Generating Synthetic Discrete Datasets with Machine Learning

(Discussion Paper)

Giuseppe Manco[1], Ettore Ritacco[1], Antonino Rullo[2], Domenico Saccà[2] and Edoardo Serra[3]

[1]*Institute for High Performance Computing and Networking (ICAR) of the Italian National Research Council (CNR), v. P. Bucci 8/9C, 87036 Rende (CS), Italy*

[2]*DIMES Department, University of Calabria, Rende, CS, 87036, Italy*

[3]*Computer Science Department, Boise State University, Boise, ID 83725, USA*

Abstract

The real data are not always available/accessible/sufficient or in many cases they are incomplete and lacking in semantic content necessary to the definition of optimization processes. In this paper we discuss about the synthetic data generation under two different perspectives. The core common idea is to analyze a limited set of real data to learn the main patterns that characterize them and exploit this knowledge to generate brand new data. The first perspective is constraint-based generation and consists in generating a synthetic dataset satisfying given support constraints on the real frequent patterns. The second one is based on probabilistic generative modeling and considers the synthetic generation as a sampling process from a parametric distribution learned on the real data, typically encoded as a neural network (e.g. Variational Autoencoders, Generative Adversarial Networks).

Keywords

Synthetic dataset, Data generation, Inverse Frequent Itemset Mining, Constraints-based models, Variational Autoencoder, Generative Adversarial Networks, Generative models

This paper is an extended abstract of [1].

1. Introduction

Emerging "Big Data" platforms and applications call for the invention of novel data analysis techniques that are capable to effectively and efficiently handle large amount of data [2]. There is therefore an increasing need to use real-life datasets for data-driven experiments but the scarcity of significant datasets is a critical issue for research papers [3]. Synthetic data generation can help in this, by reproducing the internal mechanisms and dependencies that justify the occurrence of some specific pieces of information, and hence being able to replicate them stochastically. In this paper we focus on the problem of generating high-dimensional discrete data. Generating such data is a challenge because of the combination of both high dimensionality and discrete components, which may result in a complex structural domain with lots of variety and irregularities, not necessarily smooth. Throughout the paper, we shall study approaches to data generation which rely on the idea that each point in the real domain can be mapped into a

SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy

Table 1
Transactional Dataset end Frequent Itemsets.

a	b	c	d	$\sigma^{\mathcal{D}}$
1	1	1	0	40
0	1	1	1	40
1	1	0	0	60
0	1	1	0	20
1	0	1	1	10

(a) Dataset $\mathcal{D}$

S_i	a	b	c	d	σ^{S_i}
S_1	1	1	0	0	100
S_2	0	1	1	0	100
S_3	0	0	1	1	50

(b) Frequent Itemsets with threshold 50

suitable latent space and vice versa. These mappings guarantee a consistency with regards to the original dataset. At the same time, the manifold in the latent space summarizes the main characteristics of the data, that can hence be injected into the synthesized data in a controlled way. We first discuss two main approaches, that are the Inverse Frequent Itemset Mining (IFM) and probabilistic generative modeling (PGM); finally, a comparison of the results obtained with some of the presented algorithms are provided.

