Differentiable Pattern Set Mining

Jonas Fischer
Max Planck Institute for Informatics
Saarbrücken, Germany
fischer@mpi-inf.mpg.de

Jilles Vreeken
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
jv@cispa.de

ABSTRACT

Pattern set mining has been successful in discovering small sets of highly informative and useful patterns from data. To find good models, existing methods heuristically explore the twice-exponential search space over all possible pattern sets in a combinatorial way, by which they are limited to data over at most hundreds of features, as well as likely to get stuck in local minima. Here, we propose a gradient based optimization approach that allows us to efficiently discover high-quality pattern sets from data of millions of rows and hundreds of thousands of features.

In particular, we propose a novel type of neural autoencoder called BinaPs, using binary activations and binarizing weights in each forward pass, which are directly interpretable as conjunctive patterns. For training, optimizing a data-sparsity aware reconstruction loss, continuous versions of the weights are learned in small, noisy steps. This formulation provides a link between the discrete search space and continuous optimization, thus allowing for a gradient based strategy to discover sets of high-quality and noise-robust patterns. Through extensive experiments on both synthetic and real world data, we show that BinaPs discovers high quality and noise robust patterns, and unique among all competitors, easily scales to data of supermarket transactions or biological variant calls.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

pattern mining, neural networks, explainability, autoencoder

ACM Reference Format:
Jonas Fischer and Jilles Vreeken. 2021. Differentiable Pattern Set Mining. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467348

1 INTRODUCTION

The goal of pattern mining is to discover those patterns – easily interpretable *symbolic* statements about the data – that give non-trivial insight in the process that generated it. Traditional approaches such as frequent pattern mining fail to deliver on this goal, as by returning *all* patterns that satisfy a user-defined 'interestingness' threshold they tend to swamp the analyst with extremely many results, most of which are redundant or spurious.

Modern approaches hence reformulate pattern mining as a model selection problem, in which we explicitly ask for a small *set* of patterns that together generalize the data well. This solves one problem, but creates another: the search space for pattern sets is even larger than that of patterns alone – doubly exponential in the number of features – and does not exhibit any structure that permits efficient search. As a result, existing methods all rely on heuristics, and are applicable only to modestly sized data of at most a few hundred features, by which modern biological applications such as genome-wide association studies or single-cell sequencing data with hundreds of thousands of features are far out of reach.

In this paper we propose a radically different strategy to pattern set mining that scales extremely well in both n and m, naturally handles noise, and copes equally well with sparse and dense data. We achieve this by taking a *differentiable* rather than a combinatorial approach. Our key idea is to learn a special kind of interpretable neural autoencoder for binary data, which we refer to as BinaPs, short for Binary Pattern Networks. These networks consist of two linear layers – the encoding hidden and the decoding output layer– with continuous weights, sharing weights between encoder and decoder. By using a novel type of binary activation function and binarizing weights during each forward pass, we can interpret the neurons in the hidden layer as 'classic' conjunctive patterns. Here, we propose a reconstruction loss that properly accounts for dense respectively sparse data, and as such the networks are equally well applicable to dense biological datasets as well as classical sparse transaction data.

One key benefit of our formulation is that it naturally allows for discovering noisy patterns: the network is rewarded for reconstructing the data, and hence automatically learns how much and which parts need to occur for a pattern to be considered as 'present'. Similar to other pattern set mining approaches, such as tiling [11] or boolean matrix factorization [21, 23], deciding the optimal size of the pattern set – the size of the hidden layer – is NP-hard. We however also show that in practice this is not a problem at all: initialized with a sufficiently large capacity, our networks drive the weights of edges towards 'surplus' neurons to zero and can so find an (almost) optimal number of patterns, even when initialized with as many hidden neurons as there are features. The overall most important benefit of BinaPs in contrast to existing methods, is however its massive scalability: the differentiable formulation is not only much easier to optimize, but also allows us to leverage the power of modern GPUs.

Evaluating BinaPs against the state-of-the-art on synthetic data, we show that BinaPs accurately retrieves the ground truth, and is robust to noise. BinaPs is the only method applicable on datasets

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467348

with truly large n and m, on which we confirm that it discovers meaningful patterns that provide on-trivial insight. It is also applicable to moderately sized data, on which we show that it performs at least as well as the state-of-the-art. As a case study, we consider a recent biological dataset of over two hundred thousand features on which BinaPs are able to mine patterns in mere minutes, recovering structure which can be confirmed by the literature, and generating new insights that could be leveraged by experts, such as potential roles of genes for which the functions are so far unknown.

The rest of this paper is organized as usual. We cover related work in Sec. 2, introduce BinaPs in Sec. 3, and empirically evaluate in Sec. 4. We round up with discussion and conclusions in Secs. 5 and 6. We provide additional details for reproducibility in the supplementary appendix, and make all code and data publicly available for research purposes.[1]

2 RELATED WORK

Pattern mining, a core topic of data mining, aims to explain local as well as global structures in data using easy to interpret patterns. After the seminal work by Agrawal and Srikant [1] on frequent itemset mining, research swiftly focused on more efficient techniques to mine itemsets and providing summarizations of all frequent patterns [2, 4, 13, 24]. Frequency alone, however, showed to be a poor measure of interestingness, yielding a plethora of spurious and redundant results [33]. Rather than using ad-hoc measures of interestingness and treating each pattern individually, research shifted to mine interesting *sets* of patterns that together describe the properties of the data. Different methods have been suggested to discover such pattern sets, leveraging constraint based approaches [26], boolean matrix factorization [21, 23], or information theoretic approaches that use the MDL principle [8, 9, 31, 34] or maximum entropy modeling [6, 19]. These methods retrieve succinct and interpretable summarizations of the data that do not suffer from the same problems as frequency mining, at the cost of a twice exponential search space that does not expose any easy to exploit structure. Hence, heuristic algorithms are employed that add patterns one by one in a bottom-up fashion, and are thus likely to get stuck in local minima. Furthermore, these approaches are sensitive to noise, and do not scale to modern data such as retail databases or bioinformatics data.

With the advent of Deep Learning, efficient GPU-based implementations to train neural networks experienced wide-spread adoption. In unsupervised settings, autoencoders can be trained to yield succinct representations of complex datasets, also providing local information in analogy to patterns [16, 17, 20]. Such networks are, however, inherently hard to interpret – especially with respect to how the structures that the network acrually learned link to the given input – and lack the clear interpretability of patterns barring their use for exploration. For low resource and embedded devices, network architectures with binarized $\{-1, 1\}$ weights and activations have been proposed, thus making model storage and application much more memory efficient [5, 7, 14, 18, 27, 28], for more information we refer to the review of Simons and Lee [30]. While they provide a discrete connection between neurons and input, these networks have no incentive to learn easy to interpret structures reflecting associations between input features. Preliminary experiments confirm this: these networks do not learn any evident easy-to-interpret relation between weights, neurons, and actual ground truth patterns.

