UCLA
UCLA Electronic Theses and Dissertations

Title
The Impact of Synthetic and Real Training Data on Model Vulnerability

Permalink
https://escholarship.org/uc/item/58x0h54c

Author
Swoveland, Jacob Michael

Publication Date
2023

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
Los Angeles

The Impact of Synthetic and Real Training Data on Model Vulnerability

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Statistics

by

Jacob Michael Swoveland

2023

© Copyright by
Jacob Michael Swoveland
2023

ABSTRACT OF THE THESIS

The Impact of Synthetic and Real Training Data
on Model Vulnerability

by

Jacob Michael Swoveland
Master of Science in Statistics
University of California, Los Angeles, 2023
Professor Guang Cheng, Chair

Membership inference attacks can threaten the privacy of records in machine learning models by enabling adversaries to determine whether or not a record was used to train said model. In this paper we will be exploring the use of synthetic training data to defend against this form of attack. Synthetic data here keeps the attributes of the original training data set while maintaining machine learning utility. We use CTGAN and DP-CTGAN in order to generate high quality tabular synthetic training data. We evaluate the effectiveness of this approach empirically by comparing the vulnerability and utility of models trained with synthetic and real data. We also analyze the privacy-utility trade-off that comes with using synthetic data. Synthetic data seems to be a promising defense mechanism against membership inference attacks by providing increased privacy at reasonable utility losses.

The thesis of Jacob Michael Swoveland is approved.

Mark S. Handcock

Xiaowu Dai

Guang Cheng, Committee Chair

University of California, Los Angeles

2023

To all those that supported me

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express sincere gratitude to all those who have contributed to the successful completion of this piece of research. First, I would like to express appreciation to my advisor Guang Cheng and committee members for their guidance and support that was provided throughout the duration of this project. I am also grateful to Dr. Chi-Hua Wang, whose mentoring, guidance, and conversations have played a pivotal role for this paper. Finally, I would like to thank the colleagues in my lab with who I have participated in valuble discussions with. I am fortunate to have spent time in an academic environment with such colleagues and mentors.

CHAPTER 1

Introduction

Organizations can harness the power of machine learning to analyze data effectively. For instance, a company may collect information about their individual customers in order to provide better services or advertisements. Given this collected training data, a model is trained to set some numerical parameters in order to generalize to unseen cases. In many cases, the training data can essentially be discarded as the tuned parameters have distilled that information for the organization's use case. There can be other reasons to not share training data as it may contain sensitive information, unlike the model which has replaced this sensitive information with tuned parameters.

However, the very use of training data within the model can leave the individuals within the data set vulnerable to a membership inference attack (MIA) as proposed by Shokri et al. [15] This kind of attack seeks to classify whether a specific record was used within a training data set for a machine learning model. It is also possible for the attacker to only have access to the output of the model in order to make such a determination. Such an assumption would lead us to the black box membership inference attack which will be explained in more detail later. So, an analyst may gain insight into the training data set and figure out specific individual customers at the organization based off of the outputs of the trained machine learning model. In the sections that follow, we will explore this kind of attack along with ways to mitigate its effectiveness.

The suggested effectiveness of this type of attack may come from model overfitting to the training data and the structure of the data in a way that can be identifiable by a classifier.

Training data may naturally be more accurately classified by a machine model as this is the very information that was learned from. Extreme lack of generalization can lead to extreme vulnerability. Certain data characteristics could lead a data set to be more vulnerable to this form of an attack. In a classification problem, the larger the number of possible classes, the more information that can be released by a model. Certain extreme values within the data could leave outliers vulnerable as well.

This paper builds on this previous research, suggesting that the use of differential privacy techniques and synthetic training data can aid in mitigating the effectiveness of a membership inference attack. Differential privacy in this paper will refer to ϵ-differential privacy as proposed by Dwork. [6] The general idea is that ϵ-DP offers a mathematical definition to data privacy by injecting noise into a data set, with a more rigorous definition to be provided later. Synthetic data generation offers a way to learn the characteristics of a data set in order to generate a new data set that shares enough of the same qualities of the training data to be useful to a machine learning task.

In this paper specifically, we will be taking a look at an empirical attack against a machine learning model using different synthetic data generation methods to create training data sets. In order to generate this synthetic data, we will take a look at conditional tabular generative adversarial networks (CTGAN) based generation along with differentially privatize CTGAN (DP-CTGAN) in order to better protect the training data set. The idea being that we should add layers of noise and obfuscation between the real training data from the machine learning model in order to better protect that original information. This additional noise would inherently cost a loss of utility for the machine learning model at the gain of privacy.

CHAPTER 2

Membership Inference Attack Overview

In this section, we will go over the membership inference attack framework as proposed by Shokri et al. [15] A membership inference attack seeks to recover the records of the training data set from a target machine learning model. This target machine learning model will be referred to as the victim model with a training data set referred to as a victim training data set. There are many different assumptions that can be made for membership inference in order to satisfy the different needs of rational firms weighing utility-privacy trade-offs in different scenarios.

For instance, if we assume that the attacker would have access to the internal workings of a model such as the weights then this would be considered a white-box membership inference attack. [13] Such a case may be considered when dealing with a firm's internal employees or if an adversary has gained access to a firm's internal files. For instance, a machine learning engineer may have access to the internal workings of a model but should not be able to retrieve sensitive information about specific individuals. This is usually handled through procedure protocols and security access but can also be possible to circumvent this with a membership inference attack. We will be dealing with a black-box assumption for this paper in which the attacker does not have any access or knowledge of the internal workings of the victim model and only has access to the output of the model. [10] Such a case may be considered when dealing with a firm's internal employees who have restricted access to information or for a third party's access to a machine learning tool. For instance, a low level data analyst, whether external or internal to an organization, should not have access to the

specific weights in a model but does need access to outputs in order to analyze data.

