] \leq \exp(\varepsilon) \, Pr[\mathscr{M}(d') \in S] + \delta. \tag{1.2}$$

Adjacent datasets d, d' differ only in one tuple while rest is the same.

Mechanism $\mathscr{M}$ achieves (ε, δ) -*differential privacy* by perturbing the deterministic real-valued function f with random noise defined as follows:

$$\mathscr{M}(d) = f(d) + z, \tag{1.3}$$

where z is generated randomly from zero-mean Gaussian mechanism. Here, standard deviation of Gaussian mechanism is calibrated with σ and f's sensitivity s_f defined by the maximum of the absolute distance $\|f(d) - f(d')\|$ where d and d' are adjacent datasets. Relation among (ε, δ), σ, and s_f in Gaussian mechanism is given as $\sigma^2 \varepsilon^2 \geqslant 2 \ln 1.25/\delta s_f^2$ [13].

1.3.3 Differentially Private Composition Theorem

To analyze the privacy budget of our proposed work, we employ both sequential composition [11, 12] and advanced composition theorems [1, 8].

In our proposed work, while training the auto-encoder, we track the privacy loss at the end of each batch iteration. In the optimization phase, value of the current privacy loss ε' that has been spent on the private auto-encoder in a given iteration $t \in T$ is computed. Training ends when ε' reaches the final privacy budget ε.

According to moments accountant [1], deep learning network is (ε, δ)-differentially private if the privacy loss for any $\varepsilon' < k_1(|B|/n)^2 T$ is such that for some constants k_1, k_2:

$$\varepsilon' \geq k_2 \frac{|B|/n\sqrt{T \log 1/\delta}}{\sigma},$$

where T is the number of training steps and $|B|$ is the number of samples in mini-batch with a given privacy budget ε, delta δ, and standard deviation σ of the zero-mean Gaussian distribution.

1.4 Methodology

This section describes the details of our Differentially Private Synthetic Data Generation Model (DPSYN). We introduce the main algorithm and components of DPSYN.

1.4.1 Differentially Private Synthetic Data Generation Model

DPSYN has the primary purpose of generating synthetic data that is indistinguishable from the real data from the attacker's perspective given background knowledge. DPSYN also preserves the privacy by bounding the privacy loss with differential privacy. Abadi et al. [1] apply the moment accountant on differentially private deep learning. Here, we make several modifications to this work and extend it as a data generative model.

Figure 1.1 shows the fundamental steps of DPSYN. The dataset D contains a sequence of n training examples $(x_1, y_1), \ldots, (x_m, y_m)$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Our learning approach partitions the dataset D into k groups denoted as $\{D_1, \ldots, D_k\}$. Partitioning of training examples is employed based on label $y \in \mathbb{R}$ associated with training example $x \in \mathbb{R}^d$. Group number k is identified by the unique label number. After partitioning the dataset into k groups $\{D_1, \ldots, D_k\}$, for each group private generative auto-encoder is constructed to generate synthetic data.

Algorithm 1 demonstrates the details of the proposed approach. The dataset that has sensitive information D is partitioned into k groups (Line 1) and those previously partitioned groups are used to construct the private generative auto-encoder (Line 4). This process is detailed later in Algorithm 2. Next, we obtain the private latent representation of the group (Line 5) with activation function $\mathbb{F}$ and inject it into a differentially private expectation maximization (DPEM) function. The DPEM function is detailed in [23]. The main task of DPEM is to detect different latent patterns in the encoded data and to generate output data with similar patterns. These patterns are decoded in Line 7, and appended to the synthetic data D'' (Line 8).