Inspired by the link between continuous optimization of an objective and learning discrete variables, we propose to learn interpretable patterns using a novel constrained network architecture with $\{0, 1\}$-binarized weights and activations that reflect item-pattern relationships, and propose an objective that is suited as loss function for sparse transactional databases. The learned continuous versions of the weights, binarized for every forward pass through the network, allow to explore the discrete, exponential search space of patterns efficiently by gradient based continuous optimization. On synthetic and real world data, we show that our approach scales to large data sets far beyond what state-of-the-art pattern set miners achieve, while at the same time recovering the generating patterns more accurately even in the presence of noise.

3 THEORY

In this section, we propose BinaPs, a novel type of binarized neural autoencoder capable of learning pattern sets. We start by briefly introducing notation and then provide an informal overview of how BinaPs work. After introducing them formally, we discuss practical considerations, and provide theoretical analysis.

3.1 Notation

We consider binary input data $D \in \{0, 1\}^{n \times m}$ of n samples and m features. A sample $s_i \in D$ denotes the ith row in the data matrix $D[i]$. We denote the value of feature $j \in \{1, \ldots, m\}$ for the ith sample by $D[i, j]$. We are interested to find a set of patterns P, where each pattern $p \in P$ is a set of feature indices $p \subset \{1, 2, \ldots, m\}$ representing feature co-occurrences. To learn good sets of patterns P, we propose a binarized neural network architecture, where we generally denote W for a weight matrix, W_b for its binarized version, b for a bias, and b_d for its discretized version. Activation functions are denoted as λ, functions applying a part of the network to an input, e.g. the encoding layer, as f. To ease notation, whenever we apply a univariate function to a vector, we apply it to every entry in the vector. For instance, an activation function $\lambda : \mathbb{R} \to \{0, 1\}$ applied to $x \in \mathbb{R}^3$ yields $\lambda(x) \in \{0, 1\}^3$. Partial derivatives of a function f with respect to parameter x are written as $\frac{df}{dx}$, matrices M are generally denoted with capital letters, and vectors v with lower case letters.

3.2 The idea in a nutshell

For a given binary database, we aim to find that set of patterns P that together succinctly describe the data. That is, a good set of patterns should cover the database non-redundantly while minimizing the reconstruction loss if we reconstruct the database from P alone. For continuous and unstructured data, autoencoder proved to be a successful tool to capture the main structure in the data by minimizing reconstruction loss. An autoencoder is a neural network consisting of task-specific encoding layers that end in an embedding layer, and a symmetric decoder to reconstruct the input from the embedding layer. The embedding layer is usually small compared to the input

[1] http://eda.mmci.uni-saarland.de/binaps/

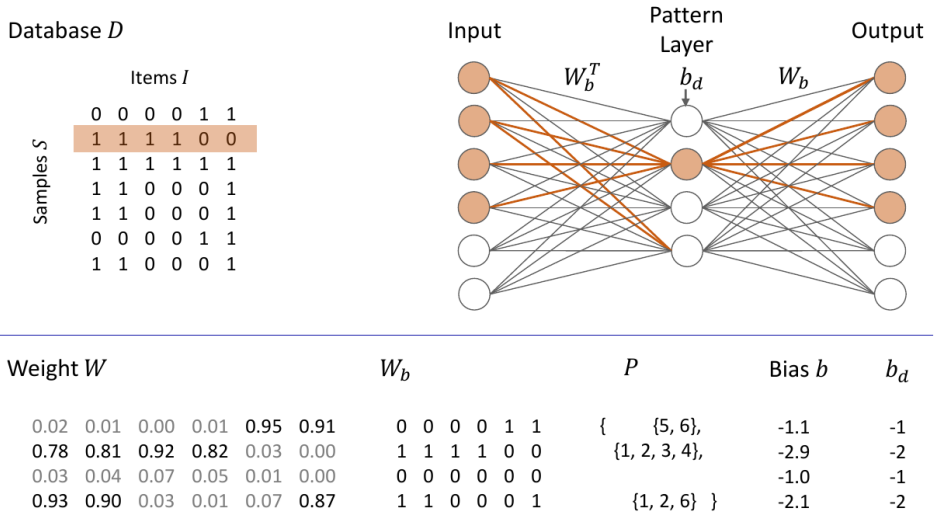


Figure 1: *Example database and BinaPs.* A binary database D is given in the top left, a binary pattern network with pattern layer size $k = 4$ on the right. Example continuous weights W and bias b used during backpropagation, and their *binarized*, respectively *discretized* counterparts W_b and b_d used during the forward pass, are given at the bottom, alongside the pattern set P that we can directly derive from these weights. Neuron activations (orange) for the orange sample in D are given according to the example binarized weights and bias.

layer, thus imposing an information bottleneck and forcing the network to learn relevant and shared structure between inputs. Neural networks are, however, inherently hard to understand for a human, as connections between input and neurons are non-symbolic and often non-linear.

Here, we propose a novel kind of neural autoencoder, where weights and activations are taking values in $\{0, 1\}$ during the forward pass. To learn in small noisy steps during backpropagation, for training we use continuous versions of the weights, optimizing reconstruction loss with respect to these continuous weights. The autoencoder consists of one linear hidden layer – the pattern layer – and one linear output layer. For each neuron in the hidden layer, incoming binary weights indicate whether an input item is part of the encoded pattern, e.g. binarized weight $W_b[i, j]$ means that input item j is part of the pattern given by hidden neuron i. As such, each neuron in the hidden layer corresponds to a pattern p, while all neurons together correspond to the pattern set P.

We construct binarized versions of the weights forward pass, and use these for reconstruction. To ensure that the hidden neurons correspond to actual patterns, and that such patterns are interpretable, two additional constraints are introduced. First, we introduce a discrete negative bias to the pattern layer to make sure that a minimum number of items is required to be present in the input for the neuron to fire – respectively the pattern to hold. Second, we "mirror" the weight matrix, such that the weight of the decoding layer is the weight of the encoding layer transposed. This ensures that the input/output relation is fixed, thus avoids that if a neuron fires – meaning a pattern over a set of input neurons holds – it activates exactly those output neurons that are part of the pattern. We give an example network along with a toy database in Fig. 1.

