Received: 11 September 2021 | Revised: 19 December 2021 | Accepted: 20 December 2021

DOI: 10.1002/widm.1450

ADVANCED REVIEW

WIREs DATA MINING AND KNOWLEDGE DISCOVERY WILEY

Machine learning methods for generating high dimensional discrete datasets

Giuseppe Manco[1] | **Ettore Ritacco**[1] | **Antonino Rullo**[2] | **Domenico Saccà**[2] | **Edoardo Serra**[3]

[1]ICAR-CNR, Rende, Calabria, Italy

[2]DIMES Department, University of Calabria, Rende, Calabria, Italy

[3]Computer Science Department, Boise State University, Boise, Idaho, USA

Correspondence
Domenico Saccà, DIMES Department, University of Calabria, Rende, CS 87036, Italy.
Email: domenico.sacca@unical.it

Funding information
European Commission, Grant/Award Number: 952026; Ministero dell'Istruzione, dell'Università e della Ricerca, Grant/Award Number: ARS01_00587; National Science Foundation, Grant/Award Number: 1820685

Edited by: Elisa Bertino, Associate Editor and Witold Pedrycz, Editor in Chief

Abstract

The development of platforms and techniques for emerging Big Data and Machine Learning applications requires the availability of real-life datasets. A possible solution is to synthesize datasets that reflect patterns of real ones using a two-step approach: first, a real dataset X is analyzed to derive relevant patterns Z and, then, to use such patterns for reconstructing a new dataset X' that preserves the main characteristics of X. This survey explores two possible approaches: (1) Constraint-based generation and (2) probabilistic generative modeling. The former is devised using inverse mining (IFM) techniques, and consists of generating a dataset satisfying given support constraints on the itemsets of an input set, that are typically the frequent ones. By contrast, for the latter approach, recent developments in probabilistic generative modeling (PGM) are explored that model the generation as a sampling process from a parametric distribution, typically encoded as neural network. The two approaches are compared by providing an overview of their instantiations for the case of discrete data and discussing their pros and cons.

This article is categorized under:

Fundamental Concepts of Data and Knowledge > Big Data Mining

Technologies > Machine Learning

Algorithmic Development > Structure Discovery

KEYWORDS

constraints-based models, data generation, generative adversarial networks, generative models, inverse frequent itemset mining, synthetic dataset, variational autoencoder

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 (Chen & Zhang, 2014; Michael & Miller, 2013). 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 studies (Weikum, 2013). Companies have data resulting directly from the services they

provide, and industrial labs have access to such data and real-life workloads; however, such datasets are often proprietary and out of reach for academic research.

A main question is hence whether it is possible to benefit of such proprietary data for academic purposes, without forcing companies and private entities to disclose the information coming from them. 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. The idea underlying a synthetic data generation process is hence to synthesize the main properties of real data, either by understanding and modeling the underlying processes or by accurately describing the distribution of their features.

In many situations, synthetically generated datasets are the only choice as comprehensive real data is often hard to come by for many reasons: it might not be available at all, it might not be comprehensive enough to evaluate the system under test, its availability is limited by copyright or intellectual property rights, or its disclosure is forbidden or hindered for privacy reasons.

The latter case includes, for instance, organizations that outsource the testing tasks of their database applications to other organizations without being able to share proprietary data due to privacy considerations. In these circumstances, data masking techniques (i.e., the process of deidentifying or obscuring data) such as encryption, shuffling, substituting, and nulling (Ravikumar et al., 2011), can be adopted in order to outsource a database where original information are hidden, so as sensitive information remain unavailable outside of the origin place. An alternative choice (or complementary, depending on the privacy degree to be pursued) to data masking techniques is to generate synthetic data based on the properties (e.g., distributions, integrity constraints, functional dependencies, domain constraints, etc.) of the original dataset, that is, summarize the features of the input tuples and generate new tuples with same features. In this way, organizations can reveal characteristics of data patterns without actually revealing the actual patterns.

As pointed out at the beginning, an other important motivation for designing synthetic datasets is "availability." For instance, a key need for many research areas studying the behavior of large populations of individuals is the ready availability of realistic synthetic datasets that reproduce relevant attributes and activities of large populations without violating confidentiality of individuals (Wu et al., 2018).

In this article, we focus essentially on the problem of generating high-dimensional discrete data. The latter is common in several scenarios, such as customer modeling, molecular biology, social media analytics, natural language processing. An example is a high-dimensional discrete tuple representing a given customer, where each feature can represent a preferences/purchase or any other attribute characterizing the customer and his behavior. Generating such data is a challenge for several reasons. First, the high dimensionality that naturally arises in the underlying domains makes it difficult to appropriately model concepts such as proximity or relevance. Although several approaches in the current literature exist, based on subspace modeling, the curse of dimensionality is still a huge problem. The situation is further exhacerbated by the discrete nature of the domain, for which the modeling can be affected by the combinatorial explosion of the possible subpatterns, which can represent the regularities in the data. In practice, the combination of both high dimensionality and discrete components may result in a complex structural domain with lots of variety and irregularities and not necessarily smooth.

Throughout the article, we study approaches to data generation, which rely on the idea that high-dimensional discrete data can be mapped into a simpler manifold where smoothness and regularity can be recovered and suitably exploited. The core idea is that each point in the real domain can be mapped into a 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. Thus, learning manifold representations for data and then utilizing it for synthetic generation can be particularly effective.

We consider two main approaches here. The first approach exploits inverse mining techniques. In practice, given a dataset D, the frequent itemsets are discovered first. Next, a new dataset can be generated that satisfies given support constraints on the itemsets of an input set. We call this approach IFM. By contrast, the recent developments in probabilistic generative modeling (PGM) has given rise to approaches which model the generation as a sampling process from a parametric distribution, typically encoded as neural network. variational autoencoders (VAEs) and generative adversarial networks (GANs) are the most prominent examples of this line of research. We will discuss these and study their adaptation for the case of discrete data.

In the end, this article will provide an overview and comparison of such lines of research. The objective is to illustrate similarities and differences, together with strengths and weaknesses.

The rest of the article is organized as follows: in Section 2 we provide a brief overview and related work of the IFM problem, along with a number of techniques used for discovering the set of frequent itemsets, as well as the PGM

techniques VAE and GANs; in Sections 3 and 4 we go into detail of IFM- and machine learning-based generative models, respectively; in Section 5 we provide a comparison of the two approaches by means of experimental results; finally, in Section 6 we conclude the article.

2 | BACKGROUND

Let us first recall some basic definitions on complexity theory—the interested reader may consult (Papadimitriou, 1994) for more details. The class P is the set of decision problems where the "yes"-instances can be decided in polynomial time by a nondeterministic Turing machine. A problem in P is P-complete if it is P-hard, that is, every problem in P can be reduced to it in logarithmic space. The class NP (resp., PSPACE) is the set of decision problems whose "yes"-instances can be decided in polynomial time (resp., space) by a nondeterministic Turing machine. The notions of NP-completeness and PSPACE-completeness are defined in a similar way as for P-completeness. It is well known that, $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE}$. The class PP is the set of decision problems that can be solved by a nondeterministic Turing machine in polynomial time, where the acceptance condition is that a majority (more than half) of computation paths accept. It is known that $\mathrm{NP} \subseteq \mathrm{PP} \subseteq \mathrm{PSPACE}$. Finally, NEXP is the class of decision problems that can be solved by a nondeterministic Turing machine using time $\mathcal{O}\left(2^{p(n)}\right)$ for some polynomial $p(n)$, where n is the size of the input.

Several techniques for generating realistic synthetic datasets have been proposed in literature that mainly follow two distinct approaches, procedural and declarative.

Procedural approaches provide for a programmer to give the features of the original dataset as input to a generation algorithm, where the steps for generating new tuples are specified. These techniques differ with each other mainly in the set of features that underlie the generation task. Bruno and Chaudhuri's Data Generation Language (Bruno & Chaudhuri, 2005) specifies value distributions. Hoag and Thompson's Synthetic Data Description Language (Hoag & Thompson, 2007) has a construct for specifying foreign keys. Houkjær et al. (2006) presented a graph-model based data-generation algorithm which uses cardinalities and value distributions to generate data for large database schemas with complex inter- and intratable relationships. More recently, Li et al. (2018) presented *Touchstone*, a query-aware synthetic data generator, which adopts a random sampling algorithm for instantiating the query parameters. The key technical challenge here is to meet the equality constraints over the join operator, which involve the dependencies among primary and foreign keys from multiple tables.

On the other hand, declarative approaches require the user to specify how new data has to look like rather than how it has to be generated. A rule-based probabilistic approach, based on an extension of Datalog, has been proposed in (Bárány et al., 2017), which is capable of generating data characterized by parametrized classical discrete distributions. Arasu et al. (2011) exploited the declarative characteristic of cardinality constraints for the design of a linear programming based algorithm which instantiates tables that satisfy those constraints. More recently, Gilad et al. (2021) have drawn on (Arasu et al., 2011) for the design of a generation algorithm which allows for the satisfaction of integrity constraints also, enabling the generation of foreign keys for existing database relations. HYDRA (Sanghi et al., 2018) uses a declarative approach that allows for the generation of a database summary that can be used for dynamically generating data for query execution purposes.

A declarative technique that has received particular attention in the context of the generation of synthetic transactional databases is the *Inverse Frequent Itemset Mining* (IFM). IFM is the problem of constructing a binary dataset satisfying given support constraints for frequent itemsets that belong to the original dataset. Given an itemset $I = \{i_1, i_2, ..., i_n\}$, its support is defined as the number of transactions in the original database that contain the items $i_1, i_2, ..., i_n$. An itemset is defined as *frequent* if its support is above a certain user-defined threshold. Frequent itemsets along with their support are considered as the features of the original data to be replicated in the synthetic data.

IFM was initially introduced by Mielikainen (2003) and characterized as a NP-hard problem. Subsequent work have mainly focused in reducing the computational workload by means of approximated solutions. Wu et al. (2005) proposed a heuristic according to which the original itemsets are divided into components that preserve maximum likelihood estimation, and then the iterative proportional fitting method is used on each component. In the approach presented by Guzzo et al. (2009), each equality support constraint is relaxed into a minimum and maximum support constraint. Then, the minimum support is satisfied as long as possible while guaranteeing the maximum support, which improves the tractability of IFM. Wong et al. (2019) introduced the *marginal support* and *global closure* concepts. The marginal support of an itemset is the nonderivable information based on the support of all of its supersets. By determining the marginal support, all itemsets whose supports can be derived by their supersets can be detected. When the sum of all

marginal supports in a set of itemsets is less than or equal to 1 then the set of itemsets is globally closed. Finally, they propose *Itemset2Data*, an algorithm for generating Boolean datasets which is efficient (i.e., it makes the problem of reconstructing data from itemsets tractable), if a globally closed set of itemsets can be derived by the set of frequent itemsets of the original dataset.

Later on, IFM has been reformulated in terms of frequencies by Calders (2004, 2007) with the name FREQSAT {NTRANS}. This version is equivalent to the original formulation and their decision complexity has been better characterized: they are in PSPACE and NP-hard. A simpler version of the frequency formulation, simply called FREQSAT, does not fix the number NTRANS of transaction in a feasible database and the decision problem has been proved to be NP-complete. In the same work Calders introduced a further variant of the problem with the name FREQSAT {NTRANS, NDUP}: all itemsets may occur as transactions in the synthetic data at most a fixed number of times (i.e., NDUP). This problem is in PSPACE and PP-hard.

Besides the NP-hardness, another drawback of the original IFM problem is that the generation task may output itemsets that result to be frequent in the synthetic dataset but that are not in the original one. To overcome this issue, Guzzo et al. (2013) proposed an alternative formulation where itemsets that are not frequent are constrained to be infrequent below a threshold, and solved using a column generation technique as a variation of the simplex method, designed to solve linear programs with a huge number of variables.

We observe that an other popular data mining technique could be used to generate realistic datasets: Association Rules (ARs), that were introduced few decades ago by Agrawal et al. (1993a) for discovering regularities between products in large-scale transactional data and are still attracting further investigations (e.g., multitask AR miner to jointly discover rules by considering multiple tasks [Taşer et al., 2020]). However, to the best of our knowledge, ARs have been rarely used as the basis of the generation task. Indeed, the only work we are aware of that exploits the potential of AR mining for generating synthetic datasets is (Ansari et al., 2018). Here, the authors proposed an ARs-driven algorithm, which generates frequent itemsets ensuring their support value is kept within an acceptable range of their original support used as the basis of the generation task. Nevertheless, ARs may play an important role in the overall process of dataset generation: they can be used to determine whether a synthetic dataset preserves the original patterns. Thus, if the AR patterns discovered in the original data are also present in the synthetically generated data, then the synthetic dataset can be considered to be realistic.

The declarative techniques previously described require a preliminary learning task to be performed for discovering the frequent itemsets in a database, given a user-defined support value. The problem of computing the frequent itemsets of a database was first stated by Agrawal et al. (1993b), and called the *frequent itemset mining* (FIM) problem. FIM was originally developed for market basket analysis, and it is used nowadays for almost any task that requires discovering regularities between nominal variables. Among the best known methods are Apriori (Agrawal et al., 1993b), Eclat (Zaki et al., 1997), FP-Growth (Frequent Pattern Growth) (Han et al., 2000), and LCM (Linear time Closed item set Miner) (Uno et al., 2003). Other frequency-based solutions include the Itemset Generating Model (IGM) (Laxman et al., 2007), and the more recent Latent Dirichlet Allocation-based (LDA) model (Lezcano & Arias, 2019). IGM connects the process of frequent itemsets discovery with the learning of generative models. An IGM generates transactions by embedding a specific pattern in a transaction according to a probability distribution that is peaked at the pattern in question, and is uniform everywhere else. In this manner, each itemset is associated with a specific IGM, so as, given any two itemsets, the IGM associated with the more frequent itemset is the one more likely to generate the database of transactions. Such a connection allows for a generative model-learning interpretation of the frequent itemsets mining process.

LDA (Blei et al., 2003) is a generative model whose main aim is to model a set of documents by discovering the principal topics each document contains, and how the words are distributed for each of these topics. Every document of the corpus is assumed to have its own probability distribution of topics, and every word in a document is created by sampling from a probability distribution determined by topic. In this regard, the authors of (Lezcano & Arias, 2019) proposed to consider each transaction as a document of a corpus, and each item of a transaction as a word in a document. This way a LDA-based model can be fitted to a transactional dataset, and a synthetic version of this dataset using the probability distribution functions thrown by the model can be created.

Unlike the above algorithms, which define the relevance of an itemset based solely on its frequency/support, alternative solutions based on different statistical models to define the relevance of an itemset. Some of the most successful approaches are MTV (Maximally informaTiVe summaries) (Mampaey et al., 2011), KRIMP (Dutch for "to shrink") (Vreeken et al., 2011), and SLIM (Dutch for smart) (Smets & Vreeken, 2012), that are based on the *minimum description length* principle, according to which the most relevant itemsets are those that best compress the dataset following a previously defined encoding scheme. These methods have been shown to lead to much less redundant pattern sets than

FIM. To find relevant itemset, Fowkes and Sutton (2016) used a probabilistic learning approach that directly infers the itemsets that best explain the underlying data. They also proposed a generative model, called *Interesting Itemset Miner*, that is, a probability distribution over the database in the form of a Bayesian network model, based on the relevant itemsets.

Approaches pertaining to a different line of research than the ones described so far, are those based on the use of deep neural networks as generative models. In the context of machine learning, generative modeling is an unsupervised learning task that involves automatically discovering and learning the patterns in the input data so as the model can be used to generate new examples that plausibly could have been drawn from the original dataset.