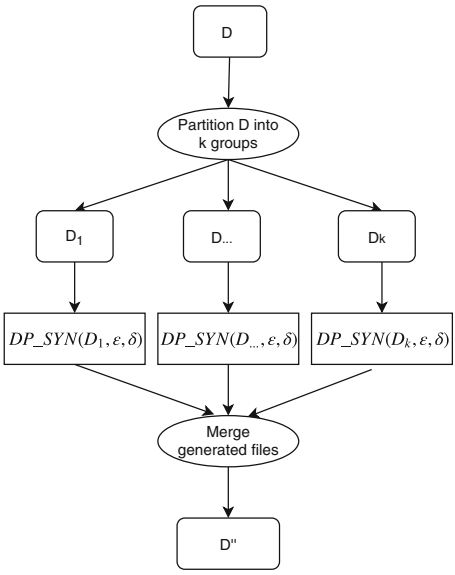

Fig. 1.1 Differentially private Synthetic data generation, DPSYN

Algorithm 1 DPSYN: Differentially Private Synthetic Data Generation

Require: D: $\{x_i, y_i\}_{i=1}^{m}$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$, α: learning rate; T: iteration number; ε: privacy budget; δ: Gaussian delta; σ: standard deviation; C: clipping constant.

$\{D_1 \dots D_k\} \leftarrow$ partition D into k groups
$D'' \leftarrow \{\}$
for $i \leftarrow 1$ **to** k **do**
  $\theta \leftarrow$ DP-Auto $(D_i, \alpha, \mathrm{T}, \varepsilon/2, \delta/2, \sigma, \mathrm{C})$ // see Algorithm 2
  $E' \leftarrow \mathscr{F}\,(X_i \cdot \theta)$ where $X_i \in D_i$
  $E'' \leftarrow$ DPEM $\left(E', \varepsilon/2, \delta/2\right)$ // see DPEM [23]
  ${D_i}' \leftarrow \mathscr{F}\left(E'' \cdot \theta^{\mathsf{T}}\right)$
  $D'' \leftarrow D'' \cup {D_i}'$
end
return D''

Algorithm 2 demonstrates the details of the DP-Auto model. Our private auto-encoder employs steps to improve the optimization process with gradient computation and clipping. While a gradient is computed for a batch in the standard stochastic training techniques, we compute the gradient for each training instance instead. This approach improves the optimization process since it reduces the sensitivity of the gradient present at each instance [15]. Norms of the gradients define the direction that optimizes the network parameters. However, in some deep networks, the gradients can be unstable and fluctuate in a large range. Such fluctuations can inhibit the learning process due to the increased vulnerability of the networks. To avoid this undesired situation, we bound norms of the previously computed gradients by a clipping constant C [24].

After clipping the gradients, noise is sampled from the Gaussian distribution with zero mean and standard deviation of σC and added to the previously clipped gradients (Line 8 in Algorithm 2). While training the auto-encoder, we track the privacy loss at the end of each batch iteration. As given in lines 2—2, we compute the value of current privacy loss ε' that has been spent on private auto-encoder in a given iteration $t \in T$. Training ends when ε' reaches the final privacy budget ε. If current privacy budget ε' is less than the final privacy budget ε, model parameters of the network are updated with the negative direction of the learning rate η multiplied by the averaged noisy gradients (Line 2 in Algorithm 2). And, current privacy budget ε' is updated by moments accountant technique in Line 2 in Algorithm 2. At the end of this step, the private auto-encoder outputs the model parameter θ based on final privacy budget ε (Line 2 in Algorithm 2).

Algorithm 2 DP-Auto: Differentially private auto-encoder

Require: α: Learning rate; T: iteration number; ε: privacy budget; δ: Gaussian delta; σ: standard deviation; C: clipping constant. $\mathscr{L}$ is the objective function

$\nabla\mathscr{L}$ is the gradient of objective function

initialize θ_0 randomly

$\varepsilon' = 0$

for $i \leftarrow 1$ **to** T **do**

  $B_t \leftarrow$ random batch

  $i_t \sim b$ where $x_{i_t} \in B_t$

  $z_{i_t} \sim \mathscr{N}(0, \sigma^2 C^2)$

  if $\varepsilon' < \varepsilon$ **then**

    $$\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \left(\frac{1}{|B_t|} \sum_{i_t} \left(\nabla\mathscr{L}(\theta_t; x_{i_t}) + z_{i_t} \right) \right)$$

    $\varepsilon' \leftarrow$ calculate privacy loss with moments accountant

  end

end

return θ

1.5 Experiments

In this section, we explain our experimental setting and discuss our results. First we briefly introduce our datasets and detail parameter settings for the used machine learning models. Afterwards, we give our results for two cyber deception tasks: (i) *attacker with no synthetic knowledge* and (ii) *attacker with synthetic knowledge*.