2. Inverse Frequent Itemset Mining-based Generative Models

Let $\mathcal{I}$ be a finite domain of n elements, also called *items*. Any subset $I \subseteq \mathcal{I}$ is an *itemset* over $\mathcal{I}$, also called a *transaction*. Let $\mathcal{U}_{\mathcal{I}}$ denote the set of all itemsets on $\mathcal{I}$; then, $|\mathcal{U}_{\mathcal{I}}| = 2^n$. A (*transactional*) *database* $\mathcal{D}$ is a set of tuples $[k, I]$, where k is the key and I is an itemset. The size $|\mathcal{D}|$ of $\mathcal{D}$ is the total number of its itemsets, i.e., transactions. A transactional database $\mathcal{D}$ is very often represented as a bag of itemsets, i.e, the keys are omitted so that tuples are simply itemsets and may therefore occur duplicated – in this case $\mathcal{D}$ is also called a transactional *dataset*. In the paper we shall also represent an itemset $I \in \mathcal{D}$ by its one-hot encoding $\boldsymbol{x}$, that is a binary vector of size n (the number of items in $\mathcal{I}$) such that its i-th position $x_i = 1$ if the i-th item in $\mathcal{I}$ is in I, 0 otherwise. Consequently, $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\eta\}$, where $\eta = |\mathcal{D}|$, is the one-hot encoding of the whole dataset $\mathcal{D}$. For each itemset $I \in \mathcal{D}$, there exist two important measures: (i) the *number of duplicates* of I, denoted as $\delta^{\mathcal{D}}(I)$, that is the number of occurrences of I in $\mathcal{D}$, and (ii) the *support* of I, denoted as $\sigma^{\mathcal{D}}(I)$, that is the sum of the number of duplicates of each itemset J in $\mathcal{D}$ containing I, i.e., $\sigma^{\mathcal{D}}(I) = \sum_{J \in \mathcal{D} \wedge I \subseteq J} \delta^{\mathcal{D}}(J)$. A dataset $\mathcal{D}$ can be represented in a succinct format as a set of pairs $(I, \sigma^{\mathcal{D}}(I))$. Given $\mathcal{I} = \{a, b, c, d\}$, an example of dataset in the succinct, one-hot format is shown in Table 1-a. We say that I is a *frequent* (resp., *infrequent*) itemset in $\mathcal{D}$ if its support is greater than or equal to (resp., less than) a given threshold. A classical data mining task over transaction datasets is to detect the set of the *frequent/infrequent itemsets*, and a rich literature deals with this topic [4, 5, 6] Given the threshold 50, the frequent itemsets for the dataset of Table 1-a are listed in Table 1-b. The perspective of the frequent itemset mining problem has been later inverted as follows: given a set of itemsets together with their frequency constraints the goal is to compute, if any, a transaction dataset satisfying the above constraints. This new problem is called the *inverse frequent itemset mining* problem (IFM) algorithms [7], and has been later investigated also in privacy preserving contexts [8]. Given a set $\mathcal{I}$ of items, the IFM problem consists in finding a dataset $\mathcal{D}$ that satisfies given support

Table 2
Examples of Feasible Datasets with size 170 for an IFM *instance.*

a	b	c	d	$\sigma^{\mathcal{D}}$
1	1	1	0	70
0	1	1	1	10
1	1	0	0	30
0	1	1	0	20
0	0	1	1	40

(a) Dataset $\mathcal{D}_1$

a	b	c	d	$\sigma^{\mathcal{D}}$
1	1	1	0	40
0	1	1	1	40
1	1	0	0	60
0	1	1	0	20
0	0	1	1	10

(b) Dataset $\mathcal{D}_2$

a	b	c	d	$\sigma^{\mathcal{D}}$
1	1	1	1	30
1	1	1	0	30
0	1	1	1	20
1	1	0	0	40
0	1	1	0	50

(c) Dataset $\mathcal{D}_3$

constraints on some itemsets S_i on $\mathcal{I}$ — the set of such itemsets is denoted by S. The support constraints are represented as follows: $\forall S_i \in S : \sigma^i_{min} \leq \sigma^{\mathcal{D}}(S_i) \leq \sigma^i_{max}$, where $\sigma^{\mathcal{D}}(S_i)$ is the sum of all number of duplicates of itemsets in $\mathcal{D}$ containing I. As an example, consider $\mathcal{I} = \{a, b, c, d\}$, $S = \{\{a, b\}, \{b, c\}, \{c, d\}\}$ and the support constraints represented in Table 1-b – in this example minimal and maximal supports coincide. The itemsets $S_1 = \{a, b\}$ and $S_2 = \{b, c\}$ must occur in exactly 100 transactions (possibly as their sub-transactions) whereas the itemset $S_3 = \{c, d\}$ must occur in exactly 50 transactions. It is also required that the dataset size (i.e., the total number of transactions) be 170. The dataset D_1 shown in Table 2-a is feasible as it satisfies all constraints: S_1 is satisfied by the transactions $\{a, b, c\}$ and $\{a, b\}$, S_2 by the transactions $\{a, b, c\}$, $\{b, c, d\}$ and $\{b, c\}$ and S_3 by the transactions $\{b, c, d\}$ and $\{c, d\}$.