3.3 BinaPs – Binary Pattern Networks

We define Binary Pattern Networks (BinaPs) as autoencoder with two linear layers. Each layer has binary weights and binary activations, and a continuous version of weights and bias for backpropagation. We first formally define the functions that this stack of layers encodes given binary weights – corresponding to the forward pass of the network – and then provide the derivatives of a loss function with respect to the continuous version of the weights – corresponding to the backward pass – and show how we can binarize the continuous weights for the next iteration.

The forward pass. A binary linear layer is generally defined as

$$f_{W_b}(x) = xW_b^\intercal \,,$$

with input x and binarized weights W_b. For sample $s \in \{0, 1\}^m$, in the forward pass, the function of the encoding layer takes the form

$$f_E(s) = \lambda_E(f_{W_b}(s)) \,,$$

where $W_b \in \{0, 1\}^{k \times m}$ is the binary weight matrix for the hidden layer of size k, and $\lambda_E : \mathbb{R} \to \{0, 1\}$ is a binary activation function.

The function of the decoding layer, taking as input the result of the encoding layer $y \in \{0, 1\}^k$, is defined as

$$f_D(y) = \lambda_D(f_{W_b^\intercal}(y)) \,,$$

where W_b is the same weight matrix as in the encoding layer.

The activation function consists of two parts, a learnable bias term b that allows to train how many items of this pattern have to be present in the input for the neuron to fire, and a binarization to

produce the output signal. First, we define the clamping function

$$\text{clamp}(x, a, b) = \begin{cases} a & \text{if } x < a \,, \\ x & \text{if } a \leq x \leq b \,, \\ b & \text{if } b < x \,. \end{cases}$$

The activation of the encoding layer is then given as

$$\lambda_E(x) = \text{round}(\text{clamp}(x + b_d, 0, 1)) \,,$$

where $x \in \mathbb{N}^k$, and $b_d \in \{-\infty, \ldots, -2, -1\}^k$ is the discretized bias parameter. The activation function of the decoding layer is $\lambda_D(x) = \text{round}(\text{clamp}(x, 0, 1))$, where no bias is involved. This concludes the definitions of all components of BinaPs in a forward pass, which is thus computed by $f_D(f_E(s))$ for given input s.

The backward pass. To learn a good model – a good set of weights and biases – for a given dataset D, our aim is to minimize the reconstruction loss across all samples $s \in D$

$$L(D; W, b) = \sum_{s \in D} ||f_D(f_E(s)) - s|| \,.$$

Given the loss term for a given batch of samples, we can then compute derivatives with respect to the continuous weights W and biases b, and backpropagate through the network. Networks are trained using gradient descent, for which the derivatives with respect to the parameters can be obtained using the chain rule. For the linear layer, the derivative with respect to the weights W, input x and incoming gradient flow g_o is similar to vanilla neural networks, given as

$$\frac{df_{W_b}}{dW} = g_o^\intercal x \,, \qquad \frac{df_{W_b}}{dx} = g_o W \,.$$

The activation functions λ in each layer resemble a step function, hence there does not exist an analytical solution to their derivative. For the decoder activation λ_D we use a version of the straight-through-estimator $\frac{d\lambda_D}{dx} = \mathbb{1} g_o$ as suggested by Bengio et al. [3].

For the hidden layer, the straight-through-estimator fails, as wrongly predicted items backpropagate a negative gradient to weights through the activation, even though the neuron was not active. In other words, patterns get penalized for wrong reconstruction, regardless of whether they were actually taking part in the reconstruction. Thus, we propose the *gated* straight-through-estimator, which distinguishes the two cases of whether λ_E was activated on input x or not. For given input x and incoming gradient flow g_o, we define gradients with respect to bias b and input x as

$$\frac{d\lambda_E}{db} = \begin{cases} g_o & \text{if } \lambda_E(x) = 1 \\ 0 & \text{if } \lambda_E(x) = 0 \end{cases}, \qquad \frac{d\lambda_E}{dx} = \begin{cases} g_o & \text{if } \lambda_E(x) = 1 \\ \max(0, g_o) & \text{if } \lambda_E(x) = 0 \end{cases}.$$

With the partial derivatives defined, the chain rule allows us to propagate gradients through the network and update the continuous parameters W and b accordingly. After backpropagation, we clamp the weights to be in range $[0, 1]$, which allow us to use a stochastic binarization. We clamp the bias to be at maximum -1, which means that the bias acts as a learnable threshold for a minimum number of items to be present for a pattern to fire.

From continuous to discrete. In the next forward pass, we then binarize respectively discretize the weight and bias parameter. As the weights are in range $[0, 1]$, we can use a stochastic interpretation and treat them as a Bernoulli variable, the binarization thus becomes a draw from a Bernoulli $\mathcal{B}$,

$$W_b[i, j] = \mathcal{B}(W[i, j]) \,.$$

Hence, the binary versions of the weights change probabilistically in every iteration. For the discretization of the bias we ceil the values $b_d[i] = \lceil b[i] \rceil$.

The algorithm. With all information in place, we are able to train a binary pattern network, for which we provide pseudocode as Alg. 1. In practice, we use batch learning, where line 3 is replaced by an appropriate batch slicing of the database, and all operations become the corresponding tensor operations. The pattern set P encoded by the network is given by a binarization of the final weight matrix $W_P = \text{round}(W)$, where row i encodes a pattern that contains each item j for which $W_P[i, j] = 1$.

Reconstruction loss for transaction data. Binary matrices, and in particular sparse transaction databases, impose an additional challenge when optimizing the loss. In analogy to a classification problem with imbalanced class labels, where there is an intrinsic bias towards correctly classifying the overrepresented class, sparsity introduces a huge imbalance when it comes to reconstruction as 1s are highly underrepresented yet most important for mining patterns. In extreme cases, such as very sparse data usually given by supermarket transaction data, a model achieves very low reconstruction loss by predicting 0s everywhere – similar to always predicting the overrepresented class in the classification analogy. We thus propose a sparsity dependent reconstruction loss, which for given sample

Algorithm 1: BinaPs training

input : dataset D, initial BinaPs (W^1, b^1), number of epochs $e_{\max}$, batch size l
output : pattern set P

