Chapter 5
Interesting Patterns

Jilles Vreeken and Nikolaj Tatti

Abstract Pattern mining is one of the most important aspects of data mining. By far the most popular and well-known approach is frequent pattern mining. That is, to discover patterns that occur in many transactions. This approach has many virtues including monotonicity, which allows efficient discovery of all frequent patterns. Nevertheless, in practice frequent pattern mining rarely gives good results—the number of discovered patterns is typically gargantuan and they are heavily redundant.

Consequently, a lot of research effort has been invested toward improving the quality of the discovered patterns. In this chapter we will give an overview of the interestingness measures and other redundancy reduction techniques that have been proposed to this end.

In particular, we first present classic techniques such as closed and non-derivable itemsets that are used to prune unnecessary itemsets. We then discuss techniques for ranking patterns on how expected their score is under a null hypothesis—considering patterns that deviate from this expectation to be interesting. These models can either be static, as well as dynamic; we can iteratively update this model as we discover new patterns. More generally, we also give a brief overview on pattern set mining techniques, where we measure quality over a set of patterns, instead of individually. This setup gives us freedom to explicitly punish redundancy which leads to a more to-the-point results.

Keywords Pattern mining · Interestingness measures · Statistics · Ranking · Pattern set mining

J. Vreeken (✉)
Max-Planck Institute for Informatics and Saarland University,
Saarbrücken, Germany
e-mail: jilles@mpi-inf.mpg.de

N. Tatti
HIIT, Department of Information and Computer Science,
Aalto University, Helsinki, Finland
e-mail: nikolaj.tatti@aalto.fi

1 Introduction

Without a doubt, pattern mining is one of the most important concepts in data mining. In contrast to the traditional task of modeling data—where the goal is to describe all of the data with one model—patterns describe only *part* of the data [27]. Of course, many parts of the data, and hence many patterns, are not interesting at all. The goal of pattern mining is to discover only those that are.

Which brings us to one of the core problem of pattern mining, and the topic of this chapter: interestingness measures. Or, how to determine whether a given pattern is interesting, and how to efficiently mine the interesting patterns from a given dataset. In particular, we find many interesting research challenges in the combination of these two problems.

Before we go into this dual, there is a key problem we have to address first: interestingness is inherently subjective. That is, what is very interesting to one may be nothing but a useless result to another. This goes both between different analysts looking at the same data, but also between different data bases, as well as data mining tasks. As such, we know that our lunch will not be free: there is not a single general measure of interestingness that we can hope to formalize and will satisfy all. Instead, we will have to define task specific interestingness measures.

Foregoing any difficulties in defining a measure that correctly identifies what we find interesting, the second key problem is the exponentially large search space. That is, there are exponentially many *potentially* interesting patterns. Naively evaluating these one by one and only reporting those that meet the criteria is hence infeasible for all but the most trivial of pattern languages [3]. As such, in addition to correctly identifying what is interesting, ideally an interestingness measure also defines a structured, easily traversable search space to find these patterns.

A big breakthrough in this regard was made in 1994 with the discovery by Agrawal and Srikant, and independently by Mannila, Toivonen, and Verkamo [1, 44], that the frequency measure exhibits anti-monotonicity, a property frequently referred to as the A Priori principle. In practice, this property allows to prune very large parts of the search space, making it feasible to mine frequent patterns from very large databases. In subsequent years, many highly efficient algorithms to this end were proposed [78, 76, 26] (See also Chaps. 2 and 3).

Soon after the discovery of the A Priori principle people found that frequency is not a very good measure for interestingness. In particular, people ran into the so-called 'pattern explosion'. While for strict thresholds only patterns expressing common knowledge were discovered, for non-trivial thresholds the exponential space of patterns made that incredibly many patterns were returned as 'interesting'—many of which only variations of the same theme.

In years since, many interestingness measures have been proposed in the literature to tackle these problems; many for specialized tasks, pattern or data types, but we also find highly general frameworks that attempt to approximate the ideal (subjective) interestingness measure. In this chapter we aim to give an overview of the work done in these respects. We will discuss a broad range of interestingness measures, as well as how we can define efficient algorithms for extracting such patterns from data. In order to keep the chapter focused and succinct we will restrict ourselves to measures for unsupervised, or exploratory, pattern mining in binary data—by far the most well studied part of pattern mining. We do note up front, however, that many of the discussed measures and algorithms are highly general and applicable to other settings.