Let S' be the set of all itemsets that are neither in S nor subsets of some itemset in S. In the example, S' consists of $\{a, b, c, d\}$, $\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$, $\{b, c, d\}$, $\{a, c\}$, $\{a, d\}$ and $\{b, d\}$. IFM does not enforce any constraint on the itemsets in S' and, therefore, it may happen that $\mathcal{D}$ contains additional (and, in some cases, unsuspected or even undesired) frequent itemsets. In the dataset $\mathcal{D}_1$ of Table 2-a, the itemset $\{a, b, c\}$ is in S' but it turns out to be frequent with a support of 70. To remove the anomaly, Guzzo et al. [9] have proposed an alternative formulation, called IFM_S, that requires that only itemsets in S can be included as transactions in $\mathcal{D}$ and, therefore, no unexpected frequent itemsets may eventually occur. Obviously, the decision complexity of this problem is lower as it is NP-complete. Despite the complexity improvement, the IFM_S formulation has a severe drawback: it is too restrictive in excluding any transaction besides the ones in S as confirmed by the fact that no feasible dataset exists for our running example. To weaken the tight restrictions of IFM_S, Guzzo et al. [10] proposed a new formulation of the problem, called IFM *with infrequency support constraints* (IFM_I for short), which admits transactions in S' to be in a feasible dataset if their supports are below a given threshold σ'. By the anti-monotonicity property, the number of infrequency support constraints can be reduced by applying them only to a subset of S' consisting of its minimal (inclusion-wise) elements. This subset, denoted by $B_{S'}$, is called the *negative border* and coincides with the set of all minimal transversals of the hypergraph $\bar{E} = \{\mathcal{I} \setminus I : I \in S\}$ (as defined in [11]). In the example, $B_{S'} = \{\{a, c\}, \{a, d\}, \{b, d\}\}$ and the dataset $\mathcal{D}_2$ in Table 2-b is a feasible dataset for IFM_I for $\sigma' = 40$. In fact, all infrequency support constraints on $B_{S'}$ are satisfied as the supports of $\{a, c\}$, $\{a, d\}$, $\{b, d\}$ are respectively 40, 0 and 40. Another possibility to enforce infrequency constraints is to fix a duplicate threshold δ' so that an itemset in S' is admitted as transaction

in a feasible dataset if its number of occurrences is at most δ'. This formulation has been given in [12] with the name of IFM *with infrequency duplicate constraints* (IFM_D for short). Observe that duplicate constraints are less restrictive than infrequency constraints in the sense that some itemset I in $B_{S'}$ may happen to be eventually frequent as it may inherit the supports of several itemsets in S' with duplicates below the threshold. For instance, given the threshold $\delta' = 30$, the dataset $\mathcal{D}_3$ in Table 2-*c* is a feasible dataset for IFM_D. However, the supports of $\{a, c\}$, $\{a, d\}$, $\{b, d\}$ are respectively 60, 30 and 50, thus $\{a, c\}$ and $\{b, d\}$ are frequent.

3. Machine Learning-based Generative Models

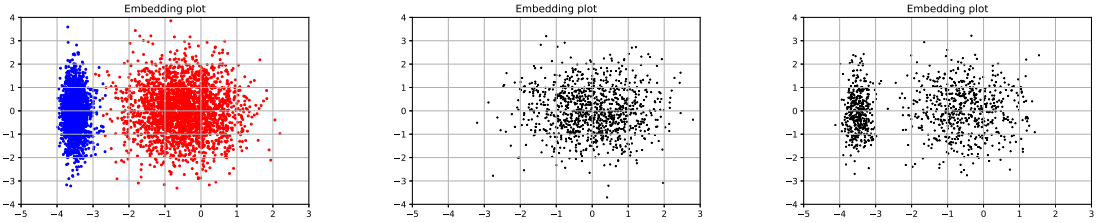
The probabilistic approach to model transactional data (PGM - *probabilistic generative modeling*) assumes that in a database $\mathcal{D}$ the itemsets are modeled as stochastic events: that is, they are sampled from an unknown *true distribution* $\mathbb{P}_r$. The analysis of the statistical distribution of the stochastic events provides insights on the mathematical rules governing the generation process. The problem hence becomes how to obtain a smooth and reliable estimate of $\mathbb{P}_r$.