Two examples of deep learning generative modeling algorithms include the VAEs, and the GANs. VAEs (Kingma & Welling, 2013) are autoencoders with the fundamental property of having a continuous latent space, which allows for an easy random sampling, thus for generating new data (Greco et al., 2020). This is achieved by making its encoder not output just an encoding vector, but rather outputting two vectors, that are a vector of means μ, and a vector of standard deviations σ. They form the parameters of a vector of random variables, with the ith element of μ and σ being the mean and standard deviation of the ith random variable from which we sample to obtain the sampled encoding passed onward to the decoder. This stochastic generation means that, even for the same input, the actual encoding will somewhat vary simply due to randomly sampling from the area of the latent space identified by the circle centered in μ_i with radius σ_i. This makes VAEs *generative* models, as opposed to *discriminative* models such as autoencoders that are designed to classify (i.e., replicate) the input data.

GANs (Goodfellow et al., 2014b) consist of two components, the generator and the discriminator. These components are trained together in a zero-sum game: the generator generates a batch of samples, and these, along with real examples from the domain, are provided to the discriminator and classified as real or fake (generated). When the discriminator is not able to distinguish about half of the generated examples, it means the generator is generating plausible examples. The zero-sum game is because when the discriminator successfully classifies the samples, no change are needed to the model, whereas the generator is penalized with large updates. Alternately, when the generator fools the discriminator, no changes are needed to the model, but the discriminator is penalized and its model parameters are updated.

The main difference between IFM and machine learning-based techniques is that the former is supervised, as it needs the user to find the characteristics of the original dataset (in terms of relevant itemsets) by means of a preliminary discovery task (e.g., FIM), and feed the generation algorithm with these characteristics so as to be replicated on the synthetic data. On the contrary, machine learning-based techniques are totally unsupervised, as they do not need any a priori knowledge to work properly, rather, the discovery of relevant itemsets is performed in parallel with the learning task. In the next sections, we provide a more detailed analysis of the two approaches.

3 | 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* $\mathcal{D}$ over R 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, that is, transactions.

A transactional database $\mathcal{D}$ is very often represented as a bag of itemsets, that is, 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 article, 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 $\mathcal{A}$) such that its i-th position $x_i = 1$ if the i-th item in $\mathcal{A}$ is in I, 0 otherwise. Consequently, $\mathcal{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_\eta\}$, 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, that is, $\sigma^{\mathcal{D}}(I) = \sum_{J \in \mathcal{D} \wedge I \subseteq J} \delta^{\mathcal{D}}(J)$.

A dataset $\mathcal{D}$ can be represented in a succinct format as a set of pairs $\left(I, \sigma^{\mathcal{D}}(I)\right)$. Given $\mathcal{I} = \{a, b, c, d\}$, an example of dataset in the succinct, one-hot format is shown in Table 1a.

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: after the seminal articles in the 1990s (Agrawal et al., 1993b;

TABLE 1 Transactional dataset end frequent itemsets

(a) Dataset $\mathcal{D}$				
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

(b) Frequent itemsets with threshold 50					
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

Gunopulos et al., 1997), additional aspects were studied in the 2000s (Han et al., 2007; Zhong, 2007) and in the last decade (Cagliero & Garza, 2013; Jindal & Malaya, 2016). Given the threshold 50, the frequent itemsets for the dataset of Table 1a are listed in Table 1b.

The perspective of the FIM 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. The new problem, called the *inverse IFM* problem, has been introduced for defining generators for benchmarks of mining algorithms (Mielikainen, 2003), and has been later investigated also in privacy preserving contexts (Agrawal & Srikant, 2000; Wu et al., 2005)).

Given a set $\mathcal{I}$ of items, the IFM problem consists in finding a dataset $\mathcal{D}$ that satisfies given support 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 mentioned in Section 2, IFM has been proved to be in PSPACE and NP-hard (Mielikainen, 2003).

As an example, consider $\mathcal{I} = \{a,b,c,d\}$, $S = \{\{a,b\}, \{b,c\}, \{c,d\}\}$, and the support constraints represented in Table 1b—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 2a 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 2a, 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. (2009) 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. (2013) 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 $\overline{E} = \{\mathcal{I} \setminus I : I \in S\}$ (see [Gunopulos et al., 1997]). In the example, $B_{S'} = \{\{a,c\}, \{a,d\}, \{b,d\}\}$ and the dataset $\mathcal{D}_2$ in Table 2b 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 $\{a,c\}$, $\{a,d\}$, and $\{b,d\}$ are respectively 40, 0, and 40.

TABLE 2 Examples of feasible datasets with size 170 for an IFM instance

a	b	c	d	$\sigma^{\mathcal{D}}$
(a) Dataset $\mathcal{D}\infty$				
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
(b) Dataset $\mathcal{D}\in$				
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
(c) Dataset $\mathcal{D}\ni$				
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

An other 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 (Saccà et al., 2019) 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 2c is a feasible dataset for IFM_D. However, the supports of $\{a,c\}$, $\{a,d\}$, and $\{b,d\}$ are respectively 60, 30, and 50, thus $\{a,c\}$ and $\{b,d\}$ are frequent.

3.1 | General formulation of IFM

In this section, we present the general formulation of IFM given in (Saccà et al., 2019) that takes into account both infrequency and duplicate constraints. Let us first recap and extend the notation introduced at the beginning of this section:

- $\mathcal{I}$ is a set of n items and $\mathcal{U}_{\mathcal{I}}$ is the set of all 2^n itemsets on $\mathcal{I}$;
- S is a set of m nonempty itemsets in $\mathcal{U}_{\mathcal{I}}$ (*frequent itemsets*);
- $S' = \{I \in \mathcal{U}_{\mathcal{I}} | \nexists J \in S : I \subseteq J\}$ (*infrequent itemsets*);
- $B_{S'} = \{I \in S' \mid \nexists I' \in S' : I' \subset I\}$ (*minimal infrequent itemsets*);
- $\Sigma_S = \left\{ \left(I, \sigma^I_{\min}, \sigma^I_{\max}\right) \mid I \in S,\ 0 \le \sigma^I_{\min} \le \sigma^I_{\max} \right\}$ is a set of triples assigning a minimum and maximum support to each itemset in S (*frequent itemset support constraints*)—an unlimited maximum support is denoted by ∞;
- σ' is the support threshold for infrequent itemsets (*infrequency support constraint*;
- δ' is the duplicate threshold for infrequent itemsets (*infrequency duplicate constraint*;
- size is the number of transactions in a feasible dataset.

The *general inverse frequent itemset mining problem* (IFM_G) consists in finding a dataset $\mathcal{D}$ over $\mathcal{I}$ such that:

$$\forall I \in S : \sigma^I_{\min} \leq \sigma^{\mathcal{D}}(I) \leq \sigma^I_{\max} \tag{1}$$

$$\forall I \in B_{S'} : \sigma^{\mathcal{D}}(I) \leq \sigma' \tag{2}$$

$$\forall I \in S' : \delta^{\mathcal{D}}(I) \leq \delta' \tag{3}$$

$$|\mathcal{D}| = \text{size}. \tag{4}$$

IFM_G reduces to: IFM if $\sigma' = \delta' = \infty$, to IFM_I if $\delta' = \infty$, and to IFM_D if $\sigma' = \infty$. It also reduces to IFM_S if the constraint (3) is replaced by $\forall I \notin S : \delta^{\mathcal{D}}(I) = 0$ so that only the itemsets in S are admitted as transactions in a feasible dataset. In the latter case, constraints (2) can be removed as they are automatically enforced by the revised formulation of constraints (3).

It is easily seen that: (i) a feasible dataset for IFM_S is also feasible for both IFM_I and IFM_D, (ii) a feasible dataset for IFM_I or IFM_D is also feasible for IFM, and (iii) if $\sigma' \leq \delta'$, a feasible dataset for IFM_I is also feasible for IFM_D.

The complexity of the decision version of IFM_G has been proved to be NEXP-complete in (Saccà et al., 2019). Next, we summarize the complexity of the decision versions of all formulations of IFM:

- Decision IFM_G and IFM_I are NEXP-complete;
- decision IFM_D is in PSPACE and PP-hard;
- decision IFM is in PSPACE and NP-hard;
- decision IFM_S is NP-complete.

The NEXP complexity of IFM_G and of IFM_I depends on the size of $B_{S'}$ that can be exponential in n and m. To reduce this complexity for IFM_I, Guzzo et al. (2013) have considered a subclass *k-bounded* IFM_I ($k-\text{IFM}_I$ for short) of instances for which the size of $B_{S'}$ is polynomial and can be computed in polynomial time as well. This subclass is composed by all IFM_I instances ℓ for which $|B_{S'}| \leq n - \overline{n} + \overline{n}^{\kappa(\ell)}$, where $\overline{n} = |\cup_{I \in S} I|$, where $\kappa(\ell)$ is an instance parameter that can be computed in polynomial time, $\kappa(\ell) \leq k$ and k is a given rational constant. Experiments conducted in (Guzzo et al., 2013) and the analysis of 12 real large datasets, performed in the same article, show that the parameter $\kappa(\ell)$ is small (around 3) in practice. Saccà et al. (2019) have applied the notion of k-boundedness to IFM_G, thus defining the class *k-bounded* IFM_G ($k-\text{IFM}_G$). As a consequence, the complexity of the two problems reduces as follows: decision $k-\text{IFM}_G$ is in PSPACE and PP-hard and decision $k-\text{IFM}_I$ is in PSPACE and NP-hard.

To get a further and more drastic reduction of the complexity, the integer constraint for the number δ^I of duplicates for a transaction I of a feasible dataset has been relaxed, that is, δ^I may be a rational number. Then, there is a *relaxed* version for each IFM formulation and their decision complexity is:

- relaxed $k-\text{IFM}_G$ and relaxed IFM_D are in PSPACE and PP-hard;
- relaxed $k-\text{IFM}_I$ and relaxed IFM are NP-complete;
- relaxed IFM_S is in P.

3.2 | Modeling general IFM by linear programming

IFM_G has been formulated in (Saccà et al., 2019) as a linear program LP with a very large number of variables and constraints, following the approach used by (Guzzo et al., 2013) to solve IFM_I. A model based on a set of decision variables, an objective function to optimize, and a set of constraints to be satisfied has been also used in (Guns et al., 2011).

As for the formulation given in (Guzzo et al., 2013) for IFM_I, the linear program LP contains a very large number of variables: 2^n—to get an idea, the number of variables for a problem with 250 items is around 1.8×10^75, which is of the order of the number of atoms in the universe, as estimated by scientists. In addition, the linear program formulation of IFM_G also includes a large number of constraints because of the presence of infrequency duplicate constraints. We anticipate that LP is represented in a succinct format (Bertsimas & Tsitsiklis, 1997) that avoids to list all variables and duplicate constraints.

We next present the LP formulation of IFM_G, which gives a unified framework for the resolution of the various specializations of IFM_G as they are achieved by simply removing some of the constraints.

Let us first take any ordering of all nonempty itemsets, say $\{I_1,...,I_{2^n-1}\}$, for example, by selecting any lexicographic order of them. The vector $v=[1,...,2^n-1]$ lists all possible nonempty itemset indices according to this ordering. We denote subsets of such indices as follows:

- the vector $s=[i_1,...,i_m,j_1,...,j_{m'}]$ contains the indices of the itemsets in S and in $B_{S'}$, that is, $S=\{I_{i_1},...,I_{i_m}\}$ and $B_{S'}=\{I_{j_1},...,I_{j_{m'}}\}$ - these indices in s are actually stored;
- the vector s' contains the indices of the itemsets in S'—these indices are not stored but it can be easily checked in $\mathcal{O}(m'))$ time whether an index is in s'.

We next present suitable data structures for representing coefficients and variables of LP:

- l and u are two vectors of m integers such that for each $j,1\le j\le m$, $l_j=\sigma^J_{\min}$ and $u_j=\sigma^J_{\max}$, where $J=I_{s_j}$ is the j-th itemset in S according to the ordering of s - the two vectors are actually stored;
- y is a vector of 2^n-1 non-negative rational variables such that, for each $j<2^n$, y_j stores the number of duplicates for the transaction I_j—a suitable data structure is used to store only the nonzero elements;
- A is a $(m+m')\times(2^n-1)$ matrix such that for each $i,1\le i\le m+m'$, and for each $j\in v$, $a_{ij}=1$ if $I_{s_i}\subseteq I_j$ or $a_{ij}=0$ otherwise—the elements are not stored but they are easily computed in linear time whenever they are needed;
- w is a vector of $2m+1$ non-negative rational number artificial variables, whose values are the costs of violating support constraints: $w_1,...,w_m$ and $w_{m+1},...,w_{2m}$ are associated to respectively lower-bound and upper-bound support constraints on the itemsets in S and w_{2m+1} is the cost of violating the database size constraint—the two vectors are actually stored.

The relaxed version of IFM_G is formulated using the following linear program, whose objective function is minimizing the overall cost of violating the constraints:

$$\text{LP}:\quad \text{minimize}\sum_{i=1}^{2m+1} w_i \tag{5}$$

$$w_i+\sum_{j\in v} a_{ij}y_j\ge l_i\quad 1\le i\le m \tag{6}$$

$$w_{m+i}-\sum_{j\in v} a_{ij}y_j\ge -u_i\quad 1\le i\le m \tag{7}$$

$$-\sum_{j\in v} a_{ij}y_j\ge -\sigma'\quad m+1\le i\le m+m' \tag{8}$$

$$y_j\le\delta'\quad j\in s' \tag{9}$$

$$w_{2m+1}+\sum_{j\in v} y_j\ge \text{size} \tag{10}$$

$$-\sum_{j\in v} y_j\ge -\text{size} \tag{11}$$

$$w_i,x_j\in\mathbb{Q}^+\quad 1\le i\le 2m+1,j\in v \tag{12}$$

Inequalities (6) and (7) enforce the satisfaction of respectively lower bounds and upper bounds of frequency support constraints. The infrequency support constraints and the infrequency duplicate constraints are modeled by the

inequalities (8) and (9), respectively—note that the number of inequalities (9) is exponential because of the size of the array s'. Inequalities (10) and (11) enforce any feasible dataset to have the required size. Finally, the constraints (12) require the variables w_i $(1 \le i \le 2m+1)$ and y_j $(\forall j \in v)$ to be non-negative rational numbers.

The artificial variables in w have the role of absorbing possible violations of the constraints (6), (7), and (10). Instead, the constraints (8), (11), and (9) must be directly satisfied as nonartificial variables are included in them. The optimal solution of *LP* consists of a dataset (described by the nonzero variables y in the optimal solution) with a minimal value for the sum of all artificial variables, that is, the minimization of such values select a dataset with the minimal number of violations of the constraints (6), (7), and (10). Two important remarks are in the order:

- an optimal solution for LP always exists as an initial feasible solution can be easily constructed as follows: $w_{2m+1} = size_1$, $w_i = l_i$, $w_{m+i} = 0$, and $y_j = 0$ $(1 \le i \le m$ and $\forall j \in v)$;
- if the optimal solution of *LP* problem happens to be greater than zero, then the dataset $\mathcal{D}$ returned by the resolution of LP is only an approximate solution, since no feasible dataset actually exists for one (or both) of the following reasons: the frequency support of at least one itemset in S is below the prescribed lower bound or the dataset size is less than the required value.

As mentioned before, LP is represented in a succinct format. In particular, the input is given by: (1) the vector s with size $(m+m')$ storing the indices of the itemsets in S and $B_{S'}$, (2) the two vectors l and u of support bounds (each with size m), (3) the values of σ' and δ', (4) the database size, and (5) two arrays of index tuples storing the indices of the itemsets in S and in $B_{S'}$ respectively, whose overall size is at most $n \times m + n \times m'$. Therefore, the input size is at most $(n+(n+1)(m+m')+2m+3) \cdot \omega$, where ω is the number of bits that are used to represents constants. The coefficients a_{ij} as well the bound constraints (9) are computed as they are needed. We stress that the algorithm for the resolution of LP, described in the next sub-section, implements suitable mechanisms to avoid the whole input expansion.