1 $t \leftarrow 0$; // Step
2 **for** $e = 1 \ldots e_{\max}$ **do** // Epochs
3 **for** $i = 1 \ldots n$ **do** // For each sample
  // **Forward pass**
4 $W_b^t \leftarrow \mathcal{B}(W^t)$; // Binarize weights
5 $b_d^t \leftarrow \lceil b^t \rceil$; // Discretize bias
6 $y_i \leftarrow \text{round}(\text{clamp}(D[i](W_b^t)^\intercal + b_d^t, 0, 1))$; // f_E
7 $z_i \leftarrow \text{round}(\text{clamp}(y_i W_b^t, 0, 1))$; // f_D
8 $L(D[i]; W, b)$; // Compute loss
  // **Backward pass**
9 $g_W \leftarrow \frac{dL}{dW}$; // Weight gradient
10 $g_b \leftarrow \frac{dL}{db}$; // Bias gradient
11 $W^{t+1} \leftarrow \text{update}(W^t, g_W)$; // Optimizer step
12 $b^{t+1} \leftarrow \text{update}(b^t, g_b)$;
13 $t \leftarrow t + 1$;
14 $P = \{\{j \mid \text{round}(W^t[i, j]) = 1\} \mid i \in 1 \ldots k\}$; // Pattern set
15 **return** P

$D[i]$, and reconstruction z_i, is

$$L_\alpha(D[i]; W, b) = \sum_{j \in [1,m]} \big((1-D[i,j])\alpha + D[i,j](1-\alpha)\big)|z_{ij} - D[i,j]| \,,$$

where $\alpha = \frac{\#1s}{\#1s+\#0s}$ is the sparsity of the data, and z_{ij} is the reconstructed feature j for sample i. The loss function thus avoids aforementioned intrinsic biases by properly modeling sparsity in the reconstruction.

Escaping poor local minima. The probabilistic nature of our binarization allows us to escape poor local minima due to the stochasticity introduced by drawing the binarized weights each round. By bounding the continuous version of the weights by a very small but positive value from below, we ensure that with low probability, every item could potentially still be learned for a neuron. In particular, we adapt the clamping of weights after updating the weights during backpropagation in iteration t such that

$$W^{t+1} = \text{clamp}(W^{t+1}, 1/m, 1) \,,$$

where the minimum $1/m$ is chosen such that on expectation 1 out of m items is randomly assigned to the pattern of a neuron, independent of data dimensions. This helps to escape poor local minima by e.g. preventing neurons from dying – meaning that the gradient of corresponding weights and biases would always be zero.

Initialization. To initialize the model, we can again take advantage of the stochasticity of the weights. By setting the initial weights $W^1_{ij} = 1/m$ for data of size m, we have uniform chances of setting any particular weight to 1, and have an initialization that is insensitive to the number of features m, thus allowing to learn even small patterns properly early on in the optimization. To enforce learning of proper patterns at the beginning of the optimization, we set all bias terms in the encoding layer to -1, which means that at least two items have to be present for a neuron to fire.

Size of hidden layer. So far, we assumed the architecture, and in particular the size of the hidden layer c, to be fixed. This hyperparameter, which we call *capacity*, corresponds to the maximum size a retrieved pattern set can attain. Existing work, often assumes the size of the optimal pattern set k^* to be given, as deciding k^* for BMF [22] and related problems such as minimum tiling of databases is NP-hard [11], where the optimal pattern set is the smallest set of patterns that together reconstruct the database. Similarly, to decide if the hidden layer capacity c corresponds to the size of an optimal pattern set, is NP-hard.

Theorem 3.1 (Estimating hidden layer size c^* is NP-hard). *For a given database D consisting of patterns P, estimating the smallest hidden layer size $c^* = |P|$, which would still result in smallest reconstruction loss, is NP-hard.*

Proof. We prove the theorem by introducing the concept of bipartite dimensions of a graph, which is known to be NP-hard, on which we reduce our problem.

Definition 3.2 (Bipartite Dimension). For a bipartite graph $G = (V_1 \cup V_2, E)$, the bipartite dimension is the minimum number of bicliques between V_1 and V_2 required to cover all edges E.

Lemma 3.3. *Deciding the bipartite dimension for a graph G is NP-hard [10, GT18].*

For a given database D of items $\mathcal{I}$ and sample identifier $\mathcal{S} = \{1, 2, \ldots, n\}$, the binary database matrix corresponds to an adjacency matrix of the (undirected) bipartite graph spanned by the two node sets of all items and all sample identifiers, with an item and an identifier having an edge iff the item occurs in the corresponding sample. More formally, we define $G_D = (\mathcal{I} \cup \mathcal{S}, \{(i,j) \in \mathcal{I} \times \mathcal{S} \mid D[i,j] = 1\})$. The perfect reconstruction of the database D with minimal number of patterns thus corresponds to the minimal number of bicliques of G_D, with each biclique corresponding to a pattern. Hence, the minimal number of patterns k^* and thereby the hidden layer size c^* required to achieve perfect reconstruction corresponds to the bipartite dimension of the underlying graph, which is NP hard to decide (Lemma 3.3). □

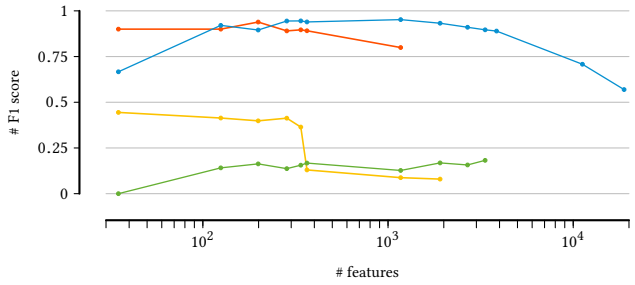
The good news is, however, that binary pattern networks – as opposed to other approaches such as BMF – are robust to optimistic estimates of k^*, i.e. the capacity c of the network only serves as an upper bound for the number of patterns it will retrieve. In practice, we can set e.g. $c = m$ and the network simply sets incoming weights to surplus neurons to 0, thus shrinking the pattern set to an almost optimal size. It is important to note that by setting c to the number of items, the network does not memorize the input, as it is forced to learn proper patterns due to the bias term $b \leq -1$.

Complexity. The complexity of learning binary pattern networks is the same as for vanilla neural networks, which is $O(t \times m^2 c)$ for t iterations and data of n samples and m items and a BinaPs of capacity c. This term is dominated by the matrix multiplication, for simplicity we assumed naive matrix multiplication. As the search space for pattern sets is twice exponential in the number of features and usually does not lend itself for easy to exploit structure, a low-degree polynomial complexity is great news. State-of-the-art approaches based on e.g. information theory do not provide differentiable objectives, the major issue being the discreteness of the search space, hence they resort to heuristic algorithms. Here, by having a link between the discrete search space and the continuous loss, we circumvent this problem, thus resulting in a differentiable objective optimizable in polynomial time. In practice, GPUs drastically speed up the optimization even further, allowing us to scale up to and beyond hundreds of thousands of features.