In this case we can further make assumptions on what the output of the model actually is. For a classification problem it is natural to generate output that would be of use to an end user. It is possible to simply state the most probable class but in some contexts this may be inadequate for the decision making process of the end user. In such a scheme, uncertainty measures have been lost so the end user may be led to an incorrectly confident decision. For instance, if there are only two categories split between a probability vector of .51 and .49, then it would be useful for the end user to know that we should not be confident in our knowledge for the classification of this record. In such a case, it may make more sense to report the classification along with a probabilities for each class or for a limited set of classes. This choice for the limitation of an output further leads to a utility-privacy trade-off.

We can also consider the possibility to allow for repeated querying of the victim model or not. There is a utility-privacy trade-off to consider when making this decision. If allowed to query the model multiple times, an end user will be able to evaluate records as they need but will also be able to gain valuable insights into the behavior of the machine learning model. If not allowed to query the model an unlimited amount of times, then the user will not gain nearly as much insight into the machine learning model at the cost of possibly not evaluating records that they are wanting to.

We will explore the following sets of assumptions here. We will make a black-box assumption in which the attacker only has access to the output of the model. This output will be the classification of the record along with the full vector of probabilities for each class. We will also allow the model to be queried repeatedly as well. In this way, we will explore the idea of allowing near unlimited access to a model to a third party end user or to an internal employee who will have some limitations to not have access to model weights imposed upon them by the firm. Such a set of choices may be made by a firm choosing to maximize utility for end users while still having some considerations for data privacy and protection. A different firm with different goals, legal obligations, or sensitivity levels of data

may make different rational decisions that will not be explored in this paper.

2.1 Membership Inference Attack

There are several possible explanations as to why membership inference attacks may be effective. This attack may exploit a possible train-test gap that is seen in machine learning in which a model will possibly react differently to training data than to do that has not been seen by the model before. Let us consider a classification problem. A model may consistently be more confident in its predicted class of a training record given its true class when compared to an unseen record. [9] The larger such a gap, the more vulnerable a model would be as the train and test distributions grow more disjoint. It is for this reason that over fitting of the model, whether on a local or global level, may lead to more effective membership inference attacks. Some model frameworks or data set structures appear to also be more vulnerable to this form of attack as well. Classification tasks with high number of possible classes appear to be more vulnerable to this form of attack. This may be due to how much information is present within an classification task output. A binary output between two classes is essentially a single signal as the probability of a class is the complement of the other. If there are large number of classes within a prediction probability vector, then there can be more useful information to exploit.

Let us look at membership inference attacks more formally. Consider the target victim model $f_{\text{target}}()$ with its associated training set of $D^{\text{train}}_{\text{target}}$. This data set contains labeled records $x^{(i)}_{\text{target}}$ input data and $y^{(i)}_{\text{target}}$ true label values from among the possible c_{target} possible categories of the target model. In this case, the output of the model is a probability vector with elements for each possible category $\mathbf{y}$. Now consider the attack model f_{attack} which has the goal to classify whether a record is in (1) or out (0) of the target's training data set. In formal terms, this can be seen as $\Pr\{(\mathbf{x}, y), \mathbf{y} \in D^{\text{train}}_{\mathbf{target}}$. The input to this f_{attack} is both the attributes of a record $\mathbf{x}$, the target output vector $\mathbf{y}$, and the true label value y in order to

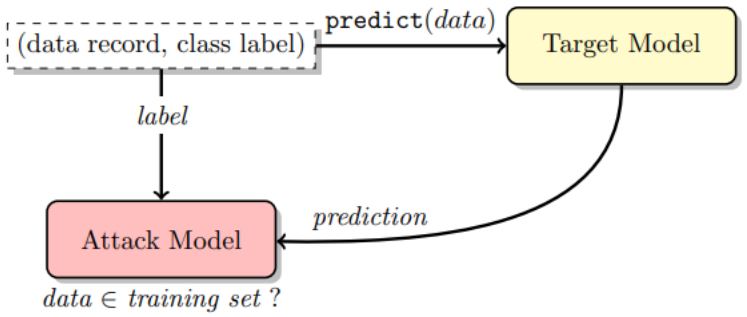

Figure 2.1: Sample Membership Inference Attack Workflow

create a classification confidence values for each record. See figure 2.1 for an example of this.

The reason to include the true label value y for a record is due to the fact that the output classification probability vector $\mathbf{y}$ heavily depends on the true value of the record. For instance, whether a record is classified as a dog heavily depends on whether or not that record is truly a dog. In the following sections, we will break down these steps further. In order to build an attack model, we will first have to create training data for that model which we will explore with shadow modeling.