3.2.1 | Column generation algorithm to solve general IFM

The linear program LP for IFM_G has been solved in (Saccà et al., 2019) by extending the column generation algorithm (see [Gilmore & Gomory, 1961]), which is variant of the simplex method used in operation research literature for solving linear programs with an exponential number of variables (Beheshti & Hejazi, 2015). A column generation algorithm has been first adopted in (Guzzo et al., 2013) to solve IFM_I and has been later extended in (Saccà et al., 2019) to handle inequalities (9) defining infrequency duplicate constraints. We recall that the number of such inequalities is exponential and the novelty of the extension is to solve them by exploiting their succinct representation.

We first present the general scheme of the classical column generation simplex by referring to a generic linear program LP_g with n_c variables and n_r inequalities for which $n_c \gg n_r$, represented in the following standard format:

$$\text{minimize } c^T y, \quad \text{subject to } Ay = b \quad \text{and } y \ge 0,$$

where y is the array of n_c variables, c^T is the transpose of the n_c-array c of costs, b is the array of bounds for the n_r inequalities, and A is a $n_r \times n_c$ matrix. We assume that (1) both the variables and the coefficients are rational numbers, (2) $m_r = \mathcal{O}(\text{pol}(\log n_c))$, where pol is a polynomial in $\log n_c$, and (3) LP_g can be represented in a succinct format with input size polynomial in n_r. Any assignments of values to the n_c variables represents a point, the *feasible region* of LP_g is a convex polytope of the points satisfying $Ay = b$ and a *basic feasible solution*, also called a *basis*, is any vertex of it. The variables taking a value different from zero in a basis are at most n_r (*basic variables*) so that any basis can be represented with size linear in n_r by simply storing the list of basic variables together with their values. From linear programming theory (see (Bertsimas & Tsitsiklis, 1997)), it is known that if there is an optimal solution, then there is an optimal basis as well.

The column generation simplex solves the linear program LP_g without explicitly including all n_c columns but only n_r of them. These columns are selected by solving an auxiliary optimization problem, called the *pricing problem*, which searches for the ones, which reduce the overall current cost by exploiting suitable properties to avoid to perform the evaluation of all possible columns.

If we modify the standard format by adding an upper bound to all n_r variables, the classical column generation simplex cannot be used anymore as the number of inequalities is now exponential. Therefore, an extension of column

generation is needed to take into account the additional bounds without expanding their representation. Such an extension has been devised in (Saccà et al., 2019) to handle the bounds introduced by the inequalities (9). In the following, we briefly describe this extension that has been implemented in (Saccà et al., 2019) for solving the linear program LP modeling IFM_G .

The linear program LP to be solved is called the *master problem* (MP) and consists of $r = 2m + m' + 2$ rows and $c = 2^n + 2m$ columns. In addition, the variables y_j with $j \in \widetilde{s}'$ have an upper bound of δ'. A linear program with only a subset of c' columns, with $c' = r$, is called the *restricted master problem* (RMP). As r is polynomial in the succinct size of the input, RMP does not need a succinct representation. We point out that the number of columns c' managed by RMP can be greater than r, provided that c' is polynomial in r.

The column generation method looks for an optimal basis as within the simplex algorithm. It starts from an initial basis that, as anticipated in the previous sub-section, is easily obtained by setting $w_{2m+1} = size_1$ and $w_i = l_i$ $(1 \le i \le m)$, whereas all other variables are set to zero. The algorithm moves from a current basis to a new one by adding a new basic column with a negative reduced cost (*iteration step*)—the reduced cost of a column can be computed by using the current dual variables. Primal feasibility is maintained during all the iteration steps and the objective function is non-increasing during each step. The selection of a column with a negative reduced cost is delegated to the pricing problem. If no such a column exists, then the algorithm terminates and the current basis is optimal.

The extension of the column generation method introduced by (Saccà et al., 2019) to handle bound constraints (9) follows the approach described in (Luenberger, 2003), which adopts an extended notion of basic solution to avoid to explicitly include the bounds as constraints of the program. An *extended basic solution* is a basic feasible solution where the variables are partitioned into three groups: the set B of the classic basic variables, the set U of the variables equal to the duplicate bound and the set N of those equal to 0. It turns out that only the additional list of the variables in U is to be stored during the iteration steps. A crucial part of the extension is the implementation of the pricing problem that is different from the one presented in (Guzzo et al., 2013) for the resolution of IFM_I because of the bound constraints (9). The problem has been then reformulated using a novel resolution scheme—more details on the formulation and resolution of the pricing problem can be found in (Saccà et al., 2019). We stress that the hardest task is the resolution of the pricing problem, that is in general NP-hard.

As the execution time for the column generation algorithm could be high, Saccà et al. (2019) have fixed a time-limit TL for termination. The algorithm stops for one of the following two conditions: (i) the time-limit has been reached so that the algorithm returns a sub-optimal solution and (ii) the pricing algorithm does not return a column with negative reduced cost and, therefore, the current solution is optimal. The overall algorithm eventually terminates, provided that certain precautions against cycling are taken. We point out that the hardest task is the implementation of procedure PRICE, that is in general NP-hard.

Saccà et al. (2019) have shown that the extended column generation algorithm can handle linear programs with an enormous number of variables and constraints (from 10^{22} to over 10^{240} in their experiments) using a reduced amount of space. These experiments reveal that time does not grow exponentially in practice as it often happens for the classical execution of the simplex algorithm. Indeed, the column generation algorithm has an attractive characteristic that allows it to overcome its theoretical intractability: it makes a bounded use of the space, proportional to the number r of constraints and of the size of the list U.

Finally, as shown in (Saccà et al., 2019), the extended column generation algorithm can be easily specialized to solve all versions of the general problem but IFM_s, in particular:

- IFM_I is modeled by LP if the inequalities (9) are removed. Then, the extension of the column generation algorithm to handle bounded variables is not necessary anymore and, then, the algorithm reduces to the one proposed in (Guzzo et al., 2013). The main simplifications w.r.t. to the extended version are two: the list U of variables with value equal to the duplicate bound is not used and (ii) the pricing problem has a simpler formulation whose implementation can be done in most cases by an efficient polynomial-time heuristic.
- IFM_D is modeled by LP if inequalities (8) are removed. Because of the presence of inequalities (9), the column generation algorithm must preserve the extension introduced to satisfy them, in particular the procedures for constructing and maintaining the list U of bounded variables and for using them in the resolution of the pricing problem. The only difference w.r.t. the general algorithm for IFM_G is efficiency: the negative border $B_{S'}$ no longer has to be computed and, as its size is typical very large also when is not exponential because of k-boundedness, the number of inequalities is drastically reduced.

- IFM is modeled by LP if both inequalities (9) and (8) are removed. Therefore, the simplifications of the two previous versions are combined and the column generation algorithm of (Guzzo et al., 2013) is performed in a very efficient way because of the reduced number of inequalities.

As for the resolution of IFM_S, there is no need to use a column generation algorithm as its linear program formulation has a polynomial number of columns, one for each frequent itemset in S. Then IFM_S can be immediately solved by any classical linear program solver—in a sense, the resolution can be simply obtained with a single call of RMP.

3.3 | Accuracy analysis

Saccà et al. (2019) performed an empirical accuracy comparison of the solutions computed by IFM_S, IFM, IFM_I, IFM_D using two accuracy indices that were computed by comparing the original dataset $\mathcal{D}$ and the dataset $\breve{\mathcal{D}}$ computed by the column generation algorithm for each of the various IFM formulations—let $\breve{S}$ denote the set of frequent itemsets in $\breve{\mathcal{D}}$, $\breve{S}_{\text{F}} = \left\{ I \in S \mid \sigma^{\breve{\mathcal{D}}}(I) > \sigma' \right\}$ (i.e., the frequent itemsets of $\breve{\mathcal{D}}$ that are also frequent in $\mathcal{D}$) and $\breve{S}_{\text{I}} = \left\{ I \in S' \mid \sigma^{\breve{\mathcal{D}}}(I) > \sigma' \right\}$ (i.e., the frequent itemsets of $\breve{\mathcal{D}}$ that are infrequent in $\mathcal{D}$):

- *Frequency Accuracy*, that evaluates how much the set S of the frequent itemsets in D is reflected in $\breve{\mathcal{D}}$:

$$A^{\text{F}}\left(\breve{\mathcal{D}}\right) = \frac{1}{|S|} \sum_{I \in S} \frac{\min\left(\sigma^{\mathcal{D}}(I), \sigma^{\breve{\mathcal{D}}}(I)\right)}{\max\left(\sigma^{\mathcal{D}}(I), \sigma^{\breve{\mathcal{D}}}(I)\right)}$$

This index is a variant of the Jaccard similarity index for bags (i.e., multi-sets) that only evaluates the similarity of all itemsets in S (i.e., the frequent ones in D) in the two datasets but not the similarity of all itemsets in $\breve{S} \backslash S$ (the frequent ones in $\breve{D}$ but not in D).

- *Overall Accuracy*, that evaluates the overall similarity between S and $\breve{S}$ using the classical Jaccard similarity index for sets:

$$A\left(\breve{\mathcal{D}}\right) = \frac{|S \cap \breve{S}|}{|S \cup \breve{S}|} = \frac{|\breve{S}_{\text{F}}|}{|S| + |\breve{S}_{\text{I}}|}$$

As the computation of $\breve{S}_{\text{I}}$ is, in general, very heavy for the datasets computed by IFM or IFM_{D}, the following approximate overall accuracy index was actually used that is obtained from $A\left(\breve{\mathcal{D}}\right)$ by replacing $\breve{S}_{\text{I}}$ with $\breve{S}_{\text{B}} = \left\{ I \in B_{S'} \mid \sigma^{\breve{\mathcal{D}}}(I) > \sigma' \right\}$ (i.e., the frequent itemsets of $\breve{\mathcal{D}}$ that are minimal infrequent ones in $\mathcal{D}$):

$$A^{\text{A}}\left(\breve{\mathcal{D}}\right) = \frac{|\breve{S}_{\text{F}}|}{|S \cup \breve{S}_{\text{B}}|} = \frac{|\breve{S}_{\text{F}}|}{|S| + |\breve{S}_{\text{B}}|}$$

As $A\left(\breve{\mathcal{D}}\right) \leq |A^{\text{A}}\left(\breve{\mathcal{D}}\right)$, the approximate index $A^{\text{A}}\left(\breve{\mathcal{D}}\right)$ gives an "optimistic" estimation of the overall accuracy measure, particularly for IFM and IFM_D.

The experiments conducted in (Saccà et al., 2019) have shown that: (1) IFM_{I} achieves very high overall accuracy but it may get a lower frequency accuracy when the number m' of infrequent itemsets in $B_{S'}$ is so large that the column generation algorithm execution is interrupted by time limit expiration, thus delivering a sub-optimal solution, (2) both IFM and IFM_D have a good frequency accuracy but return a large number of frequent itemsets that were instead supposed to be infrequent, and (3) IFM_S does respect infrequency constraints but often returns inaccurate support values for the itemsets in S mainly because enforcing the transactions to be in S may prevent it from satisfying frequent constraints.

To provide more insights on the accuracy of the various IFM versions, we next report the experiments on two of the three dataset analyzed in (Saccà et al., 2019): the real dataset BMS-WebView-1 with around 60,000 click-stream

transactions from an e-commerce web site and the artificial dataset T10I4D100K with hundred thousands transactions synthesized by the *IBM Almaden* AR data generator. Both datasets can be consulted at the link (KDDCUP, 2000).

A total of nine test instances for each of the two datasets were constructed by using standard itemsets discovery algorithms to extract the set S of all itemsets that are frequent w.r.t. the nine thresholds s: $[0.2\%, 0.3\%, ..., 0.9\%, 1\%]$—the thresholds s are expressed in percentage points w.r.t. the number of itemsets in the dataset. Table 3 reports the number n of items for the two datasets and the number m and m' of itemsets in respectively S and $B_{S'}$ in the nine instances associated to the various thresholds s of the two datasets.

For each dataset and for every S of the nine related test instances, the four IFM problems were formulated by setting: (1) $\sigma^I_{\min} = \sigma^I_{\max} = \sigma^{\mathcal{D}}(I)$ for each $I \in S$, (2) $\sigma' = \sigma_{\max}{}' - 0.04 \cdot |\mathcal{D}|$, where $\sigma_{\max}{}'$ is the greatest integer number that divided for the size of $\mathcal{D}$ is smaller than the threshold t, and (3) $\delta' = 10$, which is significantly smaller than σ'.

The column-generation algorithms were coded in Java using the Ilog cplex 12.0 library for solving the restricted master LP problem RMP. A time limit of 3 h was imposed to each test—we recall that a solution is delivered also after an interruption for time limit, although it will probably be sub-optimal.

For the dataset BMS-WebView-1, all executions were completed within the time limit. Not surprisingly, IFM_S had the best performance, followed by IFM. For this dataset IFM_I performed better than IFM_D because the value of m' was relatively "small." The performances for the dataset T10I4D100K were much worse: the executions were completed within the time limit for all values of s only for IFM_S, whereas IFM and IFM_D succeeded to complete the execution only for $s \geq 0.7\%$ and for $s \geq 0.9\%$, respectively. Due to the high values of m', IFM_I never completed the execution within the time limit.

The frequency accuracy for the nine instances for the two datasets are reported in Figure 1. The accuracy for the dataset BMS-WebView-1 of all IFM formulations but IFM_S is substantially 1 for all nine cases. The reason for the rather low accuracy of IFM_S is that frequency constraints on the itemsets in S result to be so restrictive that they cannot be satisfied without including infrequent transactions. As for the dataset T10I4D100K, the accuracy of IFM_I is not anymore 1 as instead it continues to be for IFM and IFM_D. The lower accuracy of IFM_I accuracy, which still reaches values over 0.8, was caused by the large number of infrequency support constraints that brought to prematurely interrupt the IFM_I column generation algorithm. Also in this case the accuracy of IFM_S is rather poor and decreases with increasing of s.

The overall accuracy values, reported in Figure 2, clearly show that only IFM_I is able to delivery accurate solutions for both datasets. In particular, as for the dataset BMS-WebView-1, accuracy for IFM_I is close to 1 for all nine instances, whereas it is between 0.6 and 0.8 for IFM_S and is close to zero for IFM and IFM_D when *sst* is less than 0.8% and 0.6%, respectively. The result is not surprising for IFM as it confirms that the lack of any type of infrequency constraints may yield very inaccurate solutions. Instead, the low accuracy of IFM_D for $s < 0.6\%$ is rather unexpected. Concerning the dataset T10I4D100K, the results clearly show that only IFM_I is able to delivery accurate solutions, whereas both IFM and IFM_D have accuracy close to zero and IFM_S starts from a value around 0.7 and then drops below 0.5 for $s > 0.4\%$.