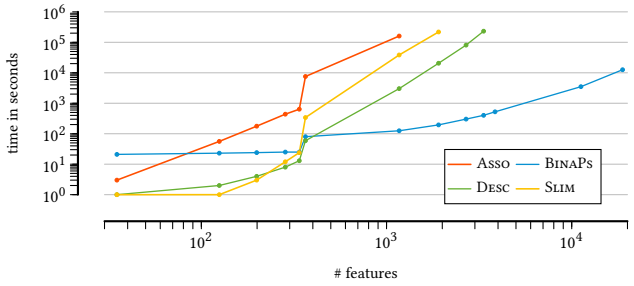
4 EXPERIMENTS

Here, we empirically evaluate BinaPs on both synthetic as well as real world data. For that, we implemented our approach in `PyTorch` and compare to state of the art pattern mining methods that leverage different objectives and approaches. We use the publicly available implementations of all methods, and make our implementation available.[2] All experiments of existing methods were carried out on Intel Xeon E5-2643 v3 machines with 256GB RAM running Linux, BinaPs were trained on an NVIDIA A100-SXM4 GPU with 40GB memory. Existing methods supporting multithreading were executed in parallel on 16 cores. In particular, we compare to Asso, which is based on boolean matrix factorization, and use an automated rank selection approach [23], Slim, an information theoretic pattern set miner based on the Minimum Description Length principle [31], and Desc, which mines pattern sets using

[2] http://eda.mmci.uni-saarland.de/binaps/



(a) F1 score on scale data (higher is better).



(b) Runtime on scale data (lower is better).

Figure 2: *BinaPs is scalable.* For synthetic data with known ground truth of varying number of features and planted patterns, we show the F1 score (a) and the runtime (b) for all methods. Experiments were aborted when exceeding 3 days of runtime.

maximum entropy modeling [6]. Preliminary experiments showed that vanilla binary networks (BNNs) [5], do not yield any evident easy-to-interpret relation between weights, neurons, and ground truth patterns. Individual experiments were stopped if taking more than 3 days. Asso is given the maximum k, and BinaPs capacity c, equal to the number of features m for all experiments. Despite being large overestimates of the actual pattern set sizes, both methods show to be robust to this choice, hence we did not optimise these parameters further. For all methods, we optimised hyperparamters for each dataset separately, for details we refer to App. A.1.

4.1 Recovering ground truth

To evaluate all methods with respect to scalability and robustness to noise, we first generate synthetic data with known ground truth. We consider data with various number of planted patterns, where each pattern is assigned 2 to 10 features at random, and a random subset of the samples of size drawn from $\mathcal{N}(.05n, .005n)$. Here, .05 corresponds to the mean density of patterns, similar to sparse real world data, and n is the number of samples. In a first set of experiments, we vary the number of samples n, to show that BinaPs can handle small data well. In a second set of experiments, we vary the number of planted patterns, comparing BinaPs with existing methods with respect to scalability. In a third set of experiments, we introduce varying amounts of noise, to evaluate how well the approaches can cope with noise. To evaluate how well the planted pattern sets are retrieved, we consider F1 – the harmonic mean between precision and recall – which is defined as

$$\mathrm{F1}(P_d, P_g) = \frac{|P_d \cap P_g|}{|P_d \cap P_g| + \frac{1}{2}|P_d \ominus P_g|},$$

measuring how well discovered pattern set P_d matches the ground truth P_g, with $\ominus$ the symmetric difference between two sets. An intersection of two pattern sets P, P' is defined as true matches between individual patterns, i.e. $P \cap P' = \{p \mid p \in P \wedge p \in P'\}$.

Small n. We start by confirming whether BinaPs can cope with *small* data. To this end we generate synthetic data density and distribution as above, planting 100 different patterns in $n = \{1000, 1100, \ldots, 10\,000\}$ samples, flipping 0.1% of data entries at random, corresponding to a signal to noise ratio of 20. Although 1000 samples may not sound very small, the probability for creating spurious but

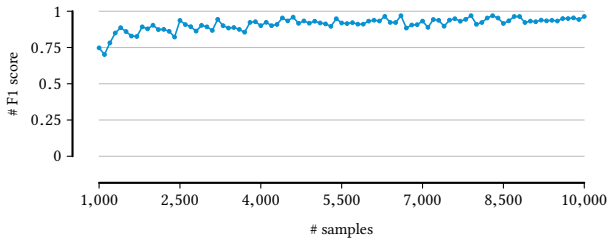


Figure 3: *BinaPs is applicable on data with few samples.* F1 score for pattern sets discovered by BinaPs on synthetic data with varying number of samples.

statistically significant co-occurrences for just one pattern when planting 100 patterns is already 23% (see App. A.2 for the derivation). In other words, considering fewer than 1000 samples does not make sense.

We report the results in terms of the F1 score in Fig. 3. We see that BinaPs robustly retrieves nearly all patterns even when the data consists of just a few thousand samples, already ably recovers 75% of the patterns completely from just 1000 samples. When investigating the retrieved pattern sets for the data of 1000 samples, we see that as expected by the above analysis, some of the retrieved patterns correspond to frequent co-occurences of ground truth patterns, which are counted as errors in the F1 score.

Recovering ground truth. To evaluate all methods with respect to scalability, we generate data with pattern sizes and densities as before, distributing patterns over $n = 10\,000$ samples uniformly at random, varying the number of different planted patterns in $\{10, 30, \ldots, 90\}$. As above, we introduce .1% of noise. Scaling further, we generate data of $n = 100\,000$ samples and plant $\{100, 300, \ldots, 900, 1000, 3000, 5000\}$ patterns, all of same frequency and level of noise. Here, we generate large data sets in terms of number of samples to avoid introducing spurious pattern sets, as two ground truth patterns are likely to co-occur significantly in small samples.

The results show that state-of-the-art methods Slim and Desc have trouble retrieving the ground truth (see Fig. 2a). They overfit respectively underfit. Asso as well as BinaPs do retrieve ground truth pattern sets accurately. We observe that Asso, Slim, and Desc all scale unfavourably, with Asso performing overall weakest in

terms of runtime (Fig. 2b), taking days for relatively small data sets, similar to Slim, and not being able to process even medium sized data sets. Desc shows to scale slightly better, being able to process moderately sized data sets, the retrieved patterns however not matching ground truth. In contrast, BinaPs scales to tens of thousands of patterns while retrieving pattern sets accurately, being able to process datasets with 100k samples and close to 20k features in slightly more than an hour.

Noise robustness. Next, we investigate the impact of noise on how well methods are able to retrieve the correct pattern set. We generate data of $n = 100\,000$ samples and 100 planted patterns with distributions as above, but now varying the percentage of noise in $\{0, 0.001, 0.002, \ldots, 0.009, 0.01, 0.02, \ldots, 0.05\}$, where 0.05 corresponds to a signal to noise ratio of 1. We visualize the results in Fig. 4a-4b. Both Asso and BinaPs are robust even to high amounts of noise, even for a signal to noise ratio of 1. Slim and Desc, on the other hand, are very sensitive to noise, both retrieving tremendous amounts of false patterns the more noise is present. On these datasets BinaPs run for minutes, whereas the competitors take up to several hours.