2.2 Shadow Modeling

Shadow modeling aims to imitate the intricacies of how a victim machine learning model will behave when trained with differing data sets. A shadow model is a machine learning model that is created which is the same structure as the victim machine learning model but with a differing training data set. For instance, if the victim model is a convolutional neural net, the shadow model should also be a convolutional neural net. In essence, an attacker will seek to create k shadow models which have differing training data sets to see how different training records would create differing responses by machine learning models. With these k shadow models, we will create a new data set in which to train a classifier, known as an attack model, to predict whether a record is within the victim model's training data set or not. Because

we have the labels for the training and test data within each shadow model, this data set that is created will eventually be used to train a binary classification supervised machine learning model. It is through this that we are exploring the possible sample space of machine learning behavior. If the shadow model structure is correctly specified to match the victim model structure, then the shadow modeling technique will be more effective. However, a misspecification can lead an attacker to perform a less effective attack. For this reason, we explore only a worst case scenario in which the model is exactly matching.

Let us look at this shadow modeling framework more formally. An attacker will create k shadow models. Each shadow model i will train on a data set $\mathrm{D}^{\mathrm{train}}_{\mathrm{shadow}^i}$. Each of these shadow models will be trained in a manner as close to the target victim model as possible. We will also keep the training data for the shadow model disjoint from the victim training data ($\forall i \mathrm{D}^{\mathrm{train}}_{\mathrm{shadow}^i} \cap \mathrm{D}^{\mathrm{train}}_{\mathrm{target}} = \emptyset$). This is done as a worst case scenario for the attacker. As we are interested in the black-box setting it would be more relevant to explore if the attacker did not have internal model and training data information. If they did have that information, then this attack may be easier to conduct. Consider figure 2.2 as a sample shadow model workflow.

In order to do this, an attacker will first need to sample from a data set that is in the same format and a similar distribution as that which was used to the victim model's training data. The way in which an attacker may sample this data is an interesting research question in its own right which will not be explored here. In general, some suggestions have been to generate sample data from the victim model, from prior information about the population distribution, or from a reference data set. For this paper, a reference data set will be used in shadow modeling which will come from the same distribution as the training data but will be disjoint.

Shadow modeling eventually outputs an attack data set of sufficient size to train a machine learning model that is able to distinguish whether a specified record was indeed used to train the victim model. This training data set will then be used to train an attack model

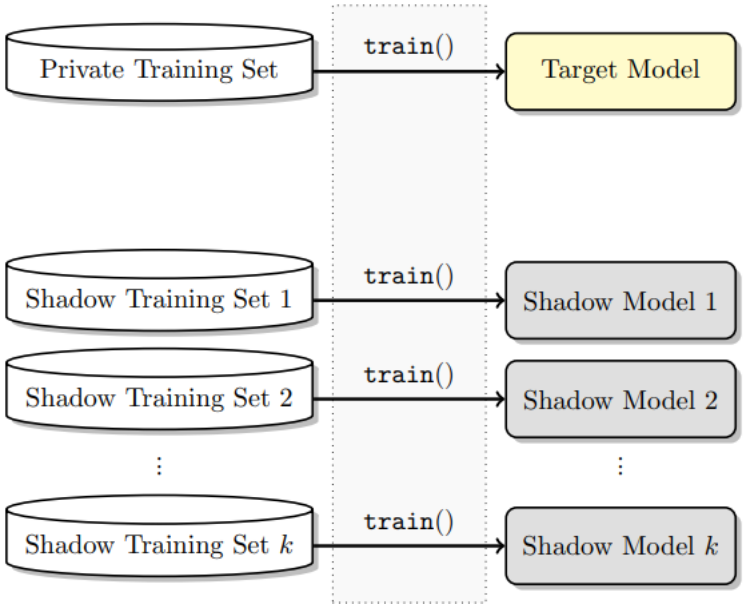

Figure 2.2: Sample Shadow Modeling Workflow

to discern whether or not a record is within the training data or not.

2.3 Attack Model

Let us now consider the attack model which will be trained with data from shadow modeling. This attack model will be trained in the binary classification task of whether a record is within a training data set or not. Let us look into this training. For simplicity, let us consider that half of the records used in a shadow model will be inside the training data and half outside. We shall label those records within the training set as 1 and those outside as 0. We will query each shadow model with the associated training and testing set in order to obtain the outputs associated with each record. Now that we have our model outputs prepared and labeled, this training of the attack model can be seen as a supervised machine learning problem. In order to boost the accuracy of the attack model, it is suggested to train an individual model for each category found within a data set. [15] For instance, if there are only "high income" and "low income" found within the output categories, then there should