In conclusion, IFM_I results to be the most effective method but it has the drawback of a heavy execution time when the number m' of infrequent itemsets in $B_{S'}$ becomes too large. A possible solution is to select a higher threshold for

TABLE 3 Characteristics of the nine test instances for the two datasets

	BMS-WebView-1 $n = 497$		**T10I4D100K** $n = 870$	
s(%)	m	m'	m	m'
0.20	798	36,506	13,255	273,436
0.30	435	25,530	4552	238,332
0.40	286	16,605	2001	197,796
0.50	201	11,498	1073	161,617
0.60	162	8732	772	133,080
0.70	133	6260	603	113,368
0.80	105	4226	494	98,294
0.90	90	3260	421	81,858
1.00	77	2633	385	70,611

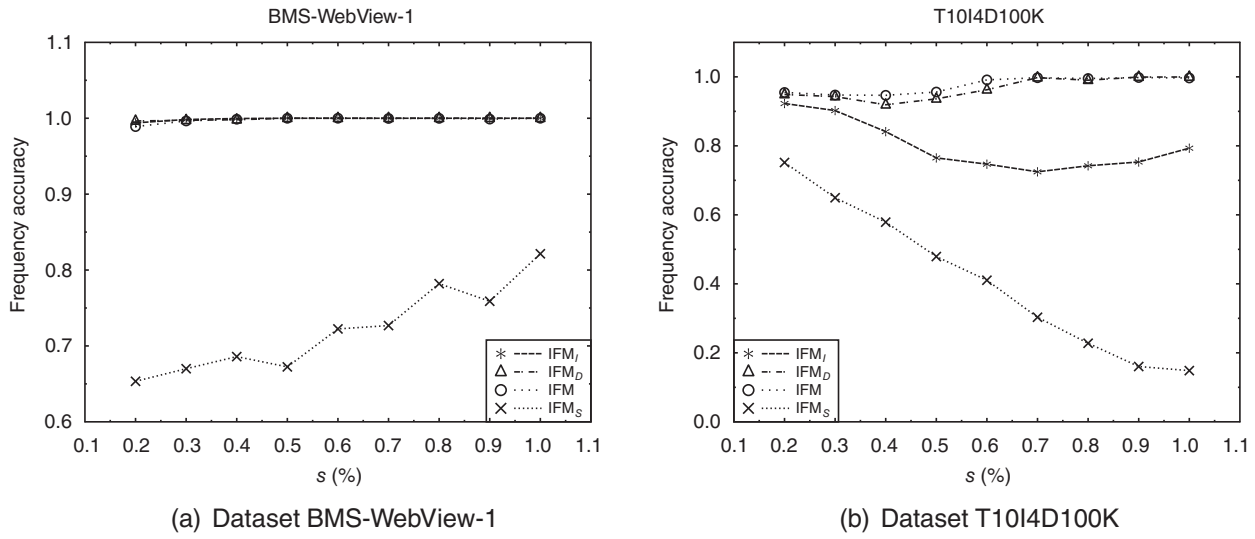


FIGURE 1 Comparison of frequency accuracy indices A^{F}

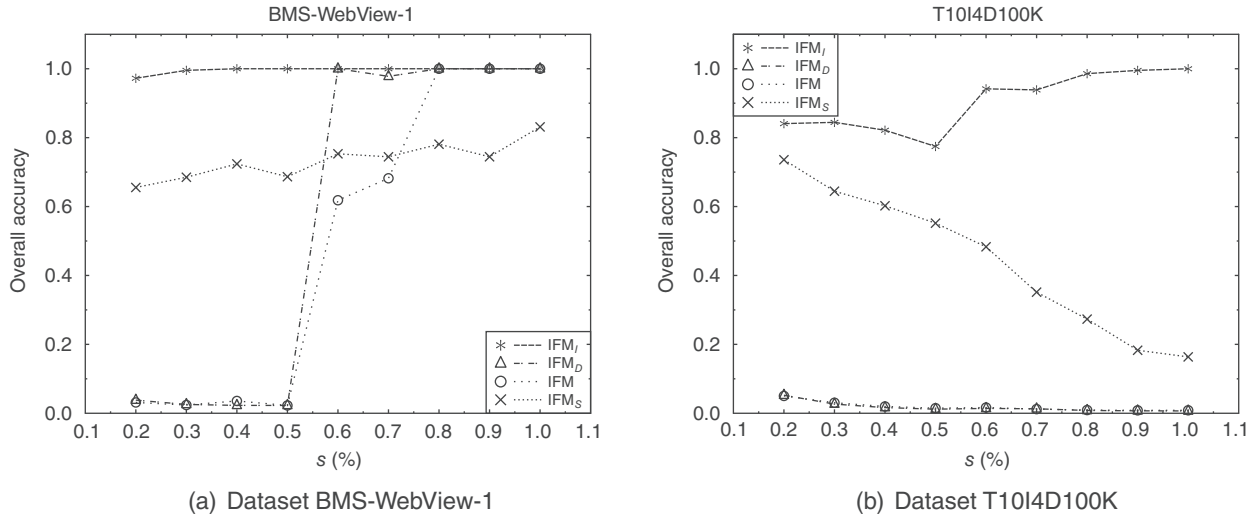


FIGURE 2 Comparison of approximate overall accuracy indices $A^{\mathrm{A}}\left(\widetilde{\mathcal{D}}\right)$

infrequency support constraints to reduce the value of m', compensated by the introduction of infrequency duplicate constraints whose execution cost is lower. In other words, the general formulation of IFM_G is a promising solution provided that the two thresholds σ' and δ' for infrequency constraints are suitably selected through careful tuning.

4 | MACHINE LEARNING-BASED GENERATIVE MODELS

In this section, we provide a probabilistic perspective on the problem of synthetic data generation. Probability theory can be applied to any problem involving uncertainty. In terms of data generation, the question that we would like to answer is: what is the best model to explain some data?

The probabilistic approach to modeling transactional data (PGM) 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:

1. to devise the optimal parameter set θ that guarantees a reliable approximation $\mathbb{P}_\theta \approx \mathbb{P}_r$, given $\mathcal{X}$, or alternatively
2. to infer $P_{\text{true}}(\boldsymbol{x}|\mathcal{X}) \approx \int P_\theta(x)P(\theta|\mathcal{X})\theta$.

The idea is that, by identifying an accurate parameter θ (or, alternatively, being able to devise a tractable inference), we can then emulate the sampling process $\boldsymbol{x} \sim \mathbb{P}_r$ in a tractable and reliable way.

An example parametric model can hence be represented by $\theta = \{\lambda, \Pi\}$ where λ is the parameter of a Poisson distribution and $\Pi \equiv \pi_1, ..., \pi_n$ represent the parameters of a multinomial distribution. The sampling process for an itemset x can hence be summarized as

$$h \sim \text{Poisson}(\lambda)$$

$$\boldsymbol{x} \sim \text{Multi}(h; \Pi)$$

The above notation encodes a two steps process: first we devise the number h of items composing the transaction; next, we sample h items from $\mathcal{A}$ according to a multinomial distribution over Π.

A natural way of estimating parameters of a given probabilistic model is via *Maximum likelihood*. The principle is that, given the underlying stochastic generation process, the observed data $\mathcal{X}$ is the most likely under $\mathbb{P}_r$. As a consequence, any model $\mathbb{P}_\theta$ should maximize the probability of observing $\mathcal{X}$ as well:

$$L(\theta|\mathcal{X}) = P(\boldsymbol{x}_1, \boldsymbol{x}_2, |\theta) = \prod_{i=1}^{\eta} P_\theta(\boldsymbol{x}_i)$$

Maximum likelihood estimation chooses the model parameter θ that maximizes $L(\theta|\mathcal{X})$ or, equivalently, minimizes the negative log-likelihood $\mathscr{L}(\theta|\mathcal{X}) = -\sum_i \log P_\theta(\boldsymbol{x}_i)$. For the example above, the likelihood on $\mathcal{X}$ is

$$\mathscr{L}(\lambda, \Pi|\mathcal{X}) = n\lambda - \log\lambda \sum_{i=1}^{\eta} n_i - \sum_{i=1}^{\eta} \sum_{j=1}^{n} x_{i,j} \log\pi_j$$

where n_i is the number of nonzero elements in $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i,j}$ represents the jth element of $\boldsymbol{x}_i$.

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, thus providing a meaningful explanation of complex phenomena. Besides data generation, probabilistic generative models can be used for compression, denoising, semi-supervised learning, unsupervised feature learning, and other tasks. However, simple parametric models, like the one exemplified above, do not accurately capture complex phenomena characterized by manifold distributions. Mixture models (McLachlan & Peel, 2000) generalize the approach by assuming more complex distributions structured into mixtures. In practice, the basic idea underlying mixtures is that the probability space can be devised into components and, as a consequence, the underlying generative process can be expressed as a composition of two hierarchical steps: first, choose the component responsible for the generation of the data; second, generate the instance according to the parameters of the components. More formally, given a multinomial distribution $\alpha_1, ..., \alpha_K$ representing the prior probabilities, and a parameter set θ_k associated with each component k, the generative process can be devised as:

$$\begin{aligned} k &\sim \text{Multi}(1; \alpha_1, ..., \alpha_k) \\ \boldsymbol{x} &\sim \mathbb{P}_{\theta_k} \end{aligned}$$

The overall probability for an example x can be obtained through the mixture

$$P(\boldsymbol{x}|\theta_1,...,\theta_K,\alpha_1,...,\alpha_K) = \sum_k \alpha_k P(\boldsymbol{x}|\theta_k),$$

which can be exploited to estimate the optimal parameter set $\theta_1,...,\theta_K,\alpha_1,...,\alpha_K$ through maximum likelihood or its approximations (Bishop, 2006).

The current literature has been focusing on deep generative models parameterized by neural networks. The adoption of Stochastic backpropagation and approximate Bayesian inference, deep neural networks and adversarial learning have made these models extremely flexible and accurate in describing the properties of the data. In the following, we shall review two main approaches and will study how they can be adapted to the task of modeling high-dimensional discrete data.

4.1 | Variational autoencoders

A VAE is an artificial neural architecture that combines traditional autoencoder architectures (Baldi, 2012) with the concept of latent variable modeling (Murphy, 2012). Essentially, we can assume the existence of a K-dimensional latent space $\mathcal{Z}$, that can be the generation engine of the samples in $\mathcal{X}$. The transactions $\mathcal{X} = \{\boldsymbol{x}_1,...,\boldsymbol{x}_\eta\}$ 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 x. Hence, the likelihood of $\mathcal{X}$ can be specified as the marginalization over $\mathcal{Z}$:

$$P(\mathcal{X}) = \prod_{i=1}^{\eta} P(x_i) = \prod_{i=1}^{\eta} \int_{z \in \mathcal{Z}} P_\phi(\boldsymbol{x}_i|\boldsymbol{z}) P_\theta(\boldsymbol{z}) \mathrm{d}z$$

The core idea is that even complex dependencies can be explained by normally distributed variables. Hence, the distribution $P_\phi(\boldsymbol{x}_i|\boldsymbol{z})$ can be seen as a function that maps the realizations z into such dependencies. A simple way to implement P_ϕ is by exploiting a neural network, whose input is z and whose parameters are ϕ.

Optimal values of ϕ can be found by trying to maximizing $P(\mathcal{X})$ (or more conveniently its logarithm), according to the maximum likelihood principle. Unfortunately, the maximization of $\log P(\mathcal{X})$ is typically an intractable problem that requires the exploitation of heuristics. Variational inference (Blei et al., 2017) introduces a proposal distribution $Q(z|x)$, whose purpose is to approximate the true posterior $P(\boldsymbol{z}|\boldsymbol{x})$. By considering a single sample $x \in \mathcal{X}$, by Jensen's inequality and the concavity of the logarithm we can in fact observe the following:

$$\begin{aligned}
\log P(\boldsymbol{x}) &= \log \textstyle\int_{z \in \mathcal{Z}} P(\boldsymbol{x}|\boldsymbol{z}) P(\boldsymbol{z}) \mathrm{d}\boldsymbol{z} \\
&= \log \textstyle\int_{\boldsymbol{z} \in \mathcal{Z}} Q(\boldsymbol{z}|\boldsymbol{x}) P(\boldsymbol{x}|\boldsymbol{z}) \dfrac{P(\boldsymbol{z})}{Q(\boldsymbol{z}|\boldsymbol{x})} \mathrm{d}\boldsymbol{z} \\
&\geq \textstyle\int_{z \in \mathcal{Z}} Q(\boldsymbol{z}|\boldsymbol{x}) \log \left\{ P(\boldsymbol{x}|\boldsymbol{z}) \dfrac{P(\boldsymbol{z})}{Q(\boldsymbol{z}|\boldsymbol{x})} \right\} \mathrm{d}z \\
&= E_{z \sim Q}[\log P(\boldsymbol{x}|\boldsymbol{z})] - KL(Q(\boldsymbol{z}|\boldsymbol{x}) \,||\, P(\boldsymbol{z}))
\end{aligned}$$

In short, the inequality

$$\log P(\mathbf{x}) \geq E_{\boldsymbol{z} \sim Q}[\log P(\boldsymbol{x}|\boldsymbol{z})] - KL[Q(\boldsymbol{z}|\boldsymbol{x}) \,||\, P(\boldsymbol{z})] \tag{13}$$

defines a suitable approximation of the likelihood, which depends on the choice of Q. In practice, the right-hand side of the inequality (called evidence lower bound, ELBO) offers a tractable version of the likelihood when Q is accurately chosen.

Based on the above inequality, a VAE can be devised by concatenating two neural networks: an "Encoder," that maps an input x into a latent variable z, exploiting $Q_\lambda(\boldsymbol{z}|\boldsymbol{x})$, and a "Decoder" that reconstructs x by applying $P_\phi(\boldsymbol{x}|\boldsymbol{z})$ to $\boldsymbol{z}$. The loss function of VAE is based on the ELBO devised Equation (13):

$$\mathscr{L}(\lambda,\phi;\mathcal{X}) = \frac{1}{\eta}\sum_{x\in\mathcal{X}} ELBO(\boldsymbol{x},\lambda,\phi) = \frac{1}{\eta}\sum_{x\in\mathcal{X}} E_{z\sim Q_\lambda}\left[\log P_\phi(\boldsymbol{x}|\boldsymbol{z})\right] - KL[Q_\lambda(\boldsymbol{z}|\boldsymbol{x}) \,\|\, P(\boldsymbol{z})]$$

As a consequence, by solving the optimization problem $\text{argmax}_{\lambda,\phi}\mathscr{L}(\lambda,\phi;\mathcal{X})$ we can finally find a suitable approximation of the likelihood function. The issue with this formulation is that the optimization with regards to λ can be problematic. In fact, optimizing the term $E_{z\sim Q_\lambda}\left[\log P_\phi(\boldsymbol{x}|\boldsymbol{z})\right]$ would require sampling from the proposal distribution $Q_\lambda(\boldsymbol{z}|\boldsymbol{x})$ that we want to optimize. However, sampling is a nondeterministic function that depends on λ and is not differentiable. Monte Carlo approximations are prone to high variance and consequently to instability in the learning process, which is hence likely to produce weak solutions.

Kingma and Welling (2014) proposed to overcome this issue by resorting to a *reparametrization trick*: instead of sampling $\boldsymbol{z}$ directly from Q, we can sample an auxiliary noise variable ϵ according to a fixed distribution $P(bs\epsilon)$, and then obtain z by means of a deterministic transformation which depends on both λ and ϵ. In practice, λ would model the core properties of the distribution underlying $Q_\lambda(z|x)$, whereas ϵ models the stochastic nature of the sampling. To illustrate this, we can model Q as a Gaussian distribution $Q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mu_\lambda(\boldsymbol{x}),\Sigma_\lambda(\boldsymbol{x}))$, whose parameters $\mu_\lambda(\boldsymbol{x}),\Sigma_\lambda(\boldsymbol{x})$ are obtained through a neural with input x and λ representing network weights and biases. By exploiting the properties of the Gaussian, the sampling $z\sim\mathcal{N}(\boldsymbol{z}|\mu_\lambda(\boldsymbol{x}),\Sigma_\lambda(\boldsymbol{x}))$, we can obtained by sampling from an auxiliary standard-normal-distributed variable $\epsilon\sim\mathcal{N}(0,I_K)$, and then obtain $\boldsymbol{z}_\lambda(\epsilon,\boldsymbol{x}) = \mu_\lambda(\boldsymbol{x}) + \sigma_\lambda(\boldsymbol{x})\cdot\epsilon$.