We will discuss the topic in three main parts, loosely following the development of the field over time. That is, in Sect. 2 we discuss relatively simple, absolute measures of interest—of which frequency is a well-known example. As we will see, applying these measures leads to problems in terms of redundancy, difficult parameterization, as well as returning trivial results. In Sect. 3 we discuss, on a relatively high level, the advanced approaches proposed aim to solve these problems. We discuss two of the main proponents in Sects. 4 and 5. In the former we go into detail on approaches that use statistical tests to select or rank patterns based on how significant they are with regard to background knowledge. In the latter we cover the relatively new approach of iterative pattern mining, or, dynamic ranking, where we iteratively update our background knowledge with the most informative patterns so far.

We note that despite our best efforts, we did not find an ideal taxonomy over all methods, as some methods exhibit aspects of more than one of these categories. In such instances we choose to discuss them in the category they fit most naturally, yet will identify alternate ways of looking at these papers. We identify open research challenges and round up with conclusions in Sect. 7.

2 Absolute Measures

In this section we discuss relatively straightforward measures of interestingness. In particular, we focus on what we call *absolute* measures. That is, measures that score patterns using only the data at hand, without contrasting their calculations over the data to any expectation using statistical tests.

More formally, in this section we consider a specific—and perhaps the most well-known—class of pattern mining problems, viz., theory mining [45]. In this setting, a pattern is defined as a description of an interesting subset of the database. Formally, this task has been described by Mannila and Toivonen [43] as follows.

Given a database D, a language $\mathcal{L}$ defining subsets of the data, and a selection predicate q that determines whether an element $\phi \in \mathcal{L}$ describes an interesting subset of D or not, the task is to find

$$T(\mathcal{L}, D, q) = \{\phi \in \mathcal{L} \mid q(D, \phi) \text{ is true}\}$$

That is, the task is to find all interesting subsets.

2.1 Frequent Itemsets

The best known instance of theory mining is frequent set mining [3]. The standard example for this is the analysis of shopping baskets in a supermarket. Let I be the set of items the store sells. The database D consists of a set of transactions in which each transaction t is a subset of I. The pattern language $\mathcal{L}$ consists of itemsets, i.e., again sets of items. The support count of an itemset X in D is defined as the number of transactions that contain X, i.e., $supp_D(X) = |\{t \in D \mid X \subseteq t\}|$. We write $fr_D(X)$ to denote the relative support of X in D, i.e., $fr_D(X) = supp_D(X)/|D|$. We do not write D wherever clear from context.

The 'interestingness' predicate is a threshold on the support of the itemsets, the minimal support: $minsup$. In other words, the task in frequent set mining is to compute

$$\{X \in \mathcal{L} \mid supp_D(X) \geq minsup\}$$

The itemsets in the result are called *frequent* itemsets.

Intuition The intuition behind this measure is simple: the more often an itemset occurs in the data, the more interesting it is.

Frequent itemset mining was originally not a goal on itself, but merely a necessary step in order to mine association rules [3]. There, the task is to discover rules $X \rightarrow Y$, where X and Y are itemsets with $X \cap Y = \emptyset$, such that when itemset X is a subset of a row $t \in D$, $X \subset t$, with high confidence we will also see itemset $Y \subset t$. Such rules express associations, possibly correlations, and can hence be useful in many applications. A main motivation was supermarket basket analysis, the idea being that by advertising X, people will also buy more of Y.

The basic strategy for mining association rules is to first mine frequent patterns, and then consider all partitionings of each frequent itemset Z into non-overlapping subsets X and Y, to form candidate rules $X \rightarrow Y$, while finally keeping only those association rules that satisfy some quality threshold [3]. Though an interesting research topic on itself, interestingness measures for association rules are beyond the scope of this chapter. We refer the interested reader to the recent survey by Tew et al. [69].