Datasets We evaluate the proposed differentially private deep learning-based honeydata generation approach on four real datasets. The following is a brief description of each dataset:

(i) The **Diabetes** [21] dataset contains the information of **768 female patients** who are at least 21 years old. Each patient is classified as diabetic or non-diabetic. The dataset contains **8 features**.
(ii) The **Adult** [21] dataset contains the information of **45222 individuals**. The dataset shows whether the income of the individuals exceeds 50K US dollars. The dataset contains **15 features**.
(iii) The **BreastCancer(Diagnostic)** [21] dataset contains the information about whether a patient has breast cancer or not. It has **569 patient records** with **32 features**.
(iv) The **Spambase** [21] dataset contains **4601 emails**, each of which is labeled as spam or non-spam. Each instance has **58 attributes**.

Parameter Setting for Data Generation Our DPSYN technique generates synthetic data by using Deep Auto-encoders [5]. An auto-encoder is trained on n data points. Once a model is learned, the auto-encoder can be used to generate *any number of data points* (e.g., honeydata). For n training samples, we report the results of the privacy loss (i.e., the measure of potential leakage to an attacker) using differential privacy with (ε, δ) parameters that is computed from the noise level σ (see Sect. 1.3.3). We fix the δ as $\frac{1}{n}$ and compute the value of ε for each iteration $t \in T$. In moment accountant, we use several noise levels to obtain consistent results. The large noise level ($\sigma = 6.0$) is implemented for small $\varepsilon = 1.0$ and the small noise level ($\sigma = 4.0$) is implemented for large $\varepsilon \in \{2.0, 4.0\}$. In these settings with the increasing ε values, synthetic data generation techniques are perturbed less since small noise is added to these techniques.

In all synthetic datasets (i.e., the generated relational honeydata), biases are initialized to zero, while the initial values of the weights θ are randomly chosen from a zero-mean normal distribution with a standard deviation of 0.05. For each dataset, we form a new auto-encoder to generate its corresponding honeydata.

Parameter Setting for Machine Learning Models We employ four machine learning models in measuring the efficiency of our approach in synthetic data generation for cyber deception: One-class SVM [27], two-class SVM [17], Logistic Regression (LR) [22], and Random Forest (RF) [7]. We chose to employ these methods because they are widely used for classification tasks [20]. Furthermore, these machine learning models will be used to explore whether an attacker can distinguish between the real data vs the honeydata easily.

For the hyper parameter of one-class SVM, we experiment with kernel types $\{linear, poly, rbf\}$ and gamma values $\{1.0, 0.1, 0.01, 0.001\}$. We select the most consistent results of one-class SVM with different (ε, δ) pairs. For two-class SVM, we employ the LinearSVM [14].

Benchmark Techniques In all experiments, we compare DPSYN results with two state-of-the-art synthetic data generation techniques: PRIVBAYES [32] and DPEM$^+$ [23]. We run the experiments 10 times and report the average of the results.

1.5.1 Task 1: Cyber Deception for Attacker with No Honeydata Knowledge

In the first task, we assume that the attacker has knowledge about real data where we model the background knowledge as the number (i.e., $\{50, 100, 200, 400\}$) of data points known to the attacker. This approach is similar to the setting reported in [6]. In this scenario, the attacker does not have access to honeydata samples. We evaluate the quality of the generated honeydata by observing whether the attacker can distinguish the real vs honeydata by leveraging the obtained real data. In these experiments, the attacker employs a one-class SVM model that is built on the $\{50, 100, 200, 400\}$ real data points.

We measure the success of DPSYN by the attacker's failure to separate honeydata and real data by using the classifier. The test data is an equal mix of 50% real and 50% honeydata. *A honeydata generation technique achieves best results if the attacker's SVM model labels all synthetic data as real, which results in a 50% accuracy*.