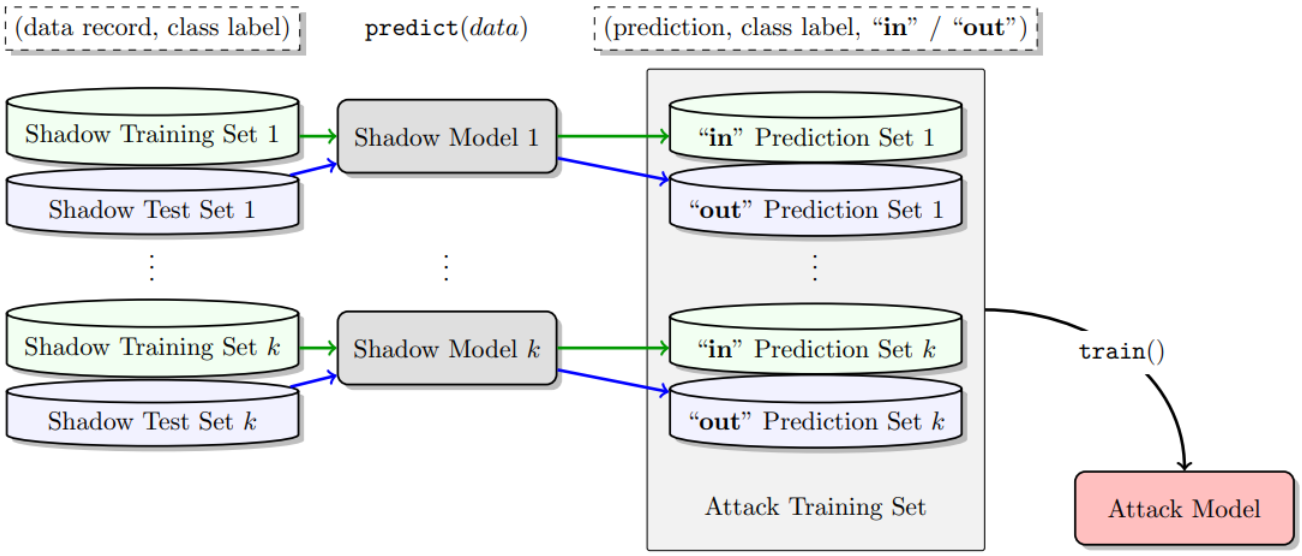

Figure 2.3: Sample Attack Classifier Training Workflow

be two attack models trained. One will aim to learn the intricacies of attacking those of high income while the other will be better suited to those with "low income". After training the model, records can be fed to the appropriate attack model in order to launch a membership inference attack.

Let us view this more formally. Figure 2.3 is a sample workflow as will be described. For all $(\mathbf{x}, y) \in D^{\text{train}}_{\text{shadow } i}$, compute the associated $\mathbf{y} = f^i_{\text{shadow}}(\mathbf{x})$. Add the associated records $(y, \mathbf{y}, 1)$ to the attack training data set $D^{\text{train}}_{\text{attack}}$ where 1 is indicating that these records are within the training data set. Along the same idea, for all $(\mathbf{x}, y) \in D^{\text{test}}_{\text{shadow } i}$, which are disjoint from $D^{\text{train}}_{\text{shadow } i}$, compute the associated $\mathbf{y} = f^i_{\text{shadow}}(\mathbf{x})$. These records $(y, \mathbf{y}, 0)$ will also be added to the attack training data $D^{\text{train}}_{\text{attack}}$ where 0 is indicating that these records are not within the training data set. Let us now split $D^{\text{train}}_{\text{attack}}$ into each of its associate c_{target} partitions for each category label. For each label, train a separate attack model that will predict whether or not a record is in (1) or out (0) of the data set. In this way we can see each individual category label as a binary classification problem.

Through this method, we try to learn the intricacies of how a machine learning model may provide differing confidence scores depending on different records. Most importantly, we see how high confidence scores may be assigned to training data and to testing data in order to differentiate these cases. In the next section, we will be going over the concepts of differential privacy and synthetic data. We will be applying these concepts in order to guard against this kind of attack method later.

2.4 Differential Privacy

Differential privacy is a concept within the field of data privacy that aims to protect the private information of an individual while still preserving the utility of the data for analysis which was first introduced by Dwork. [4] The idea is to add in noise to the data set such that it becomes more difficult to identify any individual within a data set. The more noise that is added to a data set, the more protection there will be but the less utility there will be for data scientists. Differential privacy achieves theoretical upper bounds on this privacy gain via an algorithm and the Laplacian noise mechanism. This idea can be achieved for any data set size, but the noise required for small data sets can be extreme. At very large data set size, the noise can be comparatively small and there may be very little actual change within the data set.

Let us look at this concept more formally. A main idea in differential privacy is that of two neighboring data sets D1 and D2 which are entirely the same except by a single record. A dataset is differentially private if an adversary can learn the same information from both data sets and nothing more. [5] This is important is the way privacy is defined within this framework in that the outcome of a statistical analysis should vary very little whether or not an individual is within a data set. So if two data bases D1 and D2 differed by a single person, then the outcome for that person should be the same in both data bases even though this person's record is not in one of the data bases. We will eventually look into (ϵ, δ)-differential

privacy so let up break down these terms. ϵ refers to to the maximum distance between a query on data set D1 and a different data set D2. It is a numeric measure in which we can control the privacy-utility trade off as lower values favor privacy while higher values favor utility. As such, it is often referred to as the privacy budget. δ is the probability that information is accidentally leaked.

Let us look at two scenarios, one where $\delta = 0$ and another where $\delta > 0$. When $\delta = 0$, this is often referred simply as $\epsilon-$differential privacy. Let us now look at the mathematical definition of $\epsilon-$differential privacy. [1] Define a randomized function M to be $\epsilon-$differentially private if for all neighboring data sets D1 and D2 which differ by at most one element and $\forall S \subseteq$ Range(M). Then we have:

$$\frac{Pr[M(D1) \in S]}{Pr[M(D2) \in S]} \leq e^{\epsilon} \tag{2.1}$$

Which is equivalent to the following:

$$Pr[M(D1) \in S] \geq e^{\epsilon} * Pr[M(D2) \in S] \tag{2.2}$$

$(\epsilon, \delta)-$differential privacy is very similar. Define a randomized function M to be $(\epsilon, \delta)-$differentially private if for all neighboring data sets D1 and D2 which differ by at most one element and $\forall S \subseteq$ Range(M). Then we have a modification to (2) such that:

$$Pr[M(D1) \in S] \geq e^{\epsilon} * Pr[M(D2) \in S] + \delta \tag{2.3}$$

While there are many different techniques to implement these theoretical bounds, the most common technique is the Laplace mechanism. The Laplace mechanism adds noise based on the Laplace distribution with mean 0 and sensitivity of $(\frac{s}{\epsilon})$ to each query result. This mechanism assumes a δ of 0 which makes it a popular choice since ϵ is the only parameter and the Laplace distribution is well understood. In this paper, we will be using differential privacy mechanisms in order to experiment with the creation of deferentially private synthetic data. In the following sections we will be going over the idea of using synthetic data

instead of real data in order to train a machine learning model. In this case, the idea is to use synthetic data in order to protect the real data from a membership inference attack. To explain synthetic data, let us contrast it with real data. Real data comes from direct observation and measurement from real phenomena. Synthetic data does not come from any observations but instead is generated by learning from and mimicking real data. This synthetic data should follow the same distributions, patterns, and conditional relationships that are found within the real data in which it is based on. This is referred to as the fidelity of the data. It should also perform similarly in machine learning tasks as the real data. This is referred to as the utility of the synthetic data. It is also important to consider the privacy of synthetic data as well as it is important not to leak sensitive information. In practice, these three pillars, fidelity, utility, and privacy, of synthetic data are important to consider and evaluate in order to create high quality synthetic data. This can be difficult to do in practice. In the next section, we will go over the different methods used in this paper in order to generate synthetic tabular data.

CHAPTER 3

Synthetic Data Generation

In this section we will go over the methods used to generate synthetic data for this paper. Tabular data is data that is organized in a table like structure where each column represents attributes of a specific record and each row represents a specific record within the table. Data within these attribute columns can be numeric or categorical variables, however most of the interesting tables of data contain some attributes that are numeric and categorical. Such interesting tables will now be referred to as mixed data type tables. Mixed data tables offer challenges to synthetic data generation which will be explored in this section.

3.1 GAN Framework

Let us now consider an overview of the Generative Adversarial Network (GAN) framework as proposed by Goodfellow et al which is popular for use in generating synthetic data due to its flexible nature. [8] GANs are a type of unsupervised deep learning model in which two neural networks, a generator G and discriminator D, play a two player game in order to generate new samples that are similar to a training data set. The generator in this game samples noise and passes that into a network which then will create samples. The discriminator will take samples of real and generated data and try to distinguish between the two. These two players improve from playing this game to eventually produce better quality synthetic data.

Let us look at this more formally. The generator G learns a mapping $G(\mathbf{z}; \theta_g)$ over the data $\mathbf{x}$ given a prior of input noise $p_z(z)$. Typically this noise vector z is sampled from a

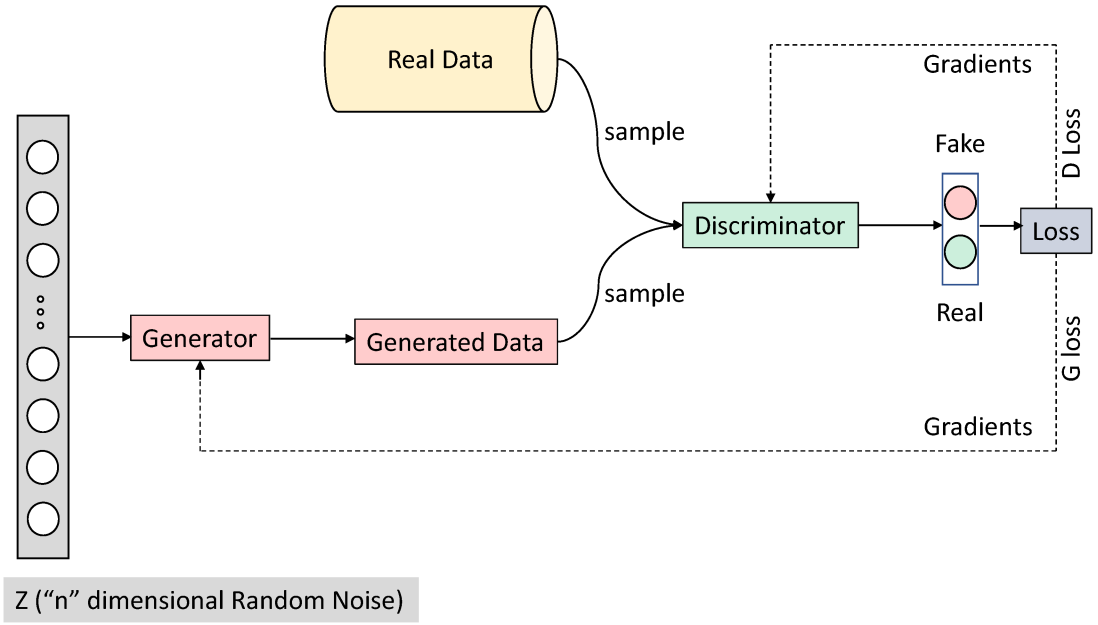

Figure 3.1: GAN Overview

standard normal as this presents a very easy way to sample. The discriminator D attempts to distinguish between records sampled from the distribution of the training data p_{data} and the distribution of generated samples p_g. In essence, D in this case can be thought of as a binary classifier aiming to classify samples originating from real data as 1 and samples originating from the generator as 0. D seeks to assign the correct classification to real and generated samples and G seeks to generate a sample that is indistinguishable from real data. See figure 3.1 for an overview of this structure. [8]

In this way, G and D play the following mini-max game with the value function V(G,D):

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(z))] \tag{3.1}$$

At the beginning of this game, G produces samples that are not like the true data samples and D will have an easier time to distinguish between real and synthetic samples. As the game progresses, it is desired that G will produce samples that are of sufficient quality that D will not be able to distinguish between real and synthetic samples. In essence D will

provide a random guess to every record, i.e. the distribution of the generated samples will be equivalent to the training data, $p_g = p_{data}$, and the probability of D providing a correct classification P(D) will be $\frac{1}{2}$. Using the Jensen-Shannon divergence between two distributions and certain assumptions upon G and D, this convergence does happen as a unique solution.