A Priori With a search space of $2^{|\mathcal{I}|}$ patterns, the naive approach of evaluating every pattern is infeasible. However, in 1994 it was discovered that support exhibits monotonicity. That is, for two itemsets X and Y, we know

$$X \subset Y \rightarrow supp(X) \geq supp(Y) \quad ,$$

which is known as the A Priori property [1, 44], and allows for efficient search for frequent itemsets over the lattice of all itemsets.

The A Priori algorithm was independently discovered by Agrawal and Srikant [1], and by Mannila, Toivonen, and Verkamo [44]. It is a so-called candidate test framework. Given a transaction database D over a set of items $\mathcal{I}$ and a support threshold minsup, it first determines the set of singleton *frequent* itemsets $\mathcal{F}_1 = \{i \in I \mid supp(i) \geq minsup\}$. Then, given a set $\mathcal{F}_k$ of frequent patterns of length k, we can construct the set $\mathcal{C}_{k+1}$ of candidate frequent patterns of length $k + 1$, by considering only itemsets that have all k sub-itemsets of length k included in $\mathcal{F}_k$. We then determine the supports of all candidates in one pass over the data, and obtain $\mathcal{F}_{k+1}$ by keeping only the candidates with $supp(X) \geq minsup$.

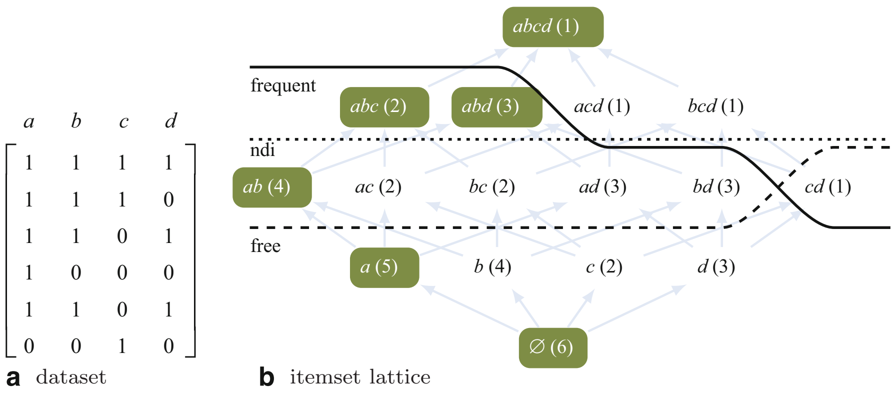

Fig. 5.1 A dataset of 4 items and 6 transactions and the corresponding lattice. The lattice shows the free, non-derivable, and, for $minsup = 2$, the frequent itemsets. Closed itemsets are *highlighted*

As an example, consider Fig. 5.1, where we depict a toy dataset and the lattice of all itemsets. Say we aim to mine all frequent itemsets with a minimal support of 2, i.e., a minimum frequency of $2/6 = 1/3$. The A Priori algorithm considers the lattice level-wise, and first identifies the frequent singleton itemsets. Here, a, b, c, and d are all frequent. It then constructs the candidate set by taking the Cartesian product with the frequent singletons. In this example this is the full set of itemsets of cardinality 2, i.e., $\mathcal{C}_2 = \{ab, ac, bc, ad, bd, cd\}$. We calculate the support of all candidates, and find that all itemsets, except cd, are frequent, i.e., $\mathcal{F}_2 = \{ab, ac, bc, ad, bd\}$. Iterating to the third level, we have $\mathcal{C}_3 = \{abc, abd\}$, as all other extensions of $\mathcal{F}_2$ contain cd, of which we know it is not frequent, and hence neither will any larger itemset containing it. We find that the two remaining candidates are frequent, $\mathcal{F}_3 = \mathcal{C}_3$. Finally, $\mathcal{C}_4 = \emptyset$ as there are no itemsets of size 4 that have all of their sub-itemsets of length 3 in $\mathcal{F}_3$. Hence, the answer to the stated problem, the complete set of frequent itemsets for $minsup = 2$, is hence $\mathcal{F} = \{a, b, c, ab, ac, bc, ad, bd, abc, abd\}$.