The original VAE framework (Kingma & Welling, 2014; Rezende et al., 2014) assumes z to be a standard normal random variable, that implies $\theta = \{0,I_K\}$ and $\boldsymbol{z}\sim\mathcal{N}(0,I_K)$. It turns out that the KL-divergence between two Gaussian distributions has a closed form:

$$KL[\mathcal{N}(\mu_1,\Sigma_1) \,\|\, \mathcal{N}(\mu_2,\Sigma_2)] = \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} - K + tr\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1)\right)$$

By simplifying the $\Sigma_\lambda(\boldsymbol{x})$ parameter into a diagonal matrix, that is, $\Sigma_\lambda(\boldsymbol{x}) = diag(\sigma_{\lambda,1}(\boldsymbol{x}),...,\sigma_{\lambda,K}(\boldsymbol{x}))$, we can finally obtain a fully tractable loss function:

$$\mathscr{L}(\phi,\lambda;\boldsymbol{X}) = \sum_{x\in\mathcal{X}}\left\{\frac{1}{2}\sum_{k=1}^{K}\left(\sigma_{\lambda,k}(\boldsymbol{x}) - 1 - \log\sigma_{\lambda,k}(\boldsymbol{x}) + \mu_{\lambda,k}(\boldsymbol{x})^2\right) - E_{\epsilon\sim\mathcal{N}(0,I)}\left[\log P_\phi(\boldsymbol{x}|\boldsymbol{z}_\lambda(\epsilon,\boldsymbol{x}))\right]\right\} \tag{14}$$

To summarize, by using the loss 14 we can learn the parameters λ and ϕ, and consequently the encoder $Q_\lambda(\boldsymbol{z}|\boldsymbol{x})$ and the decoder $P_\phi(\boldsymbol{x}|\boldsymbol{z})$.

The reference model that we shall investigate for discrete high-dimensional data is the multinomial variational autoencoder proposed in (Liang et al., 2018). This framework devises $\boldsymbol{x}$ according to the generative setting:

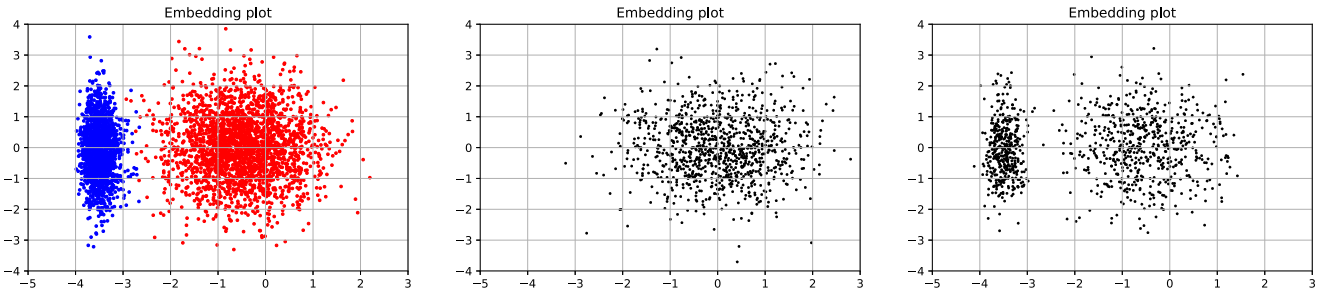
$$\begin{aligned}\boldsymbol{z} &\sim \mathcal{N}(0,I_K)\\ \pi(\boldsymbol{z}) &= \text{softmax}\left\{\exp\left[f_\phi(\boldsymbol{z})\right]\right\}\\ \boldsymbol{x} &\sim \text{Multi}(n;\pi(\boldsymbol{z}))\end{aligned} \tag{15}$$

The Decoder is modeled by:

$$\log P_\phi(\boldsymbol{x}|\boldsymbol{z}) = \sum_{i=1}^{n} x_i \log\pi_i(\boldsymbol{z})$$

thus enabling a complete specification of the overall variational framework.

Generation for new itemsets is accomplished by resorting to the learned function f_ϕ. The overall process can be devised according to the above algorithmic scheme:



(a) Real data mapping in $\mathcal{Z}$ (b) Synthetic data generation using alg. (1) (c) Synthetic data generation using alg. (2)

FIGURE 3 Illustration of the data generation process within VAE

In short, each transaction $\boldsymbol{x}_i$ is characterized by a size (sampled from a Zipf distribution with parameter α) and a latent Gaussian variable $\boldsymbol{z}_i$ upon which the sampling distribution $\pi(\boldsymbol{z}_i)$ relies. The parameter of the Zipf distribution can be estimated from the initial dataset. Below we report typical values for popular benchmark datasets.

This first algorithm has a strong drawback due to the sampling of $\boldsymbol{z}$, in fact, the term $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_K)$ in Equation (15) represents a strong bias in data generation. It assumes that data maps into the latent space according to standard normal distribution, but this assumption is very likely to not hold in many cases. To better fit the data domain, we can modify the algorithm according to the following generation process:

The new procedure requires $\mathcal{X}$, a real dataset that can represent the transaction domain. Initially, it trains a VAE over $\mathcal{X}$ (function *train_VAE*). This operation produces the embeddings $\boldsymbol{Z}$ that can be clustered by an Expectation Maximization algorithm (Dempster et al., 1977) for Gaussian Mixture (function *gmm_EM*). The clustering algorithm provides, for each cluster c, the number m_c of items belonging to its shortest transaction, its probability ξ_c, and its mean $\boldsymbol{\mu}_c$ and standard deviation $\boldsymbol{\sigma}_c$ in the latent space $\mathcal{Z}$. These parameters are the key setting to feed the loop generating synthetic transactions. For each generation: (i) a cluster c is chosen according to a multinomial distribution over the probability distribution $\boldsymbol{\xi}$; (ii) a number N_i of items is determined by the sum of m_c and a Zipf distribution governed by the input parameter α; (iii) a new latent representation $\boldsymbol{z}_i$ is created by sampling from a normal distribution with mean $\boldsymbol{\mu}_c$ and standard deviation $\boldsymbol{\sigma}_c$; and finally, (iv) a brand new synthetic transaction $\boldsymbol{x}_i$ is generated by selecting N_i items through a multinomial sampling over the VAE decoder distribution $\boldsymbol{\pi}(\boldsymbol{z}_i)$.

4.1.1 | Analysis

In order to provide a practical idea of the generative capability of a VAE, we can consider a simple example. Let us supposed to deal with a transaction domain, related to a very large item space, that is characterized by only two main patterns (i.e., maximal frequent itemsets): this means that each transaction is a variation of one of the two patterns by including some other rare items which are not able to generate a third frequent itemset.

We built a dataset $\mathcal{X}$ composed by $\eta = 4000$ transactions that are sampled over a domain of $n = 1000$ items. The first pattern p_1 is the transaction containing only the first 100 items in the domain, while the second one p_2 contains only the last 100 ones. We set the probability of sampling from p_1 to $\sim 60\%$ and to $\sim 40\%$ for p_2 (according to an exponential prior) and each transaction $\boldsymbol{x} \in \mathcal{X}$ was generated as follows:

First, a pattern p_i between p_1 and p_2 has been chosen according to their probabilities, than a number N_i of noisy items ζ_i was sampled through a truncated to 20 exponential function. ζ_i items were collected according to a discrete uniform distribution over the *candidates* list (items not in p_1 or p_2) and were added to p_i to obtain the new sample x_i. Notice that, since $n \gg N_i$, it is very unlikely to generate an unexpected third pattern.

The encoder of a VAE can map the transactions into the latent space $\mathcal{Z}$ splitting them into two clouds of points; a possible 2D graphical representation of this generated data is depicted in Figure 3a, where the red dots belong to p_1, while the blue ones to p_2. By applying the algorithms Algorithms 1 and 2, to generate new synthetic datasets, we

Algorithm 1

Itemset generation using variational autoencoders

Data: Itemset size parameter α;
η' number of transactions to generate.
Result: A new synthetic database $\mathcal{X}' = \{\boldsymbol{x}_1, ...\boldsymbol{x}_{\eta'}\}$ of transactions.
For $i \in \{1, ..., \eta'\}$ **do**
$N_i \sim Zipf(\alpha)$;
$\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_K)$;
$\boldsymbol{x}_i \sim Multi(N_i; \pi(\boldsymbol{z}_i))$;
end

Algorithm 2

Itemset generation via mixtures of Gaussian posteriors

Data: A real dataset $\mathcal{X}$ representing the transaction domain.
Itemset size parameter α;
η' number of transactions to generate.
Result: A new synthetic database $\mathcal{X}' = \{\boldsymbol{x}_1, ...\boldsymbol{x}_{\eta'}\}$ of transactions.
$\boldsymbol{Z} = train\_VAE(\mathcal{X})$;
$\{(m_1, \xi_1, \boldsymbol{\mu}_1, \boldsymbol{\sigma}_1), ..., (m_M, \xi_M, \boldsymbol{\mu}_M, \boldsymbol{\sigma}_M)\} = gmm\_EM(\boldsymbol{Z})$;
for $i \in \{1, ..., \eta'\}$ **do**
$c \sim Multi(1; \xi)$;
$N_i \sim m_c + Zipf(\alpha)$;
$\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)$;
$\boldsymbol{x}_i \sim Multi(N_i; \pi(\boldsymbol{z}_i))$
end

obtained the results shown in Figure 3b and Figure 3c, respectively. The former shows a low capability of representing the data domain, being only able to generate data around the coordinates $(0,0)$ in a single pattern of the latent space, while the latter seems to well fit the domain, by isolating the two input patterns in the same latent regions of the original data.

4.2 | Generative adversarial networks

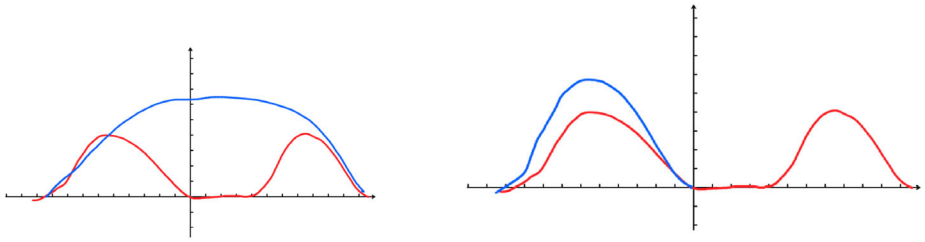
4.2.1 | General framework

We have seen in the above section that a generative approach based solely on VAEs can be problematic when the original data exhibit multiple modalities. The fact is that a VAE aims at approximating the likelihood, thus in a sense it still tries to maximize the likelihood. However, the approaches based on maximum likelihood have been shown to suffer from over-generalization (Theis et al., 2016). Let $\mathbb{P}_r$ denote the true data distribution, and $\mathbb{P}_\theta$ a generative distribution parameterized by θ. The maximum likelihood approach finds the parameter $\widehat{\theta}$ maximizing the empirical evidence, or alternatively minimizing the Kullback–Leibler divergence between the true and the generative distribution:

Algorithm 3

Synthetic transaction generation with two frequent itemsets

Data: The number of items n;
The frequent itemsets $\{p_1, p_2\}$ as binary vectors;
The number of transactions to generate η
Result: A new synthetic database $\mathcal{X} = \{\boldsymbol{x}_1, ...\boldsymbol{x}_\eta\}$ of transactions.
$candidates = \{1, ..., n\} - p_1 - p_2$
for $i \in \{1, ..., \eta\}$ **do**
$p_i \sim Binomial(P(p_1), P(p_2))$
$N_i \sim TruncatedExp(20)$
$\zeta_i \sim DiscreteUniform(floor(N_i); candidates)$
$x_i = p_i + \zeta_i$
end



(a) When the model space θ does not have enough capacity, the generated distribution tends to overgeneralize by allowing samples within regions that are sparse according to $\mathbb{P}_r$.

(b) The adversarial game forces the generated distribution to adhere completely to a portion of the true distribution.

FIGURE 4 Difference between maximum likelihood and adversarial learning. The red lines represent $\mathbb{P}_r$ and the blue lines represent $\mathbb{P}_\theta$

$$
\begin{aligned}
\widehat{\theta} &= \arg\max_\theta \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log P_\theta(\boldsymbol{x})] \\
&= \arg\max_\theta \int_{\boldsymbol{x}} P_{\text{true}}(\boldsymbol{x}) \log P_\theta(\boldsymbol{x}) \frac{P_{\text{true}}(\boldsymbol{x})}{P_{\text{true}}(\boldsymbol{x})} d\boldsymbol{x} \\
&= \arg\max_\theta \int_{\boldsymbol{x}} P_{\text{true}}(\boldsymbol{x}) \log \frac{P_\theta(\boldsymbol{x})}{P_{\text{true}}(\boldsymbol{x})} d\boldsymbol{x} + \int_{\boldsymbol{x}} P_{\text{true}}(\boldsymbol{x}) \log P_{\text{true}}(\boldsymbol{x}) d\boldsymbol{x} \\
&= \arg\max_\theta \int_{\boldsymbol{x}} P_{\text{true}}(\boldsymbol{x}) \log \frac{P_\theta(\boldsymbol{x})}{P_{\text{true}}(\boldsymbol{x})} d\boldsymbol{x} \\
&= \arg\max_\theta - \int_{\boldsymbol{x}} P_{\text{true}}(\boldsymbol{x}) \log \frac{P_{\text{true}}(\boldsymbol{x})}{P_\theta(\boldsymbol{x})} d\boldsymbol{x} \\
&= \arg\min_\theta \int_{\boldsymbol{x}} P_{\text{true}}(\boldsymbol{x}) \log \frac{P_{\text{true}}(\boldsymbol{x})}{P_\theta(\boldsymbol{x})} d\boldsymbol{x} \\
&= \arg\min_\theta KL[\mathbb{P}_r \,\|\, \mathbb{P}_\theta]
\end{aligned}
$$

Maximum likelihood is consistent: it can learn any distribution, provided that it is given infinite data and the model space for θ has sufficient capacity. However, when data from $\mathbb{P}_r$ is limited, the analysis of the above equivalence shows that there can be instances where $P_{\text{true}}(\boldsymbol{x}) < P_\theta(\boldsymbol{x})$. In such case, in fact, the contribution of such instances to the Kullback–Leibler is 0, and still the model $\mathbb{P}_\theta$ does not provide a reliable approximation of $\mathbb{P}_r$. Figure 4a shows this with an example on one-dimensional data.

GANs (Goodfellow et al., 2014a) propose an alternative modeling which departs from the maximum likelihood and instead focuses on an alternative optimization strategy. In order to learn the probability space $\mathbb{P}_\theta$, Adversarial Networks rely on an auxiliary classifier D trained to discriminate between real and generated data. In practice, optimality can be achieved when $\boldsymbol{x} \sim \mathbb{P}_\theta$ 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. By denoting with ϕ the parameters of the classifier D, we can define a value function $V(\phi,\theta)$ as

$$V(\phi,\theta) = \mathbb{E}_{x\sim\mathbb{P}_r}\left[\log D_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}\left[\log\left(1 - D_\phi(\boldsymbol{x})\right)\right]. \tag{16}$$

The adversarial game is set up by solving the optimization problem

$$\min_\theta \max_\phi V(\phi,\theta).$$

In practice, by minimizing ϕ we ensure that D_ϕ is capable of discriminating between real and generated data. By contrast, the maximization of θ aims at "fooling" D_ϕ, by generating data for which the response of the classifier is not consistent.

It can be shown (Goodfellow et al., 2014a) that the adoption of this alternate optimization is equivalent to train θ to minimize the Jensen–Shannon divergence, defined as

$$\mathbb{JS}[\mathbb{P}_r \,\|\, \mathbb{P}_\theta] = \frac{1}{2}(\mathbb{KL}[\mathbb{P}_r \,\|\, \mathbb{P}_\theta] + \mathbb{KL}[\mathbb{P}_\theta \,\|\, \mathbb{P}_r]). \tag{17}$$

Differently from the approaches based on maximum likelihood (which minimize the directional Kullback–Leibler divergence), minimizing the Jensen–Shannon has the objective of a complete adherence of $\mathbb{P}_r$ and $\mathbb{P}_\theta$. Figure 4b shows this difference on the same true distribution illustrated in Figure 4b.

The algorithmic scheme for learning both θ and ϕ is outlined in Figure 4 (Algorithm 4).

Besides the general scheme, the adoption of GANs for data generation poses some questions that we will try to answer in the next subsections.