Figure 1.2 shows the performance of techniques in deceiving the attacker on the four real datasets. In Fig. 1.2a, we show performance for various training number sizes 50, 100, 200, and 400. Each training set is used by increasing noise additions, which is calibrated to $\varepsilon = 1.0, 2.0$, and 4.0, where $\varepsilon = 4.0$ shows the least amount of added noise in the model. As training size increases, performance of all three techniques improves and accuracy values approach 50%. In Spambase and BreastCancer datasets, PRIVBAYES and DPEM$^+$ perform worse compared to Adult and Diabetes datasets. We hypothesize that PRIVBAYES and DPEM$^+$ are more

vulnerable to the curse of high dimensionality; both Adult and Diabetes datasets have less than 15 attributes, whereas BreastCancer and Spambase have more than 32 attributes.

DPSYN results are comparable with those of DPEM$^+$ and PRIVBAYES in the Adult dataset. In Diabetes and Spambase datasets, DPSYN has better accuracy. PRIVBAYES has its worst results in the BreastCancer dataset. In all figures, increasing ε values result in values closer to the desired 50% accuracy value.

1.5.2 Task 2: Cyber Deception for Attacker with Honeydata Knowledge

In the first task, the attacker had access to real data only. In this second task, the attacker has access to both real and honeydata, which may help the attacker in distinguishing honeydata. The knowledge of honeydata implies that attacker was fooled into accepting some honeydata previously; when the attacker fails while using the honeydata (e.g., could not use the honeydata for identity theft), he/she may start analyzing other stored data files to authenticate them.

The attacker employs three binary classification models (i.e., two-class SVM, LR, and RF) that are built on a mix of 50% real and 50% synthetic data. The classifier is trained to learn two labels: real and synthetic (i.e., honeydata). The classifier is tested on a mix of 50% real and 50% honeydata. The accuracy of a model is given as the percentage of correctly classified data points. For the best performance in honeydata generation, the attacker must wrongly classify all honeydata points as real, which results in a 50% accuracy.

In this set of experiments, we report our results for each of the three Machine Learning classification models separately. We begin by demonstrating the Random Forest results.

Figure 1.3 demonstrates accuracy of the three techniques for the Random Forest classifier. The honeydata generated by DPSYN is more similar to real data when compared to the synthetic data generated from PRIVBAYES for Diabetes, Spambase, and BreastCancer datasets. For the Adult dataset, DPEM$^+$ results are closer to the desired 50% level. DPSYN performs better than DPEM$^+$ for Diabetes and Spambase. The failure of DPEM$^+$ and PRIVBAYES on Spambase and Diabetes datasets is expected since the attacker can already distinguish the difference between synthetic and real data with one-class SVM (See Sect. 1.5.1). DPSYN exhibits remarkable improvement for the majority of the RF test cases when compared to DPEM$^+$ and DPSYN.