The A Priori, or, perhaps more aptly named, level-wise algorithm can be applied for any enumerable pattern language $\mathcal{L}$ and monotonic interestingness measure q. Soon after the discovery of the A Priori property, we see three major focal points for further research. In particular, a lot of attention was given to investigating more efficient algorithms for mining frequent itemsets [24] (see also Chaps. 2 and 3), methods that can mine frequent patterns from data types other than binary (see Chap. 11), and third, on further measures of interestingness (this chapter).

Pattern Explosion Now armed with the ability to mine frequent itemsets in practice, researchers quickly found that frequency is not quite the ideal interestingness measure. That is, we find that for high support thresholds only find patterns representing common knowledge are discovered. However, when we lower the threshold, we are typically quickly flooded with such enormous amounts of results that it becomes impossible to inspect or use them. Moreover, the result set is highly redundant: very many of the returned patterns are simply variations of each other. Combined, this problem is known as the pattern explosion, and stems from the interplay of using a monotonic interestingness measure, and asking for all frequent patterns.

We find many attempts in the literature aiming to solve the pattern explosion, roughly divided between three main approaches. The first is to attempt to *condense* the set of results. That is, we only want those patterns reported such that we can infer (to certain extend) the complete set of frequent patterns. These interestingness measures can hence also be regarded as extra constraints in addition to the frequency constraint.

Maximal Frequent Itemsets In this vein, Bayardo proposed to mine *maximal* frequent itemsets [6]: itemsets that are frequent and which cannot be extended without their support dropping below the threshold. In the lattice, this comes down to reporting only the longest frequent pattern in each branch. In our example, the set of maximal frequent itemsets for $minsup = 2$ is $\mathcal{F}_{max} = \{abc, abd\}$. Maximal frequent itemsets are a lossy representation of the set of frequent itemsets in that all frequent itemsets can be reconstructed, yet the individual frequencies are lost. While maximal itemsets can be useful when we are interested in long patterns, we should be aware that for very low support thresholds complete data records are returned—which beats the purpose of pattern mining. Maximal frequent itemsets can be mined efficiently using, e.g., the Max-Miner [6] and MAFIA [11] algorithms.

Closed Frequent Itemsets In contrast to maximal frequent itemsets, *closed* frequent itemsets [52] provide a lossless representation of the frequent itemsets, as both these itemsets and their frequencies can be reconstructed exactly. The definition of a closed frequent itemset is an itemset X that is frequent, $supp(X) \geq minsup$, and of which there exists no extension for which the support remains the same, i.e., there is no $Y \supsetneq X$ such that $supp(Y) = supp(X)$. Following this definition, in our example, the set of closed frequent itemsets consists of $\mathcal{F}_{closed} = \{a, ab, abc, abd\}$, which is smaller than the complete set of frequent itemsets, yet larger than for maximal itemsets. Efficient algorithms for mining closed frequent itemsets include Charm [77].

Given a set of closed frequent itemsets $\mathcal{F}_{closed}$, we can determine the support of any frequent itemset $X \in \mathcal{F}$ with ease. That is, for a given itemset X, we simply find the smallest superset $Y \in \mathcal{F}_{closed}$, with $X \subseteq Y$, and return the support of Y. If no such superset exists in $\mathcal{F}_{closed}$, X is not frequent. As such, we essentially derive the frequency of X using a very simple rule.

Free Frequent Itemsets Closed itemsets can be seen as the maximal itemsets having among the itemsets having the same support. Closely related are *free* sets [9], which are the *minimal* itemsets among the itemsets having the same support, that is, an itemset X is free if there is no $Y \subsetneq X$ such that $supp(X) = supp(Y)$. Each free itemset X has a unique closure Y, a closed itemset Y such that $X \subseteq Y$. However, a closed itemset may stem from many free itemsets. This means that free itemsets will always be a larger collection than closed itemsets. Free itemsets are handy since they form a monotonically downward closed collection, that is, all sub-itemsets of a free itemset are also free. In our example, frequent free itemsets are $\mathcal{F}_{free} = \{a, b, c, d\}$. Itemset *cd* is also free but it is not frequent.