Algorithm 4

Adversarial learning

for number of training iterations **do**
  for number of discrimination steps **do**
    Update ϕ by ascending

$$\nabla_\phi\left\{\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_r}\left[\log D_\phi(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}\left[\log\left(1 - D_\phi(\boldsymbol{x})\right)\right]\right\}$$

  end
  for number of generation steps **do**
    Update θ by descending

$$\nabla_\theta\left\{\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}\left[\log\left(1 - D_\phi(\boldsymbol{x})\right)\right]\right\}$$

  end
end

4.2.2 | Discrete data

Within the learning process, the first issue is how to compute the gradient of the generator, namely

$$\nabla_\theta \{\mathbb{E}_{x\sim\mathbb{P}_\theta}[\log(1-D_\phi(\boldsymbol{x}))]\}.$$

Once again, the main problem is how to compute the gradient of the expectation. In practice, passing the gradient to the generator is not trivial and some assumptions are needed. Within continuous domains such as image data (i.e., where $\boldsymbol{x}\in\mathbb{R}^{H\times W}$), a typical trick consists in structuring the generator as a neural network $G_\theta:\mathbb{R}^K\mapsto\mathbb{R}^{H\times W}$ which depends on a latent code $\boldsymbol{z}\sim\mathbb{P}_z$ (where $\mathbb{P}_z$ represents a prior such as, e.g., a uniform distribution). As a consequence, the gradient can be rewritten as

$$\mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z}[\nabla_\theta \log(1-D_\phi(G_\theta(\boldsymbol{z})))],$$

and can backpropagate through G.

Transactions, on the other side, represent discrete elements, where $\boldsymbol{x}\in\{0,1\}^n$. The main problem with discrete domains is that the resulting neural network G, in order to map the latent code $\boldsymbol{z}$ into $\{0,1\}^n$, has to introduce discrete random variables at some point in the computation. As a consequence, backpropagation does not directly apply and a workaround is needed.

The simplest workaround is to admit a continuous relaxation of the output of the generator. Just like with the VAE, 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 adversarial training (based on Wasserstein distance [Arjovsky et al., 2017], to be discussed in Section 4.2.3) can partially mitigate this issue. Also, using an approximated embedding layer (Xu et al., 2017) to map both real and generated transactions into dense representations can solve this. In practice, the discriminator D would be trained to provide a response on $r(\boldsymbol{x})$ rather than $\boldsymbol{x}$, where r is a pretrained embedding mapping itemsets into a lower dimensional space according, for example, to their closeness.

A possible solution is to consider the generative model as a stochastic parametrized policy which can be trained by policy gradient (Sutton et al., 2000). The latter naturally avoids the differentiation difficulty for discrete data in a conventional GAN, by means of the equality

$$\nabla_\theta \{\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}[\log(1-D_\phi(\boldsymbol{x}))]\} = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}[\log(1-D_\phi(\boldsymbol{x}))\nabla_\theta \log P_\theta(\boldsymbol{x})].$$

However, the problem with this solution lies in the estimation of the mean $\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}$. Exact estimation requires an exploration of the complete domain space,

$$\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}[\log(1-D_\phi(\boldsymbol{x}))\nabla_\theta \log P_\theta(\boldsymbol{x})] = \sum_{\boldsymbol{x}\in\{0,1\}^n} P_\theta(\boldsymbol{x})\log(1-D_\phi(\boldsymbol{x}))\nabla_\theta \log P_\theta(\boldsymbol{x})$$

which is clearly intractable when n is large. We can introduce Monte Carlo approximation: given a set $\{\boldsymbol{x}_1,...,\boldsymbol{x}_\eta\}$ where $\boldsymbol{x}_i\sim\mathbb{P}_\theta$,

$$\sum_{\boldsymbol{x}\in\{0,1\}^n} P_\theta(\boldsymbol{x})\log(1-D_\phi(\boldsymbol{x}))\nabla_\theta \log P_\theta(\boldsymbol{x}) \approx \frac{1}{\eta}\sum_{i=1}^{\eta}\log(1-D_\phi(\boldsymbol{x}_i))\nabla_\theta \log P_\theta(\boldsymbol{x}_i)$$

Again, the approximation is prone to high variance as it strongly depends on the size η of the sample. As a consequence, the learning process can become extremely unstable and the resulting $\mathbb{P}_\theta$ poorly fits $\mathbb{P}_r$ and variance reduction techniques need to be employed.

Alternative solutions to this issue consider the adaptation of the Gumbel–Max trick (Maddison et al., 2017). The trick essentially consists in a refinement of the continuous relaxation through Softmax/Sigmoid distribution by exploiting the concrete distribution (also referred to as the Gumbel-Softmax/Sigmoid distribution in [Jang et al., 2017]). To illustrate this, consider a categorical distribution with class probabilities $\pi_1, ..., \pi_n$. Maddison et al. (2017) provide a simple way of drawing a sample ξ from such distribution as

$$\xi = \text{one\_hot}\left(\arg\max_i [\log \pi_i + g_i]\right) \tag{18}$$

where $g_i = -\log(-\log u_i)$ (with $u_i \sim \text{Uniform}(0,1)$) are samples drawn from the Gumbel distribution.[1] In fact, it can be shown that, with the above construction, $\xi \sim \text{Cat}(\pi_1, ..., \pi_n)$. This simple trick allows to introduce a reparametrization in the style of (Kingma & Welling, 2014), which allows a control of the variance in the sampling procedure. In practice, when the logits of the discrete distribution depend on a parameter set θ (i.e., $\text{Cat}(\pi_1, ..., \pi_n) \equiv \text{Cat}(\pi_1, ..., \pi_n|\theta)$), the reparametrization trick induces the equivalence

$$\nabla_\theta \mathbb{E}_{\xi \sim \text{Cat}(\pi_1,...,\pi_n|\theta)}[f(\xi)] = \mathbb{E}_{u_i \sim \text{Uniform}(0,1)\ i=1,...n}[f'(\xi) \cdot \nabla_\theta \xi]$$

and the problem hence becomes the estimation of the gradient of Equation (18). The latter is still indefinite because of the nondifferentiable argmax operator, but it can be approximated by the Gumbel-Softmax distribution $\widetilde{\xi}$, defined as

$$\widetilde{\xi}_k = \frac{\exp((\log \pi_k + g_k)/\tau)}{\sum_{j=1}^{M} \exp\left(\left(\log \pi_j + g_j\right)/\tau\right)}$$

where τ is a temperature parameter. The resulting distribution is smooth for $\tau > 0$ and has the property that $\lim_{\tau \to 0} \widetilde{\xi} = \xi$. We can hence devise a Straight-Through (ST) Gumbel approximation $\nabla_\theta \xi \approx \nabla_\theta \widetilde{\xi}$, with the further trick to annihilate the temperature τ during the training process to make the trade-off between smooth samples and small variance.

The concrete distribution allows to sample single items from a categorical distribution. The generation of a transaction $\boldsymbol{x} \in \{0,1\}^n$ requires additional refinements, since the objective is to generate an itemset instead.

- We can decompose the problem in two steps: First, identify the number k of items to generate. Then, given k, a subset of k items can be obtained by repeatedly sampling from the Gumbel-Softmax. Xie and Ermon (2019), propose an extension to the Gumbel-max trick which perturbs the log-probabilities of a categorical distribution with Gumbel noise and takes the top-k elements to produce samples without replacement. The extension introduces on a differentiable approximation of a top-k operator, in the same vein as the Gumbel-Softmax proposes a differentiable approximation to argmax. The intuition is to emulate a repeated sampling with the guarantee that items previously drawn are not resampled: By defining the sequence $\forall i \in \{1, ..., n\}$

$$\alpha_i^{(1)} = \log \pi_i + g_i$$

$$p_i^{(t)} = \frac{\exp\left(\alpha_i^{(t)}/\tau\right)}{\sum_{j=1}^{n} \exp\left(\alpha_j^{(t)}/\tau\right)}$$

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + \log\left(1 - p_i^{(t)}\right),$$

where the superscript (t) indicates the iteration step in the loop for $t \in \{1,...,k\}$, we can finally obtain an approximation $\widetilde{\boldsymbol{x}} = \sum_{t=1}^{k} \boldsymbol{p}^{(t)}$, with $\boldsymbol{p}^{(t)} = \left\{p_1^{(t)},...,p_n^{(t)}\right\}$. Again, it can be shown that, when $\tau \to 0$, each $\boldsymbol{p}^{(t)}$ converges to a one-hot vector and consequently $\widetilde{\boldsymbol{x}}$ to an itemset.

- Alternatively, $\boldsymbol{x}$ can be considered as a the result of sampling from a joint distribution of n binary random variables $\boldsymbol{x}_i \sim \text{Bernoulli}(\pi_i)$, for which a simple adaptation of the Gumbel-Max trick (Maddison et al., 2017) yields

$$\boldsymbol{x}_i = \begin{cases} 1 & \text{if } \log u \ -\log\,(1-u) \ + \ \log\,\pi \ - \ \log\,(1-\pi) > 0 \\ 0 & \text{otherwise} \end{cases}$$

with $u \sim \text{Uniform}(0,1)$. Again, by defining $\alpha_i = \log\pi_i - \log(1-\pi_i)$, we can exploit the Sigmoid relaxation

$$\widetilde{\boldsymbol{x}}_i = \left(1 + \exp\left\{\frac{\log u - \log(1-u) + \alpha_i}{\tau}\right\}\right)^{-1}$$

from which the approximation x can be devised.

4.2.3 | Extensions

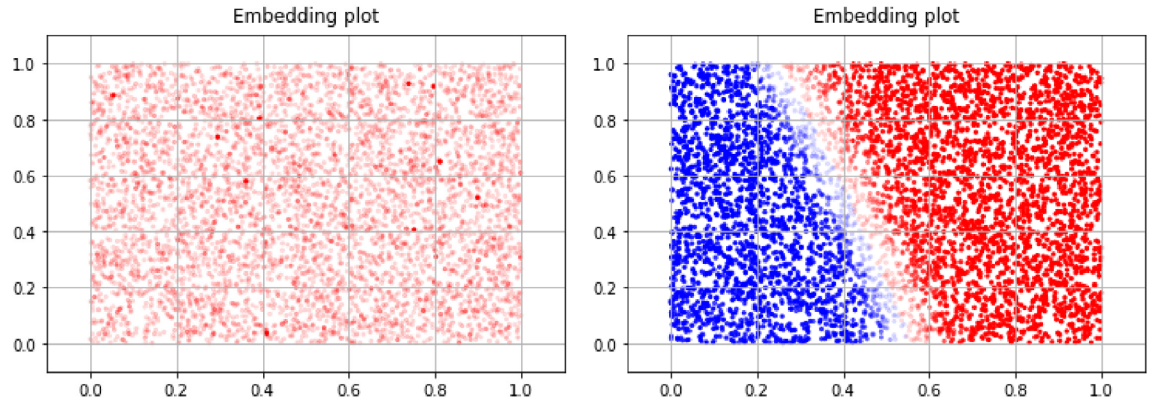
Despite its elegance in modeling, there are a number of issues with GANs that the recent literature is still investigating. First of all, the training process with GANs can be extremely unstable. Unlike other neural network architectures, a GAN does not have a fixed optimization minimum during training. Rather, the optimization minimum changes dynamically during the training procedure. The result is that the overall optimization does not converge to a minimum but to a potentially unstable equilibrium. Regularization (Roth et al., 2017) and improved training through noise (Arjovsky & Bottou, 2017) partially mitigate these issues.

However, one of the sources of instability is given by the difficulty in training a good generator, when the underlying distributions are supported by low-dimensional manifolds. In situations where $\mathbb{P}_r$ and $\mathbb{P}_\theta$ are perfectly separable, the auxiliary discriminator provides little feedback to the generator (which exhibits a vanishing gradient), which consequently does not learn to generate realistic data. The problem is to effectively measure the distance between $\mathbb{P}_r$ and $\mathbb{P}_\theta$, rather than their separability, and to use this distance as a feedback to the generator. Wasserstein GAN (Arjovsky et al., 2017) (WGAN) is an extension to the GAN model that adopts such a change. Within a WGAN, the discriminator, does not model the probability of a generated $\boldsymbol{x}$ being "real": Rather, it quantifies the "realness" of $\boldsymbol{x}$. This change complies theoretically with a different optimization objective: instead of minimizing the JS divergence of Equation (17), a WGAN minimizes the Earth Mover (or Wasserstein) distance between $\mathbb{P}_r$ and $\mathbb{P}_\theta$, through the objective

$$\min_{\theta} \max_{\phi:\|D_\phi\|_L \le 1} \mathbb{E}_{x\sim\mathbb{P}_r}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\theta}[D(\boldsymbol{x})]$$

Here, the constraint $\|D_\phi\|_L \le 1$ denotes that D_ϕ is required to be a 1-Lipschitz function. The constraint on D_ϕ can be enforced by clipping the weights (Arjovsky et al., 2017), or alternatively stabilizing the gradient (Gulrajani et al., 2017) during the learning process. The adaptation with the Wasserstein criterion allows to train a D_ϕ that approximates the real distance between $\mathbb{P}_r$ and $\mathbb{P}_\theta$. As a consequence, there is no need to balance the capacity of G_θ and D_ϕ and the overall learning process is more stable, with a resulting a more robust generator.

Another advantage of WGAN is the capability to deal with mode collapse, which occurs when the adversarial game gets stuck in a local minimum where generator identifies a subspace that the discriminator has difficulty in detecting.



(a) GAN generation.

(b) WGAN generation. The region with a high degree of transparency corresponds to the hyper-plane of separation between the two patterns.

FIGURE 5 Illustration of the data generation process within GAN and WGAN models

TABLE 4 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

This situation is illustrated in Figure 4b, where only one modality is identified, as it represents a viable solution to the adversarial game. The adoption of the EM criterion can better model the difference between the two distributions, thus alleviating the mode collapse.

TABLE 5 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 and item pairs	3.68%	0.82%	1.91%	0.82%	0.11%	3.02%	16.68%	4.71%

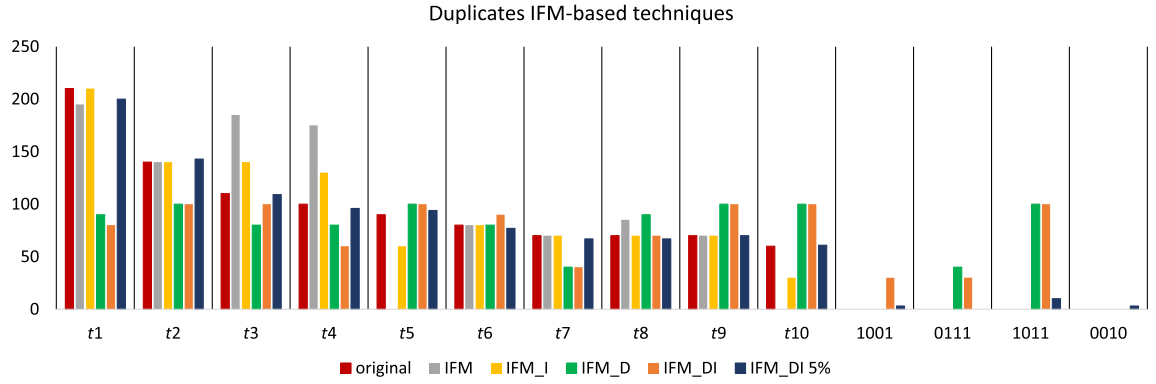


FIGURE 6 Details of the discrepancies on the number of duplicates between the original dataset and IFM-based techniques

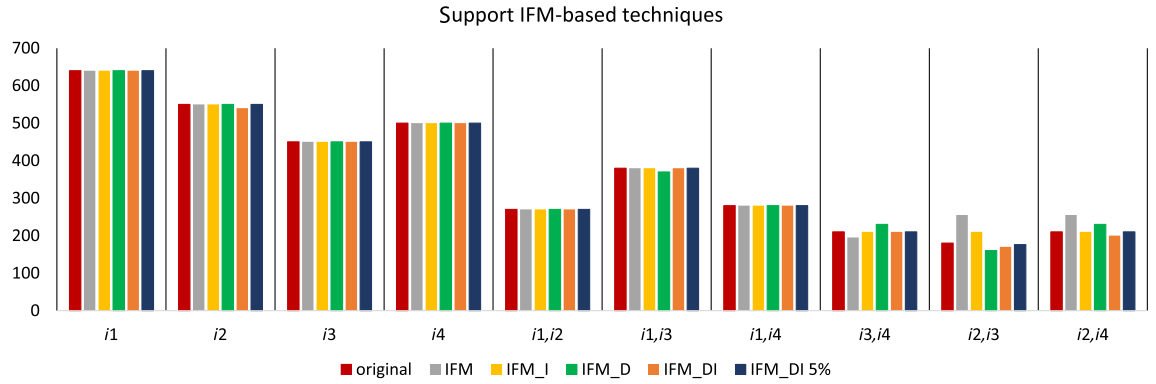


FIGURE 7 Details of the discrepancies on the itemsets support between the original dataset and IFM-based techniques

4.2.4 | Analysis

To perform an empirical comparative analysis between GAN and WGAN we used the same dataset $\mathcal{X}$ described in Section 4.1.1 generated by Algorithm 3. We trained a GAN and a WGAN model, based both on the Bernoulli's approach. This time, we have no information or assumption about the prior distribution of the latent variable Z, so we decided to explore the whole domain $\mathcal{Z}$. Both the models were trained by feeding them latent vectors z of size 2, whose elements are randomly sampled from a *Uniform* distribution and, then, were exploited to generate 20,000 data points that were clustered according to their distance with respect to the two patterns. The results of this kind of analysis are shown in Figure 5:

- Figure 5a shows that the GAN model collapsed on the first patter, associating all the latent point to it (red dots);
- Figure 5b highlight the capability of the WGAN model to mitigate the mode collapsing issues, dividing the latent space into two regions: the red one, corresponding to p_1, covers the $\sim 60\%$ of the space, while the blue one, corresponding to p_2, covers the $\sim 40\%$ of it. As it can be noticed, the coverage reflects the prior probabilities of the patterns.