In general, it is convenient to use a parametric model to estimate $\mathbb{P}_r$ when the constraints on the shape of the distribution are known. By associating each observation $\boldsymbol{x}$ with a probability measure $P(\boldsymbol{x}|\theta) \equiv P_\theta(\boldsymbol{x})$ where θ is the set of the distribution parameters, our problem hence becomes to devise the optimal parameter set θ that guarantees a reliable approximation $\mathbb{P}_\theta \approx \mathbb{P}_r$ that can emulate the sampling process $\boldsymbol{x} \sim \mathbb{P}_r$ in a tractable and reliable way. A clear advantage of a parametric approaches to data generation lies in the insights that it can provide within the data generation process. They allow to detect the factors governing the data, providing a meaningful explanation of complex phenomena. In this work, we are focusing on two state-of-the-art probabilistic approaches: Variational Autoencoders and Generative Adversarial Networks.

A **Variational Autoencoder (VAE)** is an artificial neural architecture that combines traditional autoencoder architectures [13] with the concept of latent variable modeling [14]. Essentially, we can assume the existence of a a K-dimensional latent space $\mathcal{Z}$, that can be the generation engine of the samples in $\mathcal{X}$. The transactions $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\eta\}$ can be modeled through a chain dependency: (*i*) given a distribution P_θ (over the parameter set θ) we can sample $\boldsymbol{z} \in \mathcal{Z}$, and (*ii*) given a $\boldsymbol{z}$ and another distribution P_ϕ (over the parameter set ϕ) we can sample $\boldsymbol{x}$. According to the maximum likelihood principle, optimal values of θ and ϕ can be found by trying to maximizing $P(\mathcal{X})$, but, unfortunately, this optimization is typically an intractable problem that requires the exploitation of heuristics.

Variational inference [15] introduces a proposal normal distribution $Q_\lambda(\boldsymbol{z}|\boldsymbol{x})$ (over the parameter set λ), whose purpose is to approximate the true posterior $P(\boldsymbol{z}|\boldsymbol{x})$. Hence, a VAE is devised by concatenating two neural networks: an "Encoder", that maps an input $\boldsymbol{x}$ into a latent variable $\boldsymbol{z}$, exploiting $Q_\lambda(\boldsymbol{z}|\boldsymbol{x})$, and a "Decoder" that reconstructs $\boldsymbol{x}$ by applying $P_\phi(\boldsymbol{x}|\boldsymbol{z})$ to $\boldsymbol{z}$. The gain function, to be optimized to learn the network parameters λ and ϕ, is obtained by marginalizing the log-likelihood of $P(\mathcal{X})$ over $\boldsymbol{z}$ and applying Jensen's inequality: $\log P(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z}\sim Q}[\log P(\boldsymbol{x}|\boldsymbol{z})] - KL\left[Q(\boldsymbol{z}|\boldsymbol{x})\|P(\boldsymbol{z})\right]$, where KL is the Kullback-Leibler divergence.

The Decoder can be opportunely exploit to solve the discrete data generation problem. In fact,



(a) Real data mapping in $\mathcal{Z}$

(b) Synthetic data generation using MVAE

(c) Synthetic data generation using Clustering-variant of MVAE

Figure 1: Illustration of the data generation process within VAE.

we can sample as many $\boldsymbol{z}$ as we want, feed them to the Decoder and obtain brand new synthetic data that is similar to the real one, if the VAE was properly trained. A simple generation approach, called Multinomial VAE (MVAE), was proposed in [16], where $\boldsymbol{x} \sim Multinomial\left(\pi(\boldsymbol{z})\right)$, with $\pi(\boldsymbol{z}) = softmax\left\{\exp\left[P_\phi(\boldsymbol{z})\right]\right\}$ and $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_K)$. A more sophisticated approach, that can better observe the latent space of $\boldsymbol{z}$ and overcome the strong bias of considering only one standard normal distribution, is implemented by clustering all the $\boldsymbol{z} \sim Q_\lambda(\mathcal{X})$. The generation of a synthetic $\boldsymbol{x}$ starts by choosing a cluster c, by exploiting a multinomial sampling according to the clusters' densities. Then, we can apply the MVAE sampling to pick up $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)$, where $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c$ are the mean and standard deviation within c. A comparison of these two approaches is shown in Figure 1. Unfortunately, the approaches based on maximum likelihood have been shown to suffer from over generalization [17], especially when data from $\mathbb{P}_r$ is limited.