Non-Derivable Itemsets Calders and Goethals [12] developed the notion of support derivability a step further, and proposed to mine *non-derivable* frequent itemsets. An itemset is said to be derivable if we can derive its support using inclusion/exclusion rules, i.e., the upper and lower bound of its support are equal. In our example, *abc* is a derivable itemset: since $supp(bc) = supp(c) = 2$ we know that whenever c appears, b appears as well. Hence, it follows that $supp(abc) = supp(ac) = 2$. In our example the set of non-derivable itemsets for $minsup = 2$ is $\mathcal{F}_{ndi} = \{a, b, c, ab, ac, bc\}$. Like free itemsets, non-derivable itemsets are also monotonically downward closed, which allows us to mine them efficiently.

In practice, for a given database and threshold the number of closed itemsets and non-derivable itemsets is typically comparable; how many exactly depends on the structure of the data. In both cases, for clean data, up to orders of magnitude fewer itemsets are returned than when mining frequent itemsets. However, if the data is noisy, it can be that no reduction can be obtained and we still find millions or billions of itemsets for non-trivial thresholds.

Margin-Closed and Robust Frequent Itemsets Moerchen et al. [50] hence argues to prune more aggressively, and to this end proposes to relax the requirement on maintaining frequencies exactly. That is, to mine *margin-closed* frequent itemsets; essentially reporting only those frequent itemsets for which the support deviates more than a certain amount compared to their subsets. A related, but different approach was recently proposed by Tatti and Moerchen [66], whom acknowledge the data at hand is just a sample; whether a given itemset is frequent, maximal, closed, or non-derivable may just be happenstance. To this end they propose to mine only those itemsets that exhibit a given property *robustly*, i.e., in many random subsamples of the data. For example, the idea is that in the more (sub)samples of the data we find a certain itemset to be closed, the more informative it is to report this particular itemset to the end-user. A happy coincidence of robustness is that the monotonicity of the chosen property propagates. That is, if the property is monotonic, for example, non-derivability or freeness, the robust version is also monotonic, and hence for those measures we can mine robust itemsets efficiently.

Sampling Frequent Itemsets We should stress that A Priori works for any monotonic measure, for example, the Jaccard-distance based measure Cohen et al. [14] propose,

$$supp(X)/|\{t \in D \mid X \cap t \neq \emptyset\}|,$$

the support of the itemset divided by the number of transactions that share at least one element with X. However, while monotonic, in practice A Priori is impractical for this measure: we cannot prune any singletons, and hence have $\mathcal{F}_1 = \mathcal{I}$, by which already at the first step we have to check *all* itemsets of size 2. To circumvent this problem, Cohen et al. first of all consider only itemsets of length 2, and, only calculate the actual score for a sample of the complete candidate set. However, because of the exponential search space, to avoid mining mostly itemsets with very low scores, one will have to be careful what distribution to sample from. Cohen et al. sample according to a hash-based score that estimates the correlation between two items. In theory this approach can be extended to itemsets of arbitrary sizes, but will require a non-trivial extension of this estimate.

A possible solution to this end may have been given by Boley et al. [7], whom proposed a framework that allows to directly sample itemsets proportional to *any* score based on frequency and/or cardinality of a pattern. However, the more different 'components' a score has, the more computationally expensive the pre-processing becomes. In a follow-up paper [8], the same authors refined the procedure and removed the need for this pre-processing by formalizing a coupling-from-the-past MCMC sampler.

Al Hassan and Zaki [4] proposed a different approach that allows for directly sampling the output space of any pattern miner. While the paper discusses patterns in graphs, the same techniques can be applied for mining itemsets. In follow-up work they discuss Origami [5], an approach to sample patterns that are representative, as well as orthogonal to earlier sampled patterns. By sampling patterns not just proportionally to a static distribution, but with regard to earlier sampled results, this process comes rather close to dynamic ranking, which we will discuss in more detail in Sect. 5.

2.2 Tiles

The next class of absolute interestingness measures we consider are not for itemsets, but for *tiles*. In tile mining we are particularly interested in the area a pattern covers in the data. That is, $\mathcal{L}$ consists of tiles $T = (X, Y)$ which are defined by both an *intention*, a subset of all items $X \subseteq \mathcal{I}$, as well as an *extension*, a subset of all rows $Y \subseteq \mathcal{R}$. We then use q to calculate the interestingness over the cells of D identified by $X \times Y$.