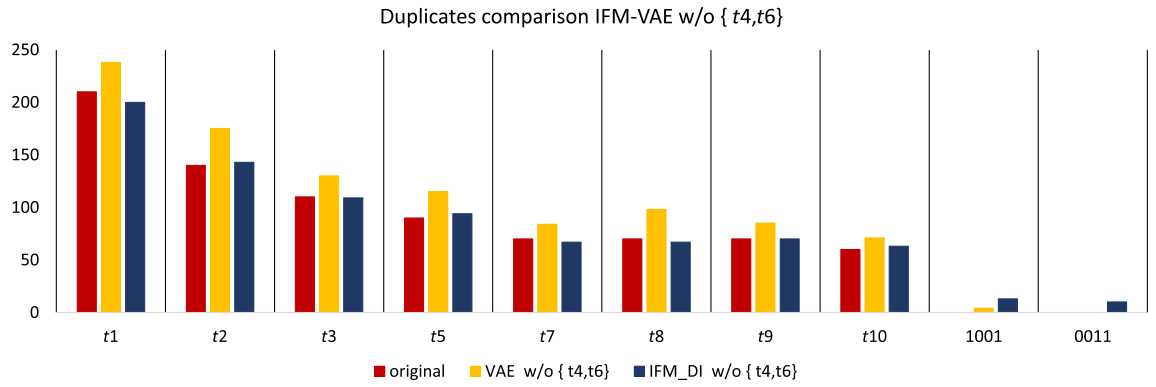


FIGURE 8 Details of the discrepancies on the number of duplicates between VAE and IFM-based techniques, without transactions t_4 and t_6

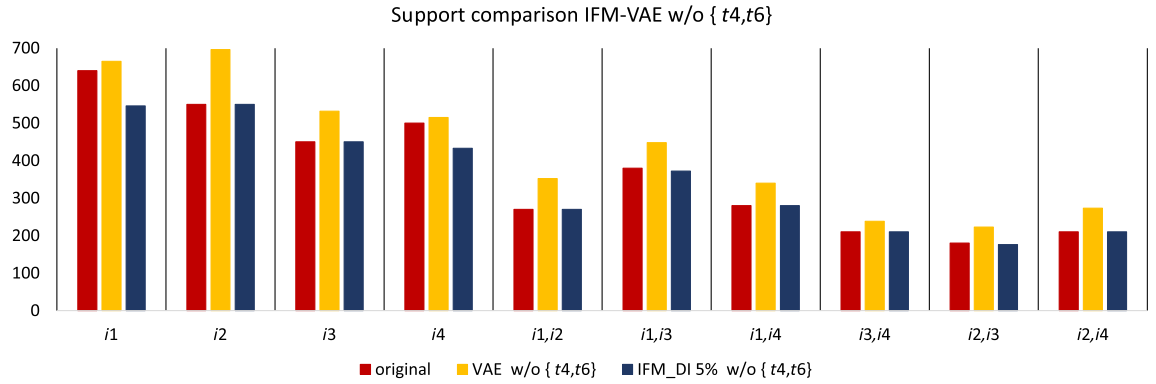


FIGURE 9 Details of the discrepancies on the itemsets support between VAE and IFM-based techniques, without transactions t_4 and t_6

5 | COMPARATIVE ANALYSIS

In this section we compare the two approaches discussed in Sections 3 and 4 by analyzing their behavior on a set of controlled experiments. For these, we use a toy dataset, shown in Table 4, 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 constraints on the frequent itemsets. In our analysis the frequent itemsets are all the itemsets with a support greater or equal of 250.
- IFM_I: IFM formulation imposing that the supports of all the infrequent itemsets to be lower or equal than a fixed threshold. In our analysis, the threshold is 210.
- IFM_D: IFM formulation imposing that the number of duplicates for all the transactions to be lower or equal than a fixed threshold. In our analysis, the threshold is 100.
- IFM_{DI}: IFM_D merged with IFM_I.
- $\text{IFM}_{DI-5\%}$: 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 (see Table 4) by means of sampling with rejection.
- $\text{IFM}_{DI-5\%}$–$\{t_4, t_6\}$: $\text{IFM}_{DI-5\%}$ formulation imposing the number of duplicates for t_4 and t_6 to be zero.

Table 4 shows the details of the dataset and the patterns exploited. 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 ($t_1 \ldots t_{10}$ in the table) as well as both simple items and item pairs. We evaluate the faithfulness of the

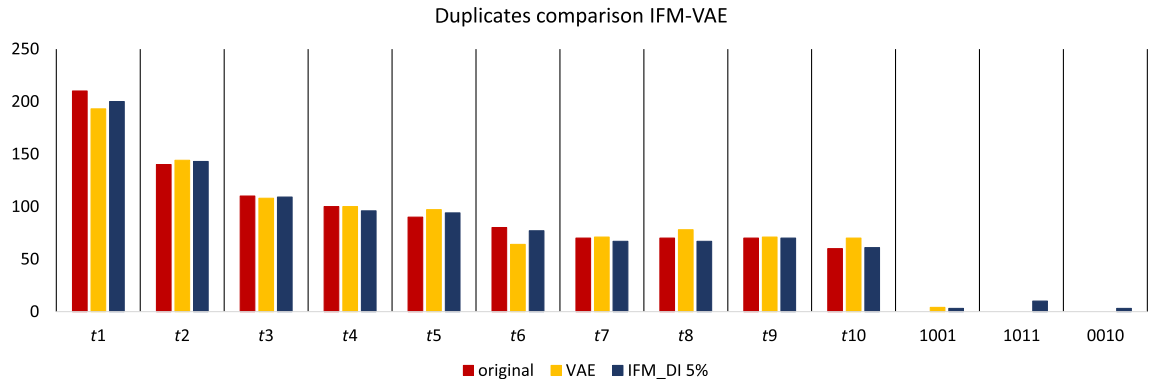


FIGURE 10 Details of the discrepancies on the number of duplicates between VAE and IFM-based techniques

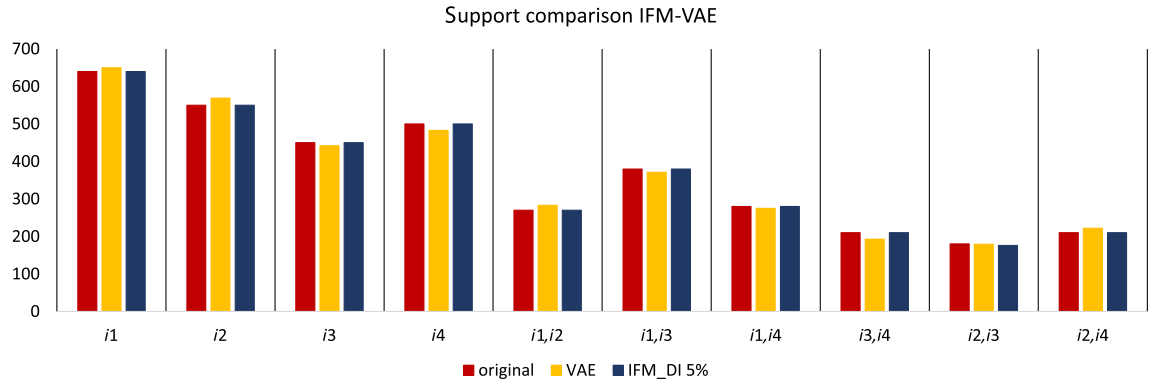


FIGURE 11 Details of the discrepancies on the itemsets support between VAE and IFM-based techniques

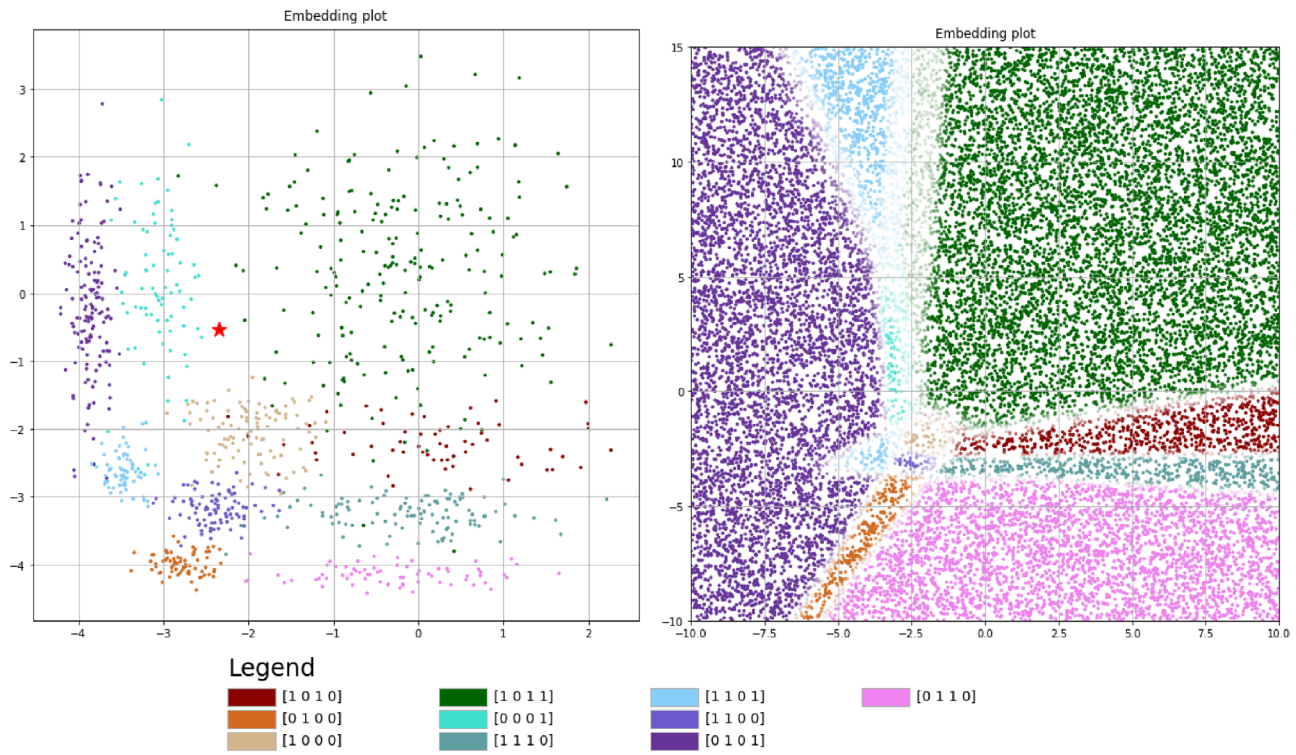


FIGURE 12 Two-dimensional representation of the latent space

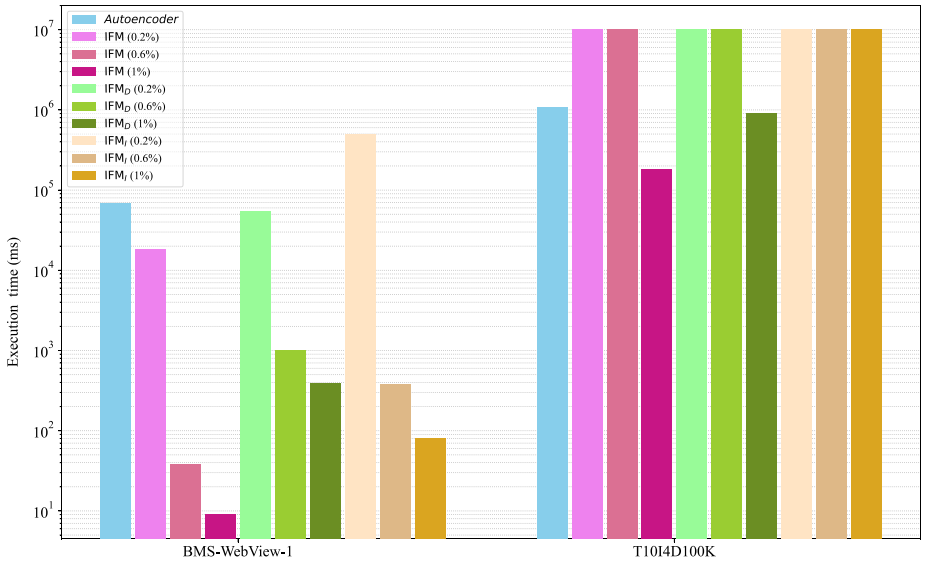


FIGURE 13 Execution times of autoencoder, IFM, IFM$_D$, and IFM$_I$ for the datasets BMS-WebView-1 and T10I4D100K

reconstruction in two respects: (1) whether the patterns are reproduced and (2) whether their frequencies are faithful. The empirical comparison we propose is meant to better characterize both groups of methods, with their advantages and limits. We deliberately choose to use a toy dataset, where the number of patterns is low and the same patterns are simple. This allows us to carefully analyze the results of the generation and compare the methods on them.

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 dataset, respectively. Table 5 reports the values of discrepancy, which are further detailed in Figures 6–9.

We first analyze the results of the reconstruction for a VAE-based generative model by means of Algorithm 2. Figures 10 and 11 show a comparison with IFM$_{DI-5\%}$, the IFM-based formulation which provides the best performances. The reconstruction provided by IFM$_{DI-5\%}$ is extremely faithful, both on the itemsets and the transactions. This because this formulation enforces that the number of duplicates of each transaction differs less than 5% from the number of duplicates in the original dataset. Figure 12 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 IFM$_{DI-5\%}$ formulation (see Figures 6 and 7). 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 IFM$_{DI-5\%} - \{t_4, t_6\}$ and VAE$ - \{t_4, t_6\}$, where transaction t_4 and t_6 are removed from the generation phases. In fact, Figures 8 and 9 shows that IFM$_{DI-5\%} - \{t_4, t_6\}$ keeps the

number of 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.

We extend the comparison also to by analyzing the running time of the proposed methods. To do so, we consider the datasets in Figures 1 and 2 and reports the results of the execution times in Figure 13. 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, constraint-based generation is more suitable for reconstructions "by design."