Generative Adversarial Networks (GANs) [18] propose an alternative modeling which departs from the maximum likelihood and instead focuses on an alternative optimization strategy. In order to optimize the weights θ of a neural network, called Generator G, able to learn the probability space $\mathbb{P}_\theta$, Adversarial Networks rely on an auxiliary classifier D, with weights ϕ, trained to discriminate between real and generated data. In practice, optimality can be achieved when $\boldsymbol{x} \sim \mathbb{P}_\theta$ is indistinguishable from $\boldsymbol{x} \sim \mathbb{P}_r$. The training process can be hence devised as a competitive game, with the generator trying to produce realistic samples, starting from random samples in the latent space $\mathcal{Z}$, and the classifier focusing on the detection of generated data, where the objective function is $\min_\theta \max_\phi \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}\left[\log D_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}\left[\log(1 - D_\phi(\boldsymbol{x}))\right]$.

It can be shown [18] that the adoption of this alternate optimization is equivalent to train θ to minimize the Jensen-Shannon divergence, that, differently from the approaches based on maximum likelihood, has the objective of a complete adherence of $\mathbb{P}_r$ and $\mathbb{P}_\theta$. In our context, transactions are discrete vectors, with $\boldsymbol{x} \in \{0, 1\}^n$, thus backpropagation does not directly apply to G and a workaround is needed. The simplest one is to admit a continuous relaxation of the output of the generator. Just like with the variational autoencoder, the output of G can be modeled a multinomial probability (with no direct sampling channel), rather than a binary vector. However, there is a major problem with this: the input of the discriminator would be a softmax distribution from the generated transactions, and a binary vector for the real transactions. As a result, the discriminator could easily tell them apart, with the result that the GAN would get stuck in an equilibrium that is not good for the Generator. Different formulations of the

Table 3
Originary experimental dataset.

Tuple ID	i_1	i_2	i_3	i_4	nd
t_1	1	0	1	1	210
t_2	0	1	0	1	140
t_3	1	1	1	0	110
t_4	1	0	0	0	100
t_5	1	1	0	0	90
t_6	0	0	0	1	80
t_7	0	1	1	0	70
t_8	1	1	0	1	70
t_9	0	1	0	0	70
t_{10}	1	0	1	0	60

itemset	frequent	support
i_1	y	640
i_2	y	550
i_3	y	450
i_4	y	500
i_1, i_2	y	270
i_1, i_3	y	380
i_1, i_4	y	280
i_3, i_4	n	210
i_2, i_3	n	180
i_2, i_4	n	210

adversarial training (based on Wasserstein distance [19], namely Wasserstein GANs or WGANs) can partially mitigate this issue.

4. Comparative Analysis

In this section we compare the two approaches discussed in sections 2 and 3 by analyzing their behavior on a set of controlled experiments. For these, we use a toy dataset, shown in Table 3, comprising 4 items upon which 10 patterns are selected. The dataset used in the experiments is hence built by replicating such patterns with some fixed frequencies. We use the following approaches to generate the synthetic datasets:

- `IFM`: `IFM` formulation with support ≥ 250.
- $\texttt{IFM}_I$: `IFM` formulation with infrequent itemsets' support ≤ 210.
- $\texttt{IFM}_D$: `IFM` formulation with transaction duplicates ≤ 100.
- $\texttt{IFM}_{DI}$: $\texttt{IFM}_D$ merged with $\texttt{IFM}_I$.
- $\texttt{IFM}_{DI-5\%}$: $\texttt{IFM}_I$ formulation imposing that each transaction has a number of duplicates that differ less than 5% from the number of duplicates in the original dataset.
- VAE: Variational Auto Encoder generator.
- VAE$-\{t_4, t_6\}$: VAE without the generation of t_4 and t_6 transactions (as visible in Table 3) by means of sampling with rejection.
- $\texttt{IFM}_{DI-5\%} - \{t_4, t_6\}$: $\texttt{IFM}_{DI-5\%}$ formulation imposing the number of duplicates for t_4 and t_6 to be zero.