Large Tile Mining The most constrained variant of this task is to mine exact tiles, tiles for which in D we find only 1s, that meet a *minarea* threshold. That is, tile for which $\mathit{area}(T) = |X||Y| \geq \mathit{minarea}$. A maximal tile is then a tile T for which we cannot add an element to X or Y, and updating the vice-versa to maintain the all-1s constraint, without the $\mathit{area}(T)$ decreasing. Note that as *area* does not exhibit monotonicity, the level-wise algorithm cannot be applied.

Intuition Large areas of only 1s in D are interesting.

Geerts et al. [21], however, gave a set of constraints that can be used to mine large tiles efficiently in practice; essentially implementing the greedy algorithm for Set Cover [21]. It is interesting to note that every large tile is a closed frequent itemset, an observation Xiang et al. [74] used in their algorithm, first mining closed frequent itemsets and then pruning this set.

Noise and large tile mining do not go together well. That is, given a dataset in which there exists one large tile against an empty background, simply by flipping one 1 to 0 makes it that the complete tile will not be discovered; instead we will find two partitions. Every further flipped value will partition the tile more.

More in particular, it may not be realistic to expect a process to generate a tile full of ones. Instead, we may need to relax our requirement and look for *noisy*, or *dense* tiles instead of *exact* tiles.

Noisy Tile Mining A noisy tile is a tile T associated with a frequency of ones in the data, for which we write, slightly abusing notation,

$$fr(T) = \frac{|\{(i,j) \in (X \times Y) \mid D_{ij} = 1\}|}{|X||Y|} \quad .$$

An exact tile then is a special case, with $fr(T) = 1.0$. When mining noisy tiles we are interested in finding large areas in the data that contain many 1s, or possibly, many 0s.

Intuition The more uniform the values of D over the area identified by T, i.e., the more 1s resp. 0s we find, the more interesting the tile.

We find the problem of mining noisy tiles in many guises and embedded in many problem settings. Examples include dense itemset mining [60], dense tile mining [75], bi-clustering [55], and Boolean Matrix Factorization [49, 40], as well as fault-tolerant itemset mining [54].

Fault tolerant itemset mining for a large part follows the regular frequent itemset mining setting, with, however, the twist that we do not just calculate support over $t \in D$ for which $X \subseteq t$, but also those transactions that *nearly* but not exactly support t. The general approach is that, per itemset X, we are given a budget of ϵ 1s that we may use to maximize the fault-tolerant support of X [54]. Clearly, a fixed budget favors small itemsets, as there per row fewer items can be missing. Poernomo and Gopalkrishnan [56] gave an efficient algorithm for mining fault-tolerant itemsets where the budget is dependent on the cardinality of the itemset.

Seppänen and Mannila [60] generalized the problem of ϵ fault-tolerant itemset mining to *dense* itemset mining. That is, instead of using a fixed budget of flips, the proposed algorithms mine itemsets for which we there exist at least σ rows such that the density of 1s in the data projected over the itemset is at least δ.

In Boolean Matrix Factorization the goal is to find a low-rank approximation of the full data matrix. Optimizing, as well as approximating this problem is NP-hard [49], and hence the standard approach is to iteratively find good rank-1 approximations of the data, i.e., large noisy tiles with high frequency. The Asso algorithm does this by searching for tiles that exhibit high association between the rows and columns, and has been shown to efficient heuristic for finding large noisy tiles [49].

Frequent itemsets can be used to bootstrap the search for dense areas in the data. Xiang et al. [75] gave a fast heuristic for finding dense tiles that first mines closed itemsets, and then iteratively combines them until a density threshold is reached. We find a similar strategy in the PandA algorithm [40].

2.3 *Low Entropy Sets*

The final absolute measure for interestingness we discuss is *entropy*. Whereas many of the above measures put explicit importance on the associations between 1s in the data, by ignoring or penalizing 0s. This, however, ignores the fact that there may be interesting associations in the data between both the 1s and the 0s. Heikinheimo et al. [30] hence argue to put equal importance on both 0/1, and instead of mining frequent itemsets, propose to mine itemsets for which the counts of the contingency table are highly skewed. That is, for an itemset X we calculate the support of all of its $2^{|X|}$ instances, and calculate the entropy over these counts. The score is minimal (0) when only one instance occurs in the data, e.g., if for itemset $X = abc$ if we find $supp(abc = 110) = |D|$, while the score is maximal ($|X|$) when all instances have the same support.