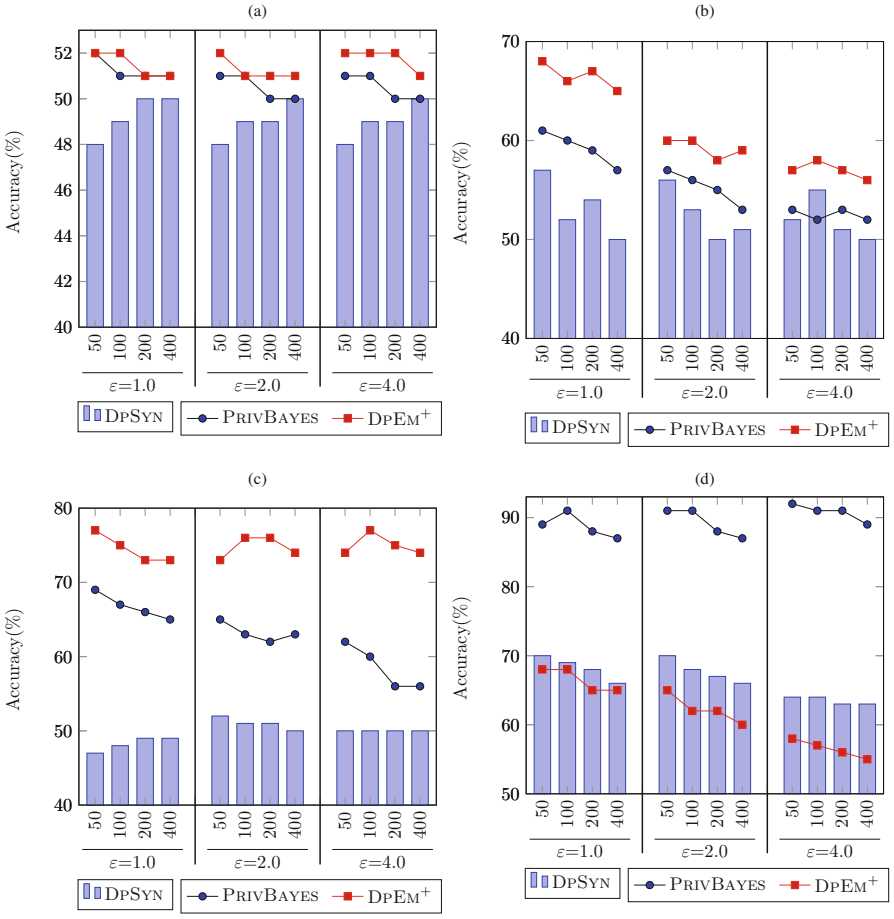

Fig. 1.2 Accuracy results of one-class SVM classifiers of attackers that are modeled on different percentages of real data with varying privacy budgets. The desired accuracy is 50%. (**a**) Adult. (**b**) Diabetes. (**c**) Spambase. (**d**) BreastCancer

The performance of the attacker with Logistic Regression (LR) is demonstrated in Fig. 1.4. The test results of LR are consistent with the other machine learning models previously shown in Figs. 1.3 and 1.2. In fact, on average, LR results are highly correlated (0.94) with RF results in all datasets. However, it is noticeable that attacker with RF is better able to distinguish between real and honeydata when compared to LR; accuracy levels are lower in the LR results. DPSYN outperforms PRIVBAYES in three out of four datasets. For Adult dataset, PRIVBAYES has slightly better results than DPSYN. For this set of experiments with LR, DPSYN results are similar to those of DPEM$^+$ in the Diabetes, Spambase, and BreastCancer datasets.