In practice, this GAN framework can present problems for generating synthetic data beyond the typical instability issues that GANs face. [2] For instance, mixed data types must be converted into sensible numeric representations in order to properly fit into this framework. Mode collapse is where multi-modal distributions are not accurately modeled and instead the most common mode is the only portion of the distribution learned from by the generator. [12] Highly sparse training data can also create issues for generators and discriminators as these portions of the distribution may not be well explored by the generator or discriminator. This leads to a bias against minorities within a dataset. It can become increasingly difficult to detect that this is indeed occurring within a GAN as the size of the minority category relative to the entire distribution decreases. A group of 100 records among 100,000 that are not well represented will not cause any noticeable decrease in any downstream machine learning tasks but may indeed be discriminatory towards that group. For a single record, the idea to favor the most probable category when generating data may make sense, but for an overall data set this can lead to imbalances that are unrepresentative or discriminative synthetic data sets. Some of these issues, such as mixed data types, are more unique to tabular data but others are more general to other types of data as well.

3.2 CTGAN

Conditional tabular generative adversarial networks (CTGAN) as proposed by Xu et al offer a way to deal with such problems as discussed above in order to generate representative synthetic data. [16] In order to deal with mixed data tables, categorical variables are encoded via a one-hot encoding in order to do a sensible pre-processing transformation to feed data

to the GAN framework before generating synthetic data.

In order to tackle the issue of mode collapse for continuous columns, CTGAN uses a method of mode-specific normalization in which the number of modes for each column is estimated via a variational gaussian mixture model (VGM) and then encoded via a one-hot labeling for each of the modes. In order to fully explore each mode, once a mode is sampled from this one-hot vector, a value is generated and then scaled up to reasonable values found within the mode. In this way, we can sample from multiple modes and generate diverse values to fill the sample space appropriately. More formally, Xu et al expressed this as the following 3 step process. [16]

1. For each continuous column C_i, use a VGM to estimate the number of modes m_i. Each mode η_k has a weight μ_k and standard deviation ϕ_k which can be represented within the learned Gaussian mixture of $\mathbb{P}_{C_i}(C_{i,j} = \sum_i^k \mu_k \mathcal{N}(C_{i,j}; \eta_k, \phi_k$

2. For each value $c_{i,j}$ in each continuous column C_i compute the probability of that value coming from the mode. The probability densities are calculated as $\rho_k = \mu_k \mathcal{N}(c_{i,j}; \eta_k, \phi_k$.

3. Sample one mode from the probability density, and use the sampled mode to normalize the value. This value within the mode is represented as $\alpha_{i,j} = \frac{c_{i,j} - \eta_k}{\phi_k]}$

For a sample workflow, see figure 3.2 which is an example with 3 modes.

CTGAN also tackles the issue of sparsity and training imbalance that is often an issue within the GAN framework. GANs can face this issue when sampling training records randomly or uniformly as this can lead to serious under sampling of sparse categories. CTGAN seeks to sample training data more evenly via adding in a conditioning layer. Here the idea is that the generator can be seen as sampling from a conditional distribution of the given discrete categorical value in order to generate synthetic row values for a given sampled record. Here the generator will first pick a categorical column to generate a value for, then sample a record which has a given value. We can then condition the synthetic record's column value

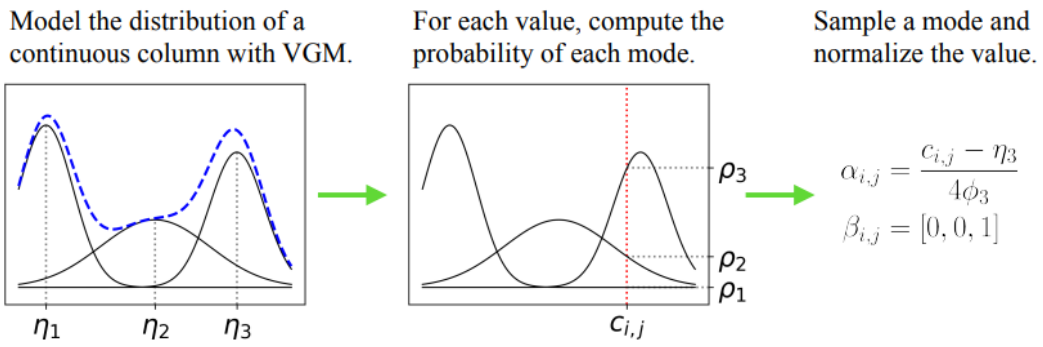

Figure 3.2: Mode-Specific Normalization Workflow Example

with a condition upon this information. The sampling of these within column values are sampled by log-frequency which can explore the sample space more evenly than uniform sampling.