Intuition An itemset X is interesting if the distribution of the data is highly skewed, i.e., either highly structured or very random.

Using this score, which exhibits monotonicity, we can use the level-wise algorithm to efficiently mine either *low entropy* sets, if one is interested in highly structured parts of the data, or to mine *high entropy* sets if one is interested in identifying the most random parts of the data. Mampaey [41] proposed to speed up the mining by using inclusion-exclusion, making use of the fact that in practice only a fraction of all $2^{|X|}$ possible instances of X occur in the data. The μ-Miner algorithm provides a speed-up of orders of magnitude compared to the level-wise algorithm.

3 Advanced Methods

Though each of the methods described above has nice properties, we find that in practice they do not perform as well as advertised. In general, we find that all absolute measures identify far too many results as interesting, with or without condensation. The key problem is redundancy. Absolute measures have no means of identifying whether the score for a pattern is expected, nor are they able to prune variants of patterns that identify single statistically significant concepts.

We identify three main lines of research aimed at tackling these problems, or in other words, aimed at identifying more interesting patterns. A common theme in these approaches is the reliance on statistical analysis. The main difference between these methods and the methods described in the previous section is that in order to rank patterns we impose a statistical model on our data, and measure how interesting are the patterns given that model.

We can divide the methods into three rough categories:

1. **Static pattern ranking**. Here we assume that we know a simple statistical model, derived from a simple background information. We assume that this model is well-understood, and any pattern that is well-explained by this model should be

discarded. Consequently, we are interested in patterns that the model considers very unlikely.

2. **Iterative pattern ranking**. While static pattern ranking addresses the problem of redundancy with respect to background knowledge, it does not explicitly address the problem of redundancy between patterns. We can approach this problem more directly with dynamic ranking: At the beginning we start with a simple model and find the most surprising pattern(s). Once this pattern is identified, we consider it 'known' and insert the pattern into our model, which updates our expectations—and repeat the process. As a result we get a sequence of patterns that are surprising and non-redundant with regard to the background knowledge and higher ranked patterns.
3. **Pattern set mining**. The methods in the above categories measure interestingness only per individual pattern. The third and last category we consider aims at identifying the best *set* of patterns, and hence propose an interestingness measure over *pattern sets*. As such, these measures directly punish redundancy—a pattern is only as good as its contribution to the set.

4 Static Background Models

In Sect. 2 we discussed absolute interestingness measures, which we can now say are essentially only based on counting. In this section we will cover slightly more advances measures. In particular, we will discuss measures that instead of relying just on absolute measurements, contrast these measurements with the *expected* measurement for that pattern. The basic intuition here is that the more strongly the observation deviates from the expectation, the more interesting the pattern is.

Clearly, there are many different ways to express such expectation. Most often these are calculated using on a probabilistic model of the data. Which model is appropriate depends on the background knowledge we have and/or the assumptions we are willing to make about the data. As such, in this section we will cover a wide range of different models that have been proposed to formalize such expectations.

However, in order to be able to identify whether a pattern is interesting, we need to be able whether the deviation between the observation and the expectation is large enough. That is, whether the deviation, and hence correspondingly the pattern, is significant or not. To this end we will discuss a variety of (statistical) tests that have been proposed to identify interesting patterns.

For clarity, we will start our discussion with the most simple model, the independence model. We will then use this model as an example to discuss a range of significance measures. We will then proceed to discuss more complex models, that can incorporate more background knowledge, for which many of these tests are also applicable. Interleaved we will also discuss interestingness measures specific to particular models and setups.

Before we start, there is one important observation to make. As opposed to the previous section, the measures we will discuss here are typically not used to mine *all* interesting patterns. This is mostly due to that these measures are typically not monotonic—and hence do not easily allow for efficient search—as well as that it is often difficult to express a meaningful threshold (i.e., significance level). Instead, these measures are used to rank a given collection of patterns, e.g., mined all frequent patterns up to a certain support threshold, or, when practical bounds are available, to mine a top-*k* of the most significant patterns. Many of the authors of work we survey in this section argue that in practice analysts do not want, nor have time, to consider all patterns, and hence a small list of the most interesting patterns is preferable.