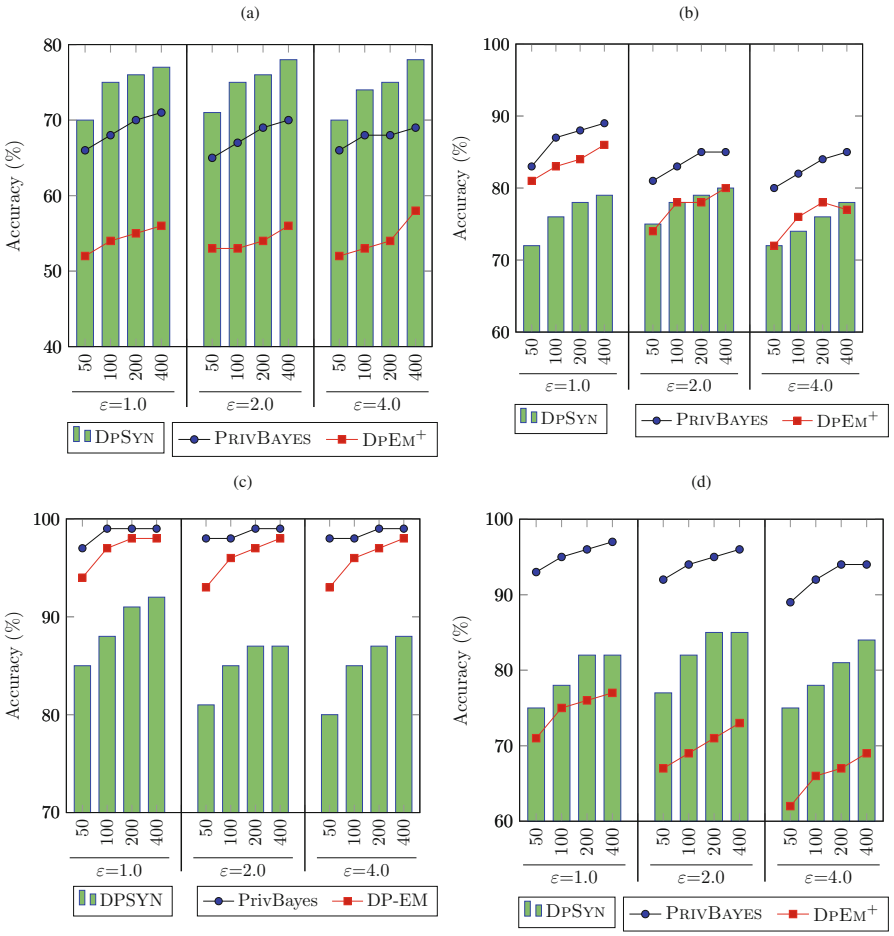

Fig. 1.3 Accuracy results of RF classifiers of attackers that are modeled on different percentages of real data with varying privacy budgets. The desired accuracy is 50%. (**a**) Adult. (**b**) Diabetes. (**c**) Spambase. (**d**) BreastCancer

The performance of the attacker with two-class SVM is demonstrated in Fig. 1.5. Results are consistent with those of LR and RF in Figs. 1.4 and 1.3. Only in the Adult dataset, we see a slight increase of difference between PRIVBAYES and DPEM$^+$ performances with the increase of training numbers.

In all Machine Learning models, our method DPSYN generates honeydata that has better indistinguishability (i.e., the attacker has less accuracy in distinguishing real vs honeydata) for cyber deception. With increasing training dataset size, the classifiers that could be used by the attacker perform better in all data generation techniques. Except for the Adult dataset, DPSYN outperforms PRIVBAYES significantly in all experimental settings. Compared to DPEM$^+$, DPSYN generates better or

comparable synthetic data for the Spambase, BreastCancer, and Diabetes datasets. For the majority of the test cases, increasing ε values harm the attacker machine learning model. However, increasing ε values result in less added noise which may cause the leakage of sensitive data. Hence, there is a trade-off between the quality of honeydata and the prevention of sensitive data leakage.

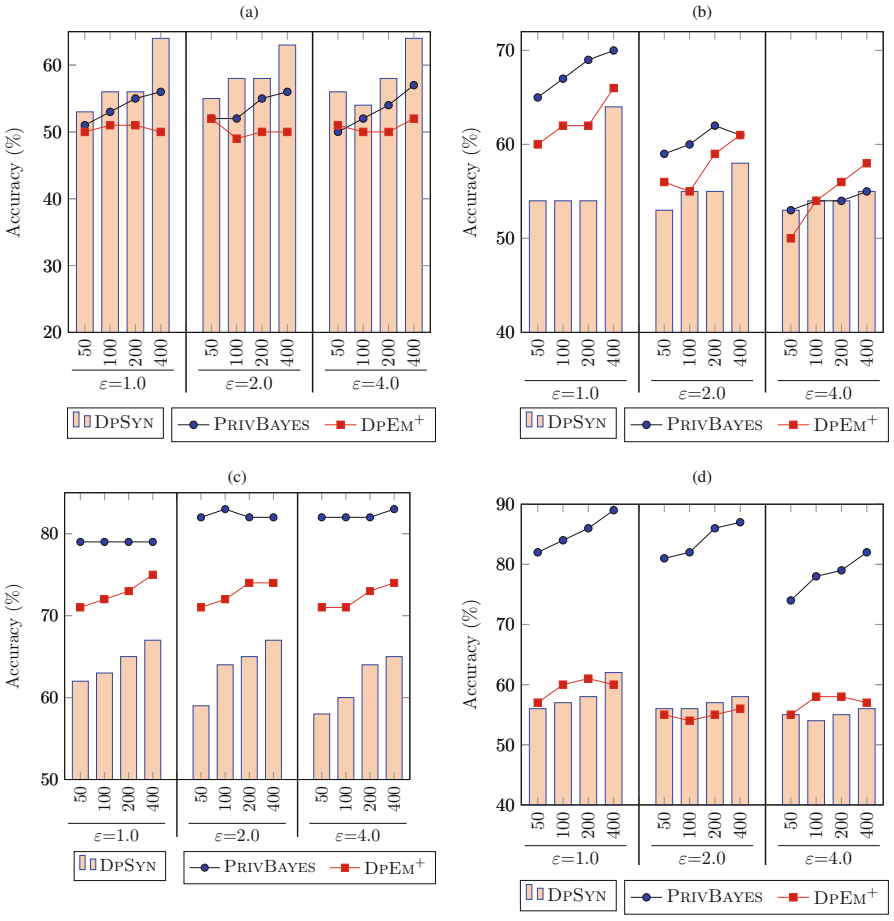

Fig. 1.4 Accuracy results of LR classifiers of attackers that are modeled on different number of real data with varying privacy budgets. The desired accuracy is 50%. (**a**) Adult. (**b**) Diabetes. (**c**) Spambase. (**d**) BreastCancer