The purpose of the experiments is to observe the reconstruction process in both methods and compare the resulting reconstructed datasets. The comparison relies on the transactions ($t_1 \dots t_{10}$ in the table) as well as both simple items and item pairs. We evaluate the faithfulness of the reconstruction in two respects: (1) whether the patterns are reproduced, and (2) whether their frequencies are faithful. A simple metric to measure the reconstruction accuracy is the discrepancy $\mathcal{S}$, computed as $\mathcal{S} = \frac{\sum_i |D_i - O_i|}{\sum_i O_i}$, where, for a pattern i (either a transaction or an item pair), D_i and O_i represent the frequency of the pattern in the reconstructed and original

Table 4
Discrepancy.

Patterns	IFM	IFM_I	IFM_D	IFM_{DI}	$\text{IFM}_{DI-5\%}$	VAE	VAE$-\{t_4, t_6\}$	$\text{IFM}_{DI-5\%} - \{t_4, t_6\}$
Transactions	33.00%	12.00%	48.00%	50.00%	4.80%	7.00%	36.00%	23.10%
Items & item pairs	3.68%	0.82%	1.91%	0.82%	0.11%	3.02%	16.68%	4.71%

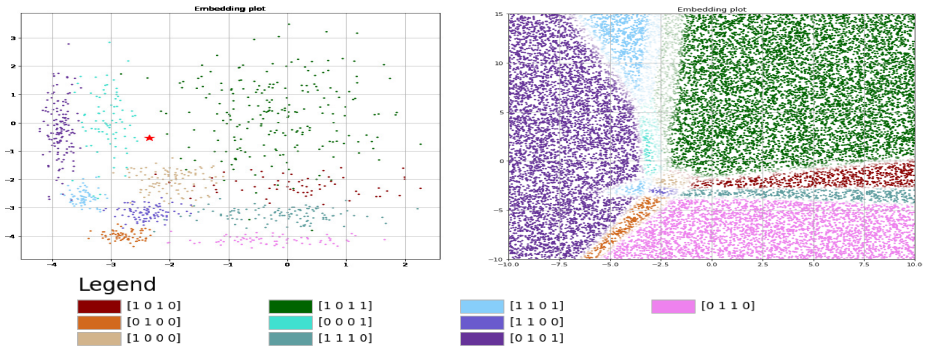


Figure 2: Two-dimensional representation of the latent space

dataset, respectively. Table 4 reports the values of discrepancy, which are further detailed in Figures 3-5.

We first analyze the results of the reconstruction for a VAE-based generative model. Figure 3 shows a comparison with $\text{IFM}_{DI-5\%}$, the IFM-based formulation which provides the best performances. The reconstruction provided by $\text{IFM}_{DI-5\%}$ is extremely faithful, both on the itemsets and the transactions. This because it enforces that the number of duplicates of each transaction differs less than 5% from the number of duplicates in the original dataset. Figure 2 shows the details of how the original patterns are mapped into the latent generative space: the leftmost picture shows the mapping of the original data, and the rightmost shows how the generation results from a larger region of the latent space. The main advantage of the approach based on generative modeling through latent variables is that the latter allows to control the reconstruction process. By acting on the latter we can modify the characteristics of the reconstructed space. For example, we see that transaction t_{11} (the only spurious transaction generated by VAE) is placed in a specific region, denoted by a red star in the figure. Sampling repeatedly from that region would allow us to change the overall distribution of the transactions while still maintaining the itemset distribution. By contrast, the IFM-based approaches are in general successful in maintaining the itemset distributions. However, they tend to produce a higher noise with transactions unless not explicitly constrained by the $\text{IFM}_{DI-5\%}$ formulation (see Figure 4). This noise can in principle be considered an advantage in specific contexts where a differentiation from the original dataset is required (e.g., due to privacy concerns).