More formally, let k^* be the value from the i^*th categorical column D_{i^*} which is associated with generated row $\hat{\mathbf{r}}$ from generator G. Where the generator conditional distribution learns from the real conditional distribution: $\hat{\mathbf{r}} \sim \mathbb{P}_G(row|D_{i^*} = k^*) = \mathbb{P}(row|D_{i^*} = k$. Such that original distribution can be reconstructed as $\mathbb{P}(row) = \sum_{k \in D_{i^*}} \mathbb{P}_G(row|D_{i^*} = k^*)\mathbb{P}(D_{i^*} = k)$. The critic could then compare the distance between the real conditional distribution and generated conditional distribution to determine whether or not the row is synthetic or real. For a sample workflow see figure 3.3 which is an example of a generator learning a conditional distribution which is then compared to the real conditional distribution by a critic.

It is for these reasons that CTGAN was used in this paper to generate synthetic tabular data. In this paper, we use the implementation of CTGAN as written in Python's CTGAN library, also published by Xu et al. [16]

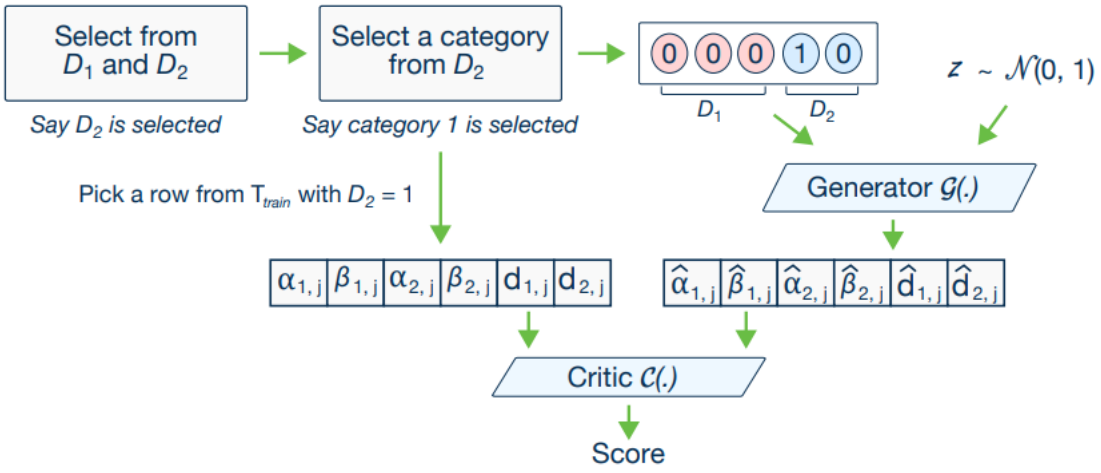

Figure 3.3: Sample CTGAN Workflow

3.3 DPCTGAN

In a previous section we have gone over the definition of differential privacy. There are many ways to implement differential privacy and where to add noise to best protect data. In this section, we will be going over one possibile implementation of differentially private conditional tabular generative adversarial networks (DP-CTGAN) as used in the smart-noise synth package in python. [7] This work is based on the previous paper by Xu et al, with choices of where to inject noise in order to reach a state of being differentially private. There are the options of when to inject noise to make differentially private data. We can inject noise into the training data, into the output data, or both. In this case, we will be injecting noise into both the training and output data. There will be a separate ϵ budget for the pre-processing privacy budget for the training data and the post-processing privacy budget for the data output by the CTGAN. This usage of differential privacy can increase the privacy of the records within a data set but can decrease the utility of the data rapidly as ϵ values decrease.

CHAPTER 4

Analysis

Here we will explore when it may be useful to use synthetic training data for a machine learning model or not. We will be measuring the privacy/utility trade-off as a trade off between the effectiveness of a membership inference attack against the machine learning utility. In this case, we will explore this idea for varying training data set sizes. For instance, the privacy gains could be much more impressive at certain sizes compared with others. In practice it is important to take into account the sensitivity of the data along with the amount of available data when deciding how and when to generate synthetic training data.

4.1 Data Description

In this paper we will analyze the the UCI Adult data set which is also known as Census Income. [3] Here, information such as race, gender, marital status, hours worked per week, and native country was taken from the 1994 US Census in order to predict the binary classification task of whether or not an individual has an income greater or less than $50K. The data is a mix of continuous, discrete, and categorical variables. This binary classification problem here can also present a natural defence against membership inference attacks as there is essentially only a single signal that is being sent from the machine learning model output of classification probabilities.

4.2 Experiment Setup

Here will will go over the experiment setup and assumptions made to explore the vulnerability of the Adult data set to membership inference attacks. To do this experiment, we will train a binary classification machine learning model, in this case it is a logistic regression model, with differing training data types and sizes. We will be using real, synthetic, and differentially private synthetic data in order to train this classification model. To generate the synthetic data, we will use CTGAN as previously mentioned and along the same lines we will use DP-CTGAN to generate differentially private data. For DP-CTGAN, we will use ϵ levels of .5 and 2.5 where the privacy budget is split evenly between pre and post processing stages. The differing levels of ϵ are used to explore reasonable values of ϵ that could be used to increase privacy while exploring possible utility loss. For each method, we will train a model with access to 100, 500, 1000, and 5000 records. Each time we will do a 50/50 train test split and will repeat this process 100 times each. We will also note any possible overfitting of the data as well as this can be seen as one possible reason why a membership inference attack may be more effective. This will be measured as the difference in accuracy between the training and testing sets. We shall also measure the accuracy of the machine learning model itself to get a sense of how much utility is lost by using synthetic data.