4.2 Quantitative results on real data

To evaluate BinaPs on real data, we consider 5 datasets of different dimensions. In particular, we consider click-stream data of the hungarian on-line news portal *Kosarak*,[3] data on Belgian traffic *Accidents* [12], *DNA*-amplification data [25], online grocery shopping data from *Instacart*, for which we give more information in App. A.3, and single nucleotide polymorphisms in genomes of human individuals from the 1000 *Genomes* project [32]. We provide details on how we processed *Genomes* in App. A.4. We provide statistics about the data and results in Table 1.

Whereas all methods are able to handel the small *DNA* and *Accidents* data, there are already orders of magnitude differences in runtime, with BinaPs finishing in minutes and Asso and Slim taking hours up to a day. With increasing number of columns, existing approaches fail to scale beyond small data, Asso not being able to handle *Instacart*. On larger data such as *Kosarak*, all existing approaches fail to terminate within 3 days, or run out of memory. BinaPs being the only approach to reliably handle large – both in n as well as m – databases, retrieving patterns for *Kosarak*, and the challenging *Genomes* data.

Looking at the statistics, BinaPs retrieves succinct and non-redundant pattern sets. Furthermore, these easy-to-interpret pattern sets are much smaller than the initial capacity c given to the network, despite not explicitly penalizing model complexity. While retrieving equally succinct pattern sets, Asso fails to scale to moderately sized data. Slim, an MDL based approach, finds thousands of partially redundant patterns already for medium sized data, which make it hard to analyze as a whole. Desc underfits for *Accidents* respectively *Instacart* only returning patterns of length 2.

4.3 Qualitative results on real data

Here, we will compare the results on two small datasets quantitatively, and then move on to analyze the insights gained for the new *Genomes* data set based on patterns found by BinaPs.

[3] http://fimi.uantwerpen.be/

DNA On this dense data, BinaPs and Asso discover compact patterns that exhibit a block structure, with consecutive items merged into a feature. Such block-like structures correspond to the larger chromosomal areas where DNA copy number amplification happens, and hence represent biologically meaningful patterns. For this particular data set, we can see the disadvantages of iterative heuristics such as from Slim, which learns parts of this block structure early on, but then overfits to overly large patterns that only appear in few samples and without evident block structure. These co-occurrences of larger patterns are however likely due to random co-occurrences. Desc here only finds short patterns, likely being caught in a local minimum.

Instacart BinaPs discover patterns describing dense, noisy blocks, for example a pattern spanning dozens of fruits, which are bought together in arbitrary combinations. Slim breaks such dense blocks in many (thousands) of individual patterns, each containing a small subset of items. Desc again underfits, finding patterns of length 2 only. BinaPs also discover small patterns that reveal the general buying behaviour of customers such as a pattern of different prepared dishes {Pizza, Ceasar salad, Hummus wrap, Filled wrap}, or certain food styles, such as {Chicken Wrap, Chile con queso}.

Genomes Last, we consider a dataset that motivated this work. The data contains information about variants of a population of human individuals for single nucleotide position within genes, and, with over 200 thousand features, *Genomes* is far beyond what existing methods can consider. BinaPs, however, is able to discover a succinct pattern set in mere minutes. Here, we analyze the discovered patterns in more detail.

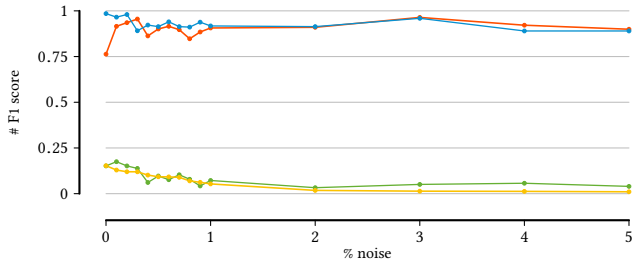
Biologically, we expect sequence stretches to be conserved and hence variants to be inherited "together". A first positive observation is that most patterns that BinaPs discovers indeed show such blocks of variants that are close-by on the genome.

Within such blocks, the rare variants often lie on the same allele – indicated by consecutive "0|1" variants – meaning that often one parent has the "common" reference variants for consecutive sites, whereas the other parent has the "rare" variants. Encouragingly, individual BinaPs patterns show this, but interestingly BinaPs often discovers a second pattern for the same sites that show the opposite, for example most sites to be "1|0". Whether this is due to phasing or has biological meaning, we leave to the experts.

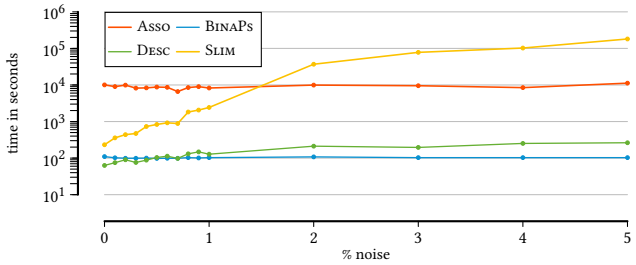
Taking a closer look at individual patterns, we find interesting connections between variants and the genes they occur in. For example, we find a pattern spanning multiple variants in the genes NUCB2, NCR3LG1, ABCC8, and ROMO1. Variants at NUCB2 and ABCC8 have together been associated with type 2 diabetes and high blood pressure in Japanese population [29]. For NCR3LG1 we know that it is close-by to ABCC8, but similar to ROMO1 there has not been any connection made between the variants of these genes, and hence could provide interesting insights for experts.

Another highly interesting pattern is over SF3A1, RRP7A, and Z82190, where SF3A1 and RRP7A encode proteins that are part of the ribosomal complex, the factory that produces proteins in a cell. Z82190 is a so far uncharacterized gene, and this pattern hence suggests what its role could be, which can guide future studies.

BinaPs discovers both large patterns of many variants within few genes, but also discover associations over many genes at once.



(a) F1 score on noise data (higher is better).



(b) Runtime on noise data (lower is better).

Figure 4: *BinaPs is robust to noise.* For synthetic data with known ground truth and varying the level of noise, we show the F1 score (a) and the runtime (b) for all methods. With 5% noise, we have a 1:1 ratio of signal-to-noise.