In principle, the adoption of IFM allows to implement a reconstruction "by design", by choosing which itemsets to maintain or suppress. As an evidence, we report the cases of $\text{IFM}_{DI-5\%} - \{t_4, t_6\}$ and VAE$-\{t_4, t_6\}$, where transaction t_4 and t_6 are removed from the generation phases. In fact, Figure 5 shows that $\text{IFM}_{DI-5\%} - \{t_4, t_6\}$ keeps the number of

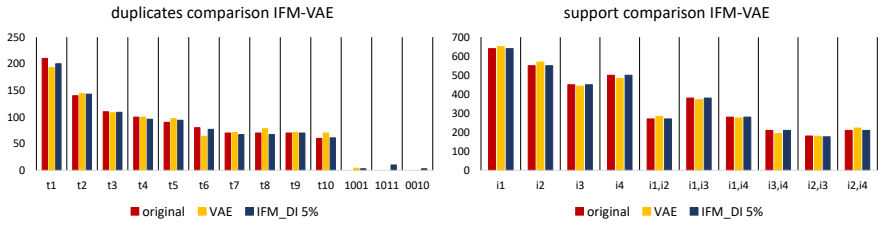


Figure 3: Details of the discrepancies on the number of duplicates (left) and the itemsets support (right) between VAE and IFM-based techniques.

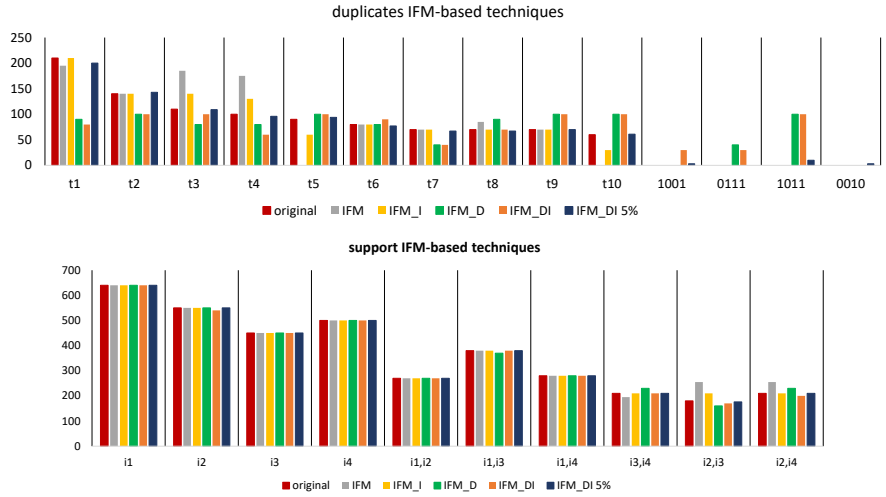


Figure 4: Details of the discrepancies on the number of duplicates (top) and the itemsets support (bottom) between the original dataset and IFM-based techniques.

duplicates and the supports almost similar to the ones of the original dataset, while VAE$-\{t_4, t_6\}$ changes many of them to remove the two transactions. The approach based on generative modeling is in general more efficient. However, constraint-based generation is sensitive to the frequency threshold and a suitable tuning can make these approaches comparable.

To summarize, these experiments support an underlying intuition: Constraint-based generation allows more control on the expected outcome at the expense of a higher computational cost, whereas probabilistic generative models provide more faithful reconstructions but are less controllable. This essentially means that, without any further modeling artifact (that we do not consider here), generative models are prone to fail in providing tailored reconstructions where some patterns can be suppressed and new ones introduced. By contrast, contraint-based generation is more suitable for reconstructions "by design".

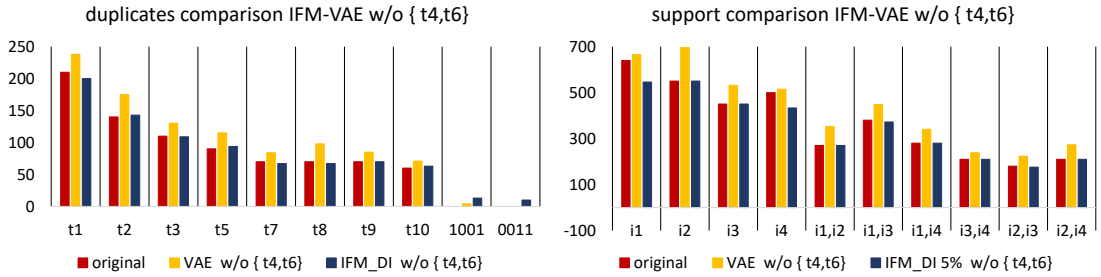


Figure 5: Details of the discrepancies on the number of duplicates (left) and the itemsets support (right) between VAE and IFM-based techniques, without transactions t_4 and t_6.