We make the assumption that the attacker has no access to any of the actual training records. In this case, we have created two disjoint data sets, one of which is available to the victim machine learning model and the other which is available to the attack model. We have also limited the amount of shadow models to 10. It is possible to increase the number of shadow models and the size of said models in order to improve the effectiveness of the attack model as well. We have also assumed that the attacker has a reference data set of the same size as that of the victim set. For instance if the victim set has 1000 records contained, then the attacker will have access to 1000 disjoint records in order to create shadow models. There can also be better performance if the attacker is allowed access to additional records.

In this way, the attacker will be allowed equal number of records that will be used to train 10 shadow models in order to launch a membership inference attack as previously described. We will be interested in the accuracy of the attack as our target metric for vulnerability to a membership inference attack.

4.3 Results

This section will go over the results of the experiment as described. Refer to table 4.1 for information about the effectiveness of the membership inference attack, model utility, and train-test accuracy gap for varying training methods and sizes.

Training with real data had seen modest vulnerability to membership inference attacks for small training set sizes and increasingly less vulnerability for larger training set sizes. Intuitively this low overall vulnerability can make sense as the Adult data set is a binary classification problem which can provide a natural defense against MIA. We will be using this real training data as a baseline to compare synthetic training data against.

Differentially private data had indeed provided additional protection against membership inference attacks even at very low training data set sizes. This privacy has come at great cost to the utility of the data though as the victim model quickly became unable to generalize to data and would randomly guess what the classification of a record should be. This makes sense as the amount of noise required to make a data set of size 50 differentially private can be quite extreme. The train-test gaps remain relatively similar to that of synthetic and real data. At higher training sizes, the utility had recovered somewhat while retaining the privacy gains. With utility this low after using DP, it may be practical to simply not release any information rather than release such a degraded machine learning model if choosing between DP synthetic data and real data to train a model.

Synthetic data seemed to have provided reasonable protection gains at reasonable utility losses in comparison to DP synthetic data. As the training size for the machine learning

MIA Results				
Method	Training Size	Mean Attack Acc	Mean ML Test Acc	Train-Test Gap
Real	50	.5326	.7833	.0356
CTGAN	50	.5160	.6408	.0095
DP-.5	50	.4953	.5598	.0348
DP-2.5	50	.5102	.5214	.0614
Real	250	.5114	.7834	.0127
CTGAN	250	.5108	.7090	.0147
DP-.5	250	.4997	.5197	.0229
DP-2.5	250	.5025	.5061	.0369
Real	500	.5037	.7973	.0025
CTGAN	500	.4990	.7183	.0256
DP-.5	500	.4971	.5092	.0168
DP-2.5	500	.4988	.5068	.0472
Real	2500	.4990	.7947	.0027
CTGAN	2500	.5004	.7947	.0027
DP-.5	2500	.4988	.4918	.0442
DP-2.5	2500	.5051	.5948	.0207

Table 4.1: Membership Inference Attack Results

model increased, the utility approached that of the real data while maintaining a higher level of privacy preservation against membership inference attacks.

CHAPTER 5

Conclusion

In this paper, we have explored the use of synthetic data for training machine learning models in order to protect records from membership inference attacks. The use of synthetic data can lead to a utility loss in the machine learning model so the trade-offs of what type of data to use should be considered. Assumptions such as how much reference information and computational power the attacker has along with how large the data set training the machine learning model should be considered. Here we had explored this concept with the Adult data set. If using small training set sizes, then it would be useful to use synthetic training data for this data set. Else, it seems reasonable for this kind of data to use real training records. This may be due to membership inference not being effective against binary classification. This specific form of membership inference attack relies upon repeated queries so another option to protect against this form of attack is to prevent repeated query by users.

For a future direction, it could be possible to explore why membership inference attacks may be less effective against models trained with synthetic data. It is unclear if this is due to noise added to training data, model utility loss, or due to other factors as well. In this experiment, it seems as though the train test gap has been roughly the same across all methods so it may be the loss of utility that comes from the use of synthetic data that caused increased gains or another property of the data. It could also be interesting to explore if a differently formulated attack could exploit the use of synthetic training data. Another area of exploration would be to explore additional methods to generate high quality synthetic data such as variational auto encoders (VAE) [11] or model based methods such as synthpop

[14].

REFERENCES

[1] Muhammad Aitsam. Differential privacy made easy. *2022 International Conference on Emerging Trends in Electrical, Control, and Telecommunication Engineering (ETECTE)*, pages 1–7, 2022.

[2] Ramiro Camino, Christian Hammerschmidt, and Radu State. Generating multi-categorical samples with generative adversarial networks, 2018.

[3] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[4] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[5] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, 2008.

[6] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, 2014.

[7] Mei Ling Fang, Devendra Singh Dhami, and Kristian Kersting. Dp-ctgan: Differentially private medical data generation using ctgans. In Martin Michalowski, Syed Sibte Raza Abidi, and Samina Abidi, editors, *Artificial Intelligence in Medicine*, pages 178–188, Cham, 2022. Springer International Publishing.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[9] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey, 2022.

[10] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples, 2019.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[12] Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks, 2018.

[13] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, may 2019.

[14] Beata Nowok, Gillian M. Raab, and Chris Dibben. synthpop: Bespoke creation of synthetic data in r. *Journal of Statistical Software*, 74(11):1–26, 2016.

[15] R. Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2016.

[16] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Neural Information Processing Systems*, 2019.