			# Patterns				Runtime			
Dataset	# rows	# cols	Asso	**BinaPs**	Desc	Slim	Asso	**BinaPs**	Desc	Slim
DNA	2458	391	134	131	345	281	4m	26s	20s	2s
Accidents	340183	468	133	78	215	12261	12h	6m	14m	21h
Instacart	2704831	1235	*n/a*	328	712	8119	∞	44m	25m	8h
Kosarak	990002	41270	*n/a*	302	*n/a*	*n/a*	∞	5h	∞	∞
Genomes	2504	226623	*n/a*	42	*n/a*	*n/a*	∞	9m	∞	∞

Table 1: *Results on Real World Data.* For five real world datasets, we give the number of rows, number of columns, and for Asso, BinaPs, Desc, and Slim we report the number of discovered patterns and runtime required to do so rounded to the most significant order of magnitude. Individual experiments were aborted after 3 days, resp. when exceeding the available 256GB of RAM, corresponding entries are marked with *n/a* and ∞.

As last example, we consider the pattern of variants in FUBP1, SELENOI, ZKSCAN5, TMEM225B, ASPN, SOX6, PLEKHA7, and ALX3. ASPN and ALX3 are taking part in chondrogenesis, an essential developmental process producing elastic tissue covering ends of bones and joints. FUBP1 and SOX6 are also genes linked to other developmental processes. For ZKSCAN5 and TMEM225B only little is known so far. These results encourage for further in-depth analysis by domain experts.

Overall, BinaPs retrieve informative patterns over variant sites, which can give interesting insights to relations between variants, genes, and their function.

5 DISCUSSION

Experiments show that BinaPs discover succinct descriptions of the data while scaling to hundreds of thousands of samples and features. On synthetic data, BinaPs reliably discover the ground truth pattern sets, even in the presence of large amounts of noise. On real data, BinaPs find succinct pattern sets that describe interesting structures in the data.

The results show that BinaPs discover patterns of varying length and frequency, that describe true structure of the given data, on sparse and dense data as well as small to large n, as well as m. Although naturally related to BinaPs, boolean matrix factorization shows good results on small data sets, but fails to scale beyond small data. Furthermore, it requires the rank of the factorization – the size of the pattern sets – to be given, wheras BinaPs needs a large enough capacity of the pattern layer, and shrinks the number of patterns – or factors – if necessary. State-of-the-art information theoretic methods, based on the Minimum Description Length principle or Maximum Entropy distributions, tend to over- or underfit, which is likely due to their heuristic combinatorial search rather than their objective. These methods do often recover (many) small fragments of ground truth patterns, which may be puzzled together by domain experts. In contrast, BinaPs tend to discover entire patterns, or larger chunks.

With BinaPs, we provide a link between the discrete, exponential search space and continuous optimization, which allows the use of gradient-based techniques to explore the large search space of pattern sets efficiently. In contrast to existing work, BinaPs are thus well equipped for the large datasets emerging e.g. in the biological domain, where exploring and understanding the given data is of great interest. Here, we considered a dataset of human variation stemming from the 1000 Genomes project, which deeply sequenced thousands of human individuals. Considering single nucleotide polymorphisms occuring within genes, resulting in several hundred thousand features, BinaPs retrieved a succinct pattern set that upon close inspection revealed known biological structure, as well as interesting new relationships between variant sites, genes, and their function. These pattern sets could generally help experts to better understand and explore their data, and could guide experimental design to elucidate the function of genes and their proteins, for many of which the function is not known.

6 CONCLUSION

We considered the problem of pattern set mining and propose BINAPs, a novel kind of binarized neural networks that are capable of discovering a compact set of interpretable patterns that together describe the data well. Overall, BINAPs show to reliable recover ground truth and show promising results on several large scale real world datasets. With their ability to learn patterns using continuous, gradient-based optimization, BINAPs paves the way towards exploratory studies of challenging data sets comprising hundreds of thousands of features that state-of-the-art algorithms for pattern set mining fail to process, while providing patterns in the interpretable language of conjunctions over input symbols.

While BINAPs can reveal interesting structure in data and provide insights for human experts, it would make interesting future work to extend BINAPs for richer pattern languages, including for example patterns over mutual exclusive items, which recently have been proposed to mine. Such more expressive patterns can help to get an even deeper understanding of the data at hand. Besides, investigating deeper networks would make for engaging future work. Simply making the network larger however comes at the cost of reduced interpretability of the patterns, thus beating the purpose of their design, however more involved architectures could for example allow to extract pattern hierarchies from the data.

ACKNOWLEDGMENTS

Jonas Fischer is supported by a scholarship of the International Max Planck Research School for Computer Science (IMPRS-CS).