1.6 Conclusions

In this book chapter, we explore the applicability of using privacy-preserving deep learning-based synthetic data generation techniques for creating HoneyData that can fool potential cyberattackers. We define a machine learning (ML)-based metric (i.e., the accuracy of any ML model in distinguishing real vs HoneyData) to measure the goodness of generated deceptive HoneyData. Although, our results indicate that existing techniques could be leveraged for HoneyData generation, care must be taken in setting the privacy parameters used in HoneyData generation.

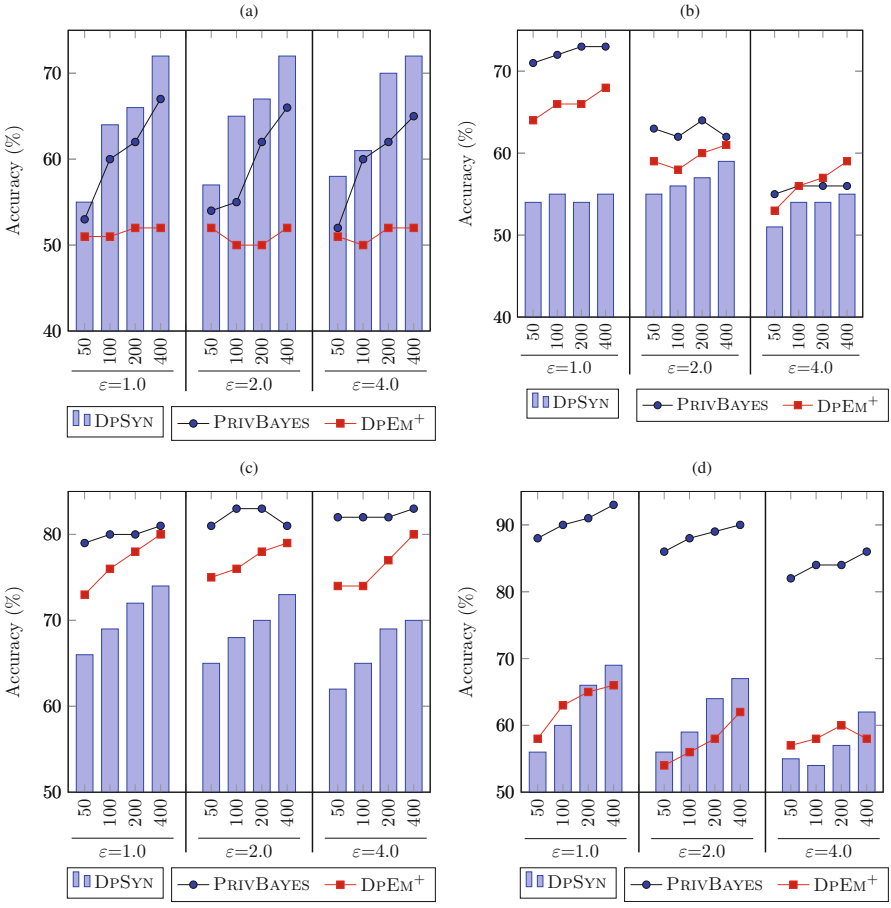

Fig. 1.5 Accuracy results of binary SVM classifiers of attackers that are modeled on different percentages of real data with varying privacy budgets. The desired accuracy is 50%. (**a**) Adult. (**b**) Diabetes. (**c**) Spambase. (**d**) BreastCancer