6 | CONCLUSIONS

This article 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 goal of the generation is to build synthetic transactions with a high degree of realism, by: (i) keeping all the statistical properties of a reference domain, (ii) avoiding to introduce inexistent or unexpected patterns, and (iii) preventing any data leakage of confidential information. In this perspective, the biggest challenges to tackle are essentially the scalability of the discovery of the patterns, due to their exponential explosion, the exploration and understanding of the complex manifold that may characterize the domain.

The algorithms, investigated in this work, are actually machine learning models, which need a training phase for learning the underlying patterns in a real dataset extracted from the domain, before being able to infer new data. The studied approaches can be divided into two families:

- Inverse frequent itemset mining (IFM), whose objectives are to: (i) find out the frequent itemsets (of any size) in the training set, (ii) exploit them as constraints for the new data generation, and (iii) prune unexpected patterns;
- Probabilistic generative models (PGMs), which map the training data patterns into suitable data generative functions from which sampling the synthetic data.

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. On the other hand, they proved to have extremely onerous computational costs, to the point of not being 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.

We argue that the need for methods for synthetic data generation is crucial when real-life datasets are scarce or incomplete (due essentially to privacy concerns or high costs of data collection), or even inaccurate (due to inconsistencies in the data collection process). This is an extremely relevant topic in several areas, including, but not limited to, financial services, healthcare, manufacturing, security, marketing and social media. We would like to stress that in all these scenarios the aforementioned issues are likely to occur and, since the data can be deemed "transactional" (i.e., organized in transactions representing events or properties), the approaches that we survey here can be applied. For example, when transactions represent patients' health records, synthetic data generation can be exploited to generate data preserving all the original statistical properties. Thus, the generated data can be distributed as a viable alternative to the adoption of complex data anonymization processes. More in general, synthetic data generation can be used with several objectives.

- *Data quality improvement*. Generated data fit the original patterns, hence, they are: smooth (with low presence of outliers); and denoised (since noise is not learnt);
- *Domain focus*. Generation algorithms allow generation control: empowerment of chosen patterns; class imbalance mitigation; and smart oversampling;
- *Reducing the costs of data collection*. Collecting brand new data may be expensive in term of resources or time;

- *Support for next data manipulation and analysis*. Smart data compression (data can be summarized by the functions and the patterns learnt by the data generation algorithms); Improved classification (the generating algorithms can be exploited to produce samples hard to predict, in order to force any classifier to better define their decision boundaries);
- *Data obfuscation*, since synthetic data generation enables a control of which information to highlight or hide.

As future work, an interesting research line is trying to investigate and define 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 (Saccà et al., 2019). This extension considers more structured schemes for the datasets to be generated, as required by emerging big data applications, for example, social network analytics. To this end, the basic simple schema $R(K,A)$ is replaced with a more general NoSQL schema $R\left(K,A_1,...,A_p,A_1,\ ...,\ A_q\right)$, where K is the table key, $A_1,...,A_p$ are single-valued attributes, and $A_1,...,A_q$ are multi-valued attributes. This problem reduces to classical case when $p=0$ and $q=1$, that is, there is exactly one multi-valued attribute. As an example, consider individuals who are characterized by the SV attributes *Gender*, *Location* and *Age* and by the MV attributes *Groups* and *Events*: an individual may belong to various groups and may attend a number of events. A transaction $I=[Male,Rome,25,\{g_1,\ g_4\},\ \{e_1,e_3\}]$ represents a 25-year old male individual located in Rome who belongs to the groups g_1 and g_4 and attends the events e_1 and e_3.

ACKNOWLEDGMENT

This work has been partially funded by the PON-MIUR Project ARS01_00587 "SecureOpenNet: Distributed Ledgers for Secure Open Communities—SON," by the EU H2020 ICT48 Project "HumanE-AI-Net" under contract #952026, and by the National Science Foundation under the award #1820685. Open Access Funding provided by Universita della Calabria within the CRUI-CARE Agreement. [Correction added on 25 May 2022, after first online publication: CRUI funding statement has been added.]

CONFLICT OF INTEREST

The authors have declared no conflicts of interest for this article.

AUTHOR CONTRIBUTIONS

Giuseppe Manco: Methodology (equal); supervision (equal); writing – review and editing (equal). **Ettore Ritacco:** Data curation (equal); investigation (equal); software (equal); writing – original draft (equal). **Antonino Rullo:** Data curation (equal); investigation (equal); validation (equal); visualization (equal). **Domenico Sacca':** Conceptualization (equal); supervision (equal). **Edoardo Serra:** Data curation (equal); investigation (equal); software (equal); writing – original draft (equal).

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

ENDNOTE

[1] A nice property of the Gumbel distribution is its reparametrization: given $u\sim\text{Uniform}(0,1)$, we have that $-\log(-\log u)\sim\text{Gumbel}(0,1)$.

REFERENCES

Agrawal, R., Imieliński, T. & Swami, A. (1993a). Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD International conference on Management of data. Washington, DC, USA, May 26-28, 1993. Vol. 22, pp. 207–216.

Agrawal, R., Imieliński, T., & Swami, A. (1993b). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD'93. ACM, New York, NY. pp. 207–216.

Agrawal, R. & Srikant, R. (2000). Privacy-preserving data mining. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD'00. ACM, New York, NY. pp. 439–450.

Ansari, M., Rasoolian, B. & Smith, J. S., & Synthetic Order Data Generator for Picking Data. (2018). 15th IMHRC Proceedings (Savannah, Georgia. USA – 2018). 15. https://digitalcommons.georgiasouthern.edu/pmhr_2018/15

Arasu, A., Kaushik, R. & Li, J. (2011). Data generation using declarative constraints. Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. Athens, Greece, June 12-16. pp. 685–696.

Arjovsky, M. & Bottou, L. (2017). Towards principled methods for training generative adversarial networks. Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. *Proceedings of the 34th International Conference on Machine Learning*. pp. 214–223.

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In I. Guyon, G. Dror, V. Lemaire, G. Taylor, & D. Silver (Eds.). *Proceedings of ICML workshop on unsupervised and transfer learning*, *Volume 27 of Proceedings of Machine Learning Research*. PMLR, Bellevue, WA. pp. 37–49.

Bárány, V., Cate, B. T., Kimelfeld, B., Olteanu, D., & Vagena, Z. (2017). Declarative probabilistic programming with datalog. *ACM Transactions on Database Systems (TODS)*, *42*(4), 1–35.

Beheshti, A. K., & Hejazi, S. R. (2015). A novel hybrid column generation-metaheuristic approach for the vehicle routing problem with general soft time window. *Information Sciences*, *316*, 598–615.

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to linear optimization*. Athena Scientific.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, *112*(518), 859–877.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Bruno, N. & Chaudhuri, S. (2005). Flexible database generators. Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, pp. 1097–1107.

Cagliero, L., & Garza, P. (2013). Itemset generalization with cardinality-based constraints. *Information Sciences*, *244*, 161–174.

Calders, T. (2004). Computational complexity of itemset frequency satisfiability. *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04. ACM, New York. pp. 143–154.

Calders, T. (2007). The complexity of satisfying constraints on databases of transactions. *Acta Informatica*, *44*(7–8), 591–624.

Chen, C. P., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, *275*, 314–347.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38.

Fowkes, J. & Sutton, C. (2016). A bayesian network model for interesting itemsets. Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer. pp. 410–425

Gilad, A., Patwa, S., & Machanavajjhala, A. (2021). Synthesizing linked data under cardinality and integrity constraints. *arXiv preprint arXiv: 2103.14435*.

Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, *9*(6), 849–859.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014a). Generative adversarial nets. Advances in Neural Information Processing Systems. Vol. 27, December 8-13 2014, Montreal, Quebec, Canada.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014b). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Greco, G., Guzzo, A. & Nardiello, G. (2020). FD-VAE: A feature driven VAE architecture for flexible synthetic data generation. International Conference on Database and Expert Systems Applications, Bratislava, Slovakia. Springer. pp. 188–197

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. (2017). Improved training of wasserstein GANs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Long Beach, CA, USA, pp. 5769–5779.

Gunopulos, D., Khardon, R., Mannila, H. & Toivonen, H. (1997). Data mining, hypergraph transversals, and machine learning. In A. O. Mendelzon & Z. M. Özsoyoglu (Eds.). Proceedings of the 16-th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '97, Tucson, Arizona, USA. ACM Press. pp. 209–216

Guns, T., Nijssen, S., & Raedt, L. D. (2011). Itemset mining: A constraint programming perspective. *Artificial Intelligence*, *175*(12), 1951–1983.

Guzzo, A., Moccia, L., Saccà, D., & Serra, E. (2013). Solving inverse frequent itemset mining with infrequency constraints via large-scale linear programs. *ACM Transactions on Knowledge Discovery from Data*, *7*(4), 18:1–18:39.

Guzzo, A., Saccà, D., & Serra, E. (2009). An effective approach to inverse frequent set mining. *Proceedings of the 2009 9th IEEE International Conference on Data Mining*, ICDM '09. IEEE Computer Society, Washington, DC. pp. 806–811.

Han, J., Cheng, H., Xin, D., & Yan, X. (2007). Frequent pattern mining: Current status and future directions. *Data Mining and Knowledge Discovery*, *15*(1), 55–86.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM SIGMOD Record*, *29*(2), 1–12.

Hoag, J. E., & Thompson, C. W. (2007). A parallel general-purpose synthetic data generator. *ACM SIGMOD Record*, *36*(1), 19–24.

Houkjær, K., Torp, K. & Wind, R. (2006). Simple and realistic data generation. Proceedings of the 32nd International Conference on Very Large Databases, Seoul, Korea, pp. 1243–1246.

Jang, E., Gu, S. & Poole, B. (2017). Categorical reparameterization with gumbel-softmax. Proceedings of the 5th International Conference on Learning Representations, Toulon, France, (ICLR'17).

Jindal, R., & Malaya, D. B. (2016). A novel approach for mining frequent patterns from incremental data. *International Journal of Data Mining, Modelling and Management*, *8*(3), 244–264.

KDDCUP 2000 http://www.ecn.purdue.edu/KDDCUP.

Kingma, D. & Welling, M. (2014). Auto-encoding variational bayes. Proceedings of the 2nd International Conference on Learning Representations, ICLR'14,

Kingma, D. P. & Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Laxman, S., Naldurg, P., Sripada, R. & Venkatesan, R (2007). Connections between mining frequent itemsets & learning generative models. Seventh IEEE International Conference on Data Mining (ICDM 2007), Omaha, Nebraska, USA. IEEE. pp. 571–576.

Lezcano, C. & Arias, M. (2019). Synthetic dataset generation with itemset-based generative models. 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, October 27-30, 201. IEEE. pp. 288–293

Li, Y., Zhang, R., Yang, X., Zhang, Z. & Zhou, A. (2018). Touchstone: Generating enormous query-aware test databases. 2018 {USENIX} annual Technical Conference ({USENIX}{ATC} 18), Boston, MA, USA. pp. 575–586.

Liang, D., Krishnan, R. G., Hoffman, M. & Jebara, T. (2018). Variational autoencoders for collaborative filtering. Proceedings of the 2018 World WideWeb Conference,WWW'18, Lyon, France. pp. 689–698.

Luenberger, D. G. (2003). *Linear and nonlinear programming* (2nd ed.). Springer.

Maddison, C. J., Mnih, A. & Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. Proceedings of the 5th International Conference on Learning Representations, (ICLR'17), Toulon, France.

Mampaey, M., Tatti, N. & Vreeken, J. (2011). Tell me what i need to know: Succinctly summarizing data with itemsets. Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA. pp. 573–581.

McLachlan, G. J., & Peel, D. (2000). *Finite mixture models*. Wiley.

Michael, K., & Miller, K. W. (2013). Big data: New opportunities and new challenges [guest editors' introduction]. *Computer*, *46*(6), 22–24.

Mielikainen, T. (2003). On inverse frequent set mining. *Proceedings of 2nd Workshop on Privacy Preserving Data Mining*, PPDM'03. IEEE Computer Society, Washington, DC. pp. 18–23.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. The MIT Press.

Papadimitriou, C. H. (1994). *Computational complexity*. Addison-Wesley.

Ravikumar, G., Manjunath, T., Hegadi, R. S., & Umesh, I. (2011). A survey on recent trends, process and development in data masking for testing. *International Journal of Computer Science Issues (IJCSI)*, *8*(2), 535.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *Proceedings of the 31th International Conference on Machine Learning*, *ICML 2014*, *Beijing*, *China*, *21–26 June 2014*, Volume 32 of *JMLR Workshop And Conference Proceedings*. pp. 1278–1286. JMLR.org.

Roth, K., Lucchi, A., Nowozin, S., & Hofmann, T. (2017). Stabilizing training of generative adversarial networks through regularization.

Saccà, D., Serra, E., & Rullo, A. (2019). Extending inverse frequent itemsets mining to generate realistic datasets: Complexity, accuracy and emerging applications. *Data Mining and Knowledge Discovery*, *33*(6), 1736–1774.

Sanghi, A., Sood, R., Haritsa, J. R. & Tirthapura, S. (2018). Scalable and dynamic regeneration of big data volumes. International Conference on Extending Database Technology (EDBT), Vienna, Austria.. pp. 301–312.

Smets, K. & Vreeken, J. (2012). Slim: Directly mining descriptive patterns. Proceedings of the 2012 SIAM International Conference on Data Mining, Brussels, Belgium Belgium. SIAM. pp. 236–247.

Sutton, R. S., McAllester, D., Singh, S. & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. Advances in Neural Information Processing Systems, Denver, Colorado, USA. Vol. 12.

Taşer, P. Y., Birant, K. U., & Birant, D. (2020). Multitask-based association rule mining. *Turkish Journal of Electrical Engineering & Computer Sciences*, *28*(2), 933–955.

Theis, L., van den Oord, A. & Bethge, M. (2016). A note on the evaluation of generative models. International Conference on Learning Representations (ICLR), San Juan, Puerto Rico.

Uno, T., Asai, T., Uchida, Y., & Arimura, H. (2003). LCM: An efficient algorithm for enumerating frequent closed item sets. *Fimi*. Vol. 90. Citeseer.

Vreeken, J., Van Leeuwen, M., & Siebes, A. (2011). Krimp: Mining itemsets that compress. *Data Mining and Knowledge Discovery*, *23*(1), 169–214.

Weikum, G. (2013). Where's the data in the big data wave? ACM Sigmod Blog, MARCH 6, 2013. http://wp.sigmod.org/?p=786.

Wong, I. S., Dobbie, G. & Koh, Y. S. (2019). Items2data: Generating synthetic boolean datasets from itemsets. Australasian Database Conference, Sydney, NSW, Australia. Springer. pp. 79–90.

Wu, H., Ning, Y., Chakraborty, P., Vreeken, J., Tatti, N., & Ramakrishnan, N. (2018). Generating realistic synthetic population datasets. *ACM Transactions on Knowledge Discovery from Data*, *12*(4), 45:1–45:22.

Wu, X., Wu, Y., Wang, Y., & Li, Y. (2005). Privacy aware market basket data set generation: A feasible approach for inverse frequent set mining. *Proceedings of SIAM International Conference on Data Mining*, SDM'05. SIAM, Philadelphia, PA. pp. 103–114.

Xie, S. M. & Ermon, S. (2019). Reparameterizable subset sampling via continuous relaxations. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19), Macao, SAR, China.

Xu, Z., Liu, B., Wang, B., Sun, C., Wang, X., Wang, Z. & Qi, C. (2017). Neural response generation via GAN with an approximate embedding layer. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), Copenhagen, Denmark.

Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). Parallel algorithms for discovery of association rules. *Data Mining and Knowledge Discovery*, *1*(4), 343–373.

Zhong, S. (2007). Privacy-preserving algorithms for distributed mining of frequent itemsets. *Information Sciences*, *177*(2), 490–503.

How to cite this article: Manco, G., Ritacco, E., Rullo, A., Saccà, D., & Serra, E. (2022). Machine learning methods for generating high dimensional discrete datasets. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *12*(2), e1450. https://doi.org/10.1002/widm.1450