REFERENCES

[1] R. Agrawal, T. Imielinksi, and A. Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. ACM, 207–216.
[2] R. Bayardo. 1998. Efficiently mining long patterns from databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. 85–93.
[3] Y. Bengio, N. Léonard, and A. C. Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR* abs/1308.3432 (2013). arXiv:1308.3432 http://arxiv.org/abs/1308.3432
[4] T. Calders and B. Goethals. 2007. Non-derivable itemset mining. *Data Mining and Knowledge Discovery* 14, 1 (2007), 171–206.
[5] M. Courbariaux, Y. Bengio, and J.-P. David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems (NeurIPS)* 28 (2015), 3123–3131.
[6] S. Dalleiger and J. Vreeken. 2020. Explainable Data Decompositions. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (2020), 3709–3716.
[7] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li. 2018. GXNOR-Net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks* 100 (2018), 49 – 58.
[8] J. Fischer and J. Vreeken. 2019. Sets of Robust Rules, and How to Find Them. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. 38–54.
[9] J. Fischer and J. Vreeken. 2020. Discovering Succinct Pattern Sets Expressing Co-Occurrence and Mutual Exclusivity. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 813–823.
[10] M. R. Garey and D. S. Johnson. 2000. *Computers and intractability : a guide to the theory of NP-completeness* (22. pr. ed.). Freeman.
[11] F. Geerts, B. Goethals, and T. Mielikäinen. 2004. Tiling Databases. In *Proceedings of Discovery Science*. 278–289.
[12] K. Geurts, G. Wets, T. Brijs, and K. Vanhoof. 2003. Profiling High Frequency Accident Locations Using Association Rules. In *Proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA)*. 18pp.
[13] J. Han, J. Pei, and Y. Yin. 2000. Mining frequent patterns without candidate generation. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. ACM, 1–12.
[14] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. 2016. Binarized neural networks. *Advances in neural information processing systems (NeurIPS)* 29 (2016), 4107–4115.
[15] D. P. Kingma and J. Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.
[16] D. P. Kingma and M. Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*.
[17] M. Kramer. 1991. Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal* 37 (1991), 233–243.
[18] F. Li and B. Liu. 2016. Ternary Weight Networks. *CoRR* abs/1605.04711 (2016).
[19] M. Mampaey, J. Vreeken, and N. Tatti. 2012. Summarizing Data Succinctly with the Most Informative Itemsets. *ACM Transactions on Knowledge Discovery from Data* 6 (2012), 1–44. Issue 4.
[20] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. 2011. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In *Artificial Neural Networks and Machine Learning (ICANN)*. 52–59.
[21] P. Miettinen. 2010. Sparse Boolean Matrix Factorizations. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 935–940.
[22] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. 2008. The Discrete Basis Problem. *IEEE Transactions on Knowledge and Data Engineering* 20, 10 (2008), 1348–1362.
[23] P. Miettinen and J. Vreeken. 2011. Model Order Selection for Boolean Matrix Factorization. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 51–59.
[24] F. Moerchen, M. Thies, and A. Ultsch. 2011. Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. *Knowledge and Information Systems* 29, 1 (2011), 55–80.
[25] S. Myllykangas, J. Himberg, T. Böhling, B. Nagy, J. Hollmén, and S. Knuutila. 2006. DNA copy number amplification profiling of human neoplasms. *Oncogene* 25, 55 (2006), 7324–7332.
[26] L. D. Raedt and A. Zimmermann. 2007. Constraint-Based Pattern Set Mining. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 237–248.
[27] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. 2016. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision (ECCV)*. 525–542.
[28] D. Saad and E. Marom. 1990. Training Feed Forward Nets with Binary Weights via a Modified CHIR Algorithm. *Complex Systems* 4, 5 (1990).
[29] Y. Sakamoto, H. Inoue, P. Keshavarz, K. Miyawaki, Y. Yamaguchi, M. Moritani, K. Kunika, N. Nakamura, T. Yoshikawa, N. Yasui, H. Shiota, T. Tanahashi, and M. Itakura. 2007. SNPs in the KCNJ11-ABCC8 gene locus are associated with type 2 diabetes and blood pressure levels in the Japanese population. *Journal of Human Genetics* 52, 10 (2007), 781–793.
[30] T. Simons and D.-J. Lee. 2019. A Review of Binarized Neural Networks. *Electronics* 8, 6 (2019).
[31] K. Smets and J. Vreeken. 2012. SLIM: Directly Mining Descriptive Patterns. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 236–247.
[32] The 1000 Genomes Project Consortium. 2015. A global reference for human genetic variation. *Nature* 526, 7571 (2015), 68–74.
[33] J. Vreeken and N. Tatti. 2014. *Interesting Patterns*. Springer, 105–134.
[34] J. Vreeken, M. van Leeuwen, and A. Siebes. 2011. KRIMP: Mining Itemsets that Compress. *Data Mining and Knowledge Discovery* 23, 1 (2011), 169–214.

A APPENDIX

A.1 Training parameters

For Slim, we use the proposed default candidate ordering from their manuscript and provide a minimum support threshold of 10 for pruning. For Asso, we test $T = \{0.1, 0.2, \cdots, 1\}$, as suggested in their example experiment setup in the codebase. We implemented BinaPs in Pytorch, and for all experiments train the networks using Adam [15], with an initial learning rate of 0.01, and an adaptive learning rate schedule lowering the base learning rate to 0.001 and 0.0001 after epoch 5 and 7, respectively, and train for overall 10 epochs. The exception is *Instacart*, with relatively few features but comparatively many samples. There, we observe a saturation of loss in the third epoch and hence stop after that. As discussed in the main text, Asso is trained for rank selection testing k up to the number of features in the data, similarly we set the capacity c of BinaPs to the number of features. For *Kosarak* and *Genomes*, we provide the methods with $k = c = 1000$. For medium sized data with less than $20k$ features, such as the synthetic data and the *DNA* data set, we use a batch size of 64, for all other data we use a batch size of 32. In general, we recommend these as default values as this setup proved robust about the wide range of experiments we carried out. It is however easy to carry out a parameter grid search evaluating reconstruction loss on a test set. To extract pattern sets from the network, we binarize the weights at a threshold of .2.

A.2 BinaPs with small n

Here we provide a derivation of the claim that already a single pattern is likely ($> 23\%$) to co-occur with other patterns when planting 100 patterns in 1000 samples with a density of .05 uniformly at random over the transactions. For a patterns p and any other pattern q with marginal frequencies $n_p = 50$ and $n_q = 50$, we are interested in the minimum joint frequency n_{pq} such that the pattern occur statistically significant. To test the hypothesis assuming indpendence between patterns, we use Fisher's exact test $\mathcal{F}$, setting the significance threshold to $\alpha = 0.01$. Searching for the smallest joint frequency n_{pq} such that $\mathcal{F} < \alpha$, we obtain $n_{pq} \geq 8$, meaning that if the patterns co-occur in at least 8 samples their relation is likely to be statistically significant. The next question is how likely an event of p co-occuring with any of the other 99 pattern is, which are planted in the data set. Hence we compute this probability P using the hypergeometric distribution now taking into account the overall number of patterns, yielding

$$P = 99 * \sum_{i=8}^{50} \frac{\binom{50}{i}\binom{950}{50-i}}{\binom{1000}{50}} \approx 0.233.$$

Hence, the chance of observing even just one pattern co-occuring significantly with another pattern is larger than 23%.

A.3 Instacart data

We obtained the instacart dataset from the official Kaggle challenge[4] and merged food items of the same type (e.g. all sugar of different brands) each into a single item. This allows us to circumvent problems induced by the extreme sparsity of the database, where many items only occur extremely infrequent, even just once, and thus do not expose any statistical significant relationships, and to be able to find actual patterns such as e.g. *Milk* and *Cookie*, which would not be possible if we would consider all combinations of e.g. brands and types of chocolate cookies. Treating each transaction separately, independent of time and customer id, we obtain a dataset of 1236 food items appearing in 2704831 transactions.

[4] https://www.kaggle.com/c/instacart-market-basket-analysis/data

A.4 1000 Genomes data

We processed the variant calls of all individuals available in phase 3 of the 1000 Genomes project[5], filtering for autosomal single nucleotide variants (SNVs) with an allel frequency of at least .01. For all protein coding genes specified for the reference genome, we define windows from the transcription start site (TSS) to 1000 base pairs downstream of the TSS. We then filter for SNPs that appear in such a window, and define features in our binary matrix M for all cases where at least one of the alleles show the rare variant ("1|0", "0|1", "1|1"). Thus, the data matrix is of size 3·#filtered variants × #individuals. For each individual i, we set the data entry $M_{ij} = 1$ if the individual shows genotype j.

[5] ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/