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 308–318. ACM (2016)
2. Abay, N.C., Zhou, Y., Kantarcioglu, M., Thuraisingham, B., Sweeney, L.: Privacy preserving synthetic data release using deep learning. The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (PKDD 2018) (2018)
3. Ács, G., Melis, L., Castelluccia, C., Cristofaro, E.D.: Differentially private mixture of generative neural networks. CoRR **abs/1709.04514** (2017). URL http://arxiv.org/abs/1709.04514
4. Almeshekah, M.H., Spafford, E.H.: Cyber security deception. In: Cyber Deception, pp. 23–50. Springer (2016)
5. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning, pp. 37–49 (2012)
6. Bindschaedler, V., Shokri, R., Gunter, C.A.: Plausible deniability for privacy-preserving data synthesis. Proceedings of the VLDB Endowment **10**(5), 481–492 (2017)
7. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
8. Bun, M., Steinke, T.: Concentrated differential privacy: Simplifications, extensions, and lower bounds. In: Theory of Cryptography Conference, pp. 635–658. Springer (2016)
9. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: Advances in Neural Information Processing Systems, pp. 289–296 (2009)
10. Dwork, C.: Differential privacy. In: Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II, ICALP'06, pp. 1–12. Springer-Verlag, Berlin, Heidelberg (2006). DOI 10.1007/11787006_1. URL http://dx.doi.org/10.1007/11787006_1
11. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 486–503. Springer (2006)
12. Dwork, C., Lei, J.: Differential privacy and robust statistics. In: Proceedings of the forty-first annual ACM symposium on Theory of computing, pp. 371–380. ACM (2009)
13. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)
14. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. Journal of machine learning research **9**(Aug), 1871–1874 (2008)
15. Goodfellow, I.: Efficient per-example gradient computations. arXiv preprint arXiv:1510.01799 (2015)
16. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
17. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. IEEE Intelligent Systems and their applications **13**(4), 18–28 (1998)
18. Holz, T., Raynal, F.: Detecting honeypots and other suspicious environments. In: Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC, pp. 29–36. IEEE (2005)
19. Juels, A., Rivest, R.L.: Honeywords: Making password-cracking detectable. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 145–160. ACM (2013)
20. Kotsiantis, S.B., Zaharakis, I., Pintelas, P.: Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering **160**, 3–24 (2007)
21. Lichman, M.: UCI machine learning repository (2013). URL http://archive.ics.uci.edu/ml
22. Nerlove, M., Press, S.J.: Univariate and multivariate log-linear and logistic models, vol. 1306. Rand Santa Monica (1973)

23. Park, M., Foulds, J., Chaudhuri, K., Welling, M.: Practical privacy for expectation maximization. CoRR **abs/1605.06995** (2016). URL http://arxiv.org/abs/1605.06995
24. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318 (2013)
25. Rubin, D.B.: Discussion statistical disclosure limitation. Journal of official Statistics **9**(2), 461 (1993)
26. Rubinstein, B.I., Bartlett, P.L., Huang, L., Taft, N.: Learning in a large function space: Privacy-preserving mechanisms for SVM learning. arXiv preprint arXiv:0911.5708 (2009)
27. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001). DOI 10.1162/089976601750264965. URL https://doi.org/10.1162/089976601750264965
28. Song, S., Chaudhuri, K., Sarwate, A.D.: Stochastic gradient descent with differentially private updates. In: Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE, pp. 245–248. IEEE (2013)
29. Spitzner, L.: Honeypots: tracking hackers, vol. 1. Addison-Wesley Reading (2003)
30. Vaidya, J., Shafiq, B., Basu, A., Hong, Y.: Differentially private naive Bayes classification. In: Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01, pp. 571–576. IEEE Computer Society (2013)
31. Yuill, J., Zappe, M., Denning, D., Feer, F.: Honeyfiles: deceptive files for intrusion detection. In: Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC, pp. 116–122. IEEE (2004)
32. Zhang, J., Cormode, G., Procopiuc, C.M., Srivastava, D., Xiao, X.: Privbayes: Private data release via Bayesian networks. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pp. 1423–1434. ACM (2014)
33. Zhang, J., Zhang, Z., Xiao, X., Yang, Y., Winslett, M.: Functional mechanism: regression analysis under differential privacy. Proceedings of the VLDB Endowment **5**(11), 1364–1375 (2012)