5. Conclusions

This paper has provided an overview about state-of-the-art approaches for synthetic transactional data generation. A transaction has been modeled as a high-dimension sparse itemset, that can be mapped into a binary vector, defined over the item space. The investigated algorithms are (variants of the) Inverse Frequent Itemset Mining (IFM), and Probabilistic Generative Models (PGMs). According to our analysis, the IFM approaches result to be extremely flexible and understandable; they enable the control of the data generation procedure by the direct identification of the discovery patterns to preserve. However, they proved to have extremely onerous computational costs, making them not feasible in high-dimensional contexts. An opposite conclusion has been obtained by analyzing PGMs: they are extremely fast and accurate, but strongly lacking in control, flexibility and understandability. As future work, an interesting research line is trying to investigate novel methodologies and techniques that are able to take advantage of IFM and PGMs, by combining their strong points and mitigating their weakness. Another promising research line is to apply the combination of the two approaches to NoSQL applications by considering the extension of IFM that has been recently proposed in [12].

References

[1] G. Manco, E. Ritacco, A. Rullo, D. Saccà, E. Serra, Machine learning methods for generating high dimensional discrete datasets, WIREs Data Mining Knowl. Discov. 12 (2022). URL: https://doi.org/10.1002/widm.1450. doi:10.1002/widm.1450.

[2] C. P. Chen, C.-Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on big data, Information Sciences 275 (2014) 314 – 347. doi:https://doi.org/10.1016/j.ins.2014.01.015.

[3] G. Weikum, Where's the data in the big data wave?, 2013. *ACM Sigmod Blog:* http://wp.sigmod.org/?p=786.

[4] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, SIGMOD '93, ACM, New York, NY, USA, 1993, pp. 207–216.

[5] J. Han, H. Cheng, D. Xin, X. Yan, Frequent pattern mining: current status and future directions, Data Mining and Knowledge Discovery 15 (2007) 55–86. doi:10.1007/s10618-006-0059-1.

[6] L. Cagliero, P. Garza, Itemset generalization with cardinality-based constraints, Information Sciences 244 (2013) 161 – 174. doi:https://doi.org/10.1016/j.ins.2013.05.008.

[7] T. Mielikainen, On inverse frequent set mining, in: Proceedings of 2nd Workshop on Privacy Preserving Data Mining, PPDM '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 18–23.

[8] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, ACM, New York, NY, USA, 2000, pp. 439–450. doi:10.1145/342009.335438.

[9] A. Guzzo, D. Saccà, E. Serra, An effective approach to inverse frequent set mining, in: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, ICDM '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 806–811. doi:10.1109/ICDM.2009.123.

[10] A. Guzzo, L. Moccia, D. Saccà, E. Serra, Solving inverse frequent itemset mining with infrequency constraints via large-scale linear programs, ACM Transactions on Knowledge Discovery from Data 7 (2013) 18:1–18:39. doi:10.1145/2541268.2541271.

[11] D. Gunopulos, R. Khardon, H. Mannila, H. Toivonen, Data mining, hypergraph transversals, and machine learning, in: Proceedings of the 16-th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '97, 1997, pp. 209–216. doi:10.1145/263661.263684.

[12] D. Saccà, E. Serra, A. Rullo, Extending inverse frequent itemsets mining to generate realistic datasets: complexity, accuracy and emerging applications, Data Mining and Knowledge Discovery 33 (2019) 1736–1774. doi:10.1007/s10618-019-00643-1.

[13] P. Baldi, Autoencoders, unsupervised learning, and deep architectures, in: In Proceedings of the International Conference on Unsupervised and Transfer Learning workshop (UTLW), volume 27, PMLR, 2011, pp. 37–49.

[14] K. P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.

[15] D. M. Blei, A. Kucukelbir, J. D. McAuliffe, Variational inference: A review for statisticians, Journal of the American Statistical Association 112 (2017) 859–877. doi:10.1080/01621459.2017.1285773.

[16] D. Liang, R. G. Krishnan, M. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, 2018, pp. 689–698.

[17] L. Theis, A. van den Oord, M. Bethge, A note on the evaluation of generative models, in: International Conference on Learning Representations, 2016. doi:10.48550/arXiv.1511.01844.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, volume 27, 2014.

[19] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 214–223.