4.1 Independence Model

We start with the simplest background model, which is the model where we assume that the individual items are all independent. Under this assumption we expect the frequency of a given itemset $X = x_1 \cdots x_n$ to be equal to

$$ind(X) = \prod_{i=1}^{n} fr(x_i) \quad .$$

The background knowledge we use are simply the frequencies of the individual items, which can be straightforwardly computed from the data. Moreover, it seems reasonable to expect to expect the data analyst (e.g., store manager) to know these margins (e.g., how often each product is sold) and hence be able to make such inferences intuitively. As such, the independence model is expected to correctly identify 'boring' patterns, patterns for which the frequencies follow under the independence model.

Testing Observations against Expectations Now that we have a model, we will use it as an exemplar to discuss a range of widely used methods for comparing the observed measurement with the expectation. After covering these general methods, we will discuss more detailed models, and more specialized measures.

With the above, we can compute both the observed frequency $fr(X)$ and the expectation $ind(X)$ of the independence model. The next step is compare these two quantities. A straightforward way to do this is to consider their ratio, a measure known as *lift* [33], and formally defined as

$$lift(X) = fr(X)/ind(X).$$

Here we consider itemsets that have a high lift to be interesting, that is, itemsets whose observed support is substantially higher than the independence assumption. Hence, a larger ratio implies higher interestingness.

In our example, we have $fr(ab) = 0.66$, while under the independence model we have $ind(ab) = \frac{5\times4}{6\times6} = 0.55$. As such, we find $lift(ab) = 1.2$. For abc, on the other hand, we have $lift(abc) = 0.33/0.18 = 1.83$. While both patterns have a positive lift score, and are hence potentially interesting, the higher score for abc identifies this pattern as the most interesting of the two.

In this example the outcome follows our intuition, but in practice this is not always the case: lift is a rather ad-hoc score. This is due to it comparing the two absolute values directly, without taking into account how likely these values, or their ratio is, given the background model.

We can, however, also compare the deviation by performing a proper statistical test. In order to do so, note that according the independence model the probability of generating a transaction containing an itemset X is equal to $ind(X)$. Assume that our dataset contains N transactions, and let Z be a random variable stating in how many transactions X occurs. The probability that $Z = M$ is equal to binomial distribution,

$$p(Z = M) = \binom{N}{M} q^M (1-q)^{N-M}, \quad \text{where} \quad q = ind(X) \quad .$$

Now that we have this probability, we can perform a one-sided statistical test by computing the probability that we observe a support of $fr(X)$ or higher, $p(Z \geq Nfr(X))$. Note that the larger frX, the smaller the p-value is.

Computing the right-hand side amounts to computing a sum of probabilities $p(Z = M)$, which as there are $2^|Z|$ possible values for M, may prove to be restrictively slow in practice. However, as exactly in those cases binomial distributions are accurately approximated by a normal distribution, we can perform an alternative test by considering a *normal approximation* of the binomial distribution. In this case, we can obtain the p-value by computing the tail of the normal distribution $N\left(Nq, \sqrt{Nq(1-q)}\right)$, where $q = ind(X)$. This is estimate is inaccurate if q is very close to 0 or 1 and N is small. One rule of thumb is that if $Nq > 5$ and $N(1-q)$, then this approximation is fairly accurate.

4.1.1 Beyond Frequency

So far, we only considered comparing the frequency of an itemset against its expected value. Clearly, we do not have to limit ourselves to only this measure (or, better, statistic).

Related to fault-tolerant itemsets we saw in Sect. 2, we can say that an itemset X is a *violation* of a transaction t if t does not contain X, $X \notin t$, yet t does contain some elements from X, $t \cap X \neq \emptyset$. We denote the fraction of transactions being violated by X as $v(X)$. The quantity $1 - v(X)$ is then a fraction of transactions that either contain (X) or do not contain any items from X. If items are highly correlated we expect $1 - v(X)$ to be high and $v(X)$ to be low.

Now, let q be the *expected* value of $v(X)$ based on the independence model. We can now calculate what Aggarwal and Yu call [2] the *collective strength* of a pattern as follows

$$cs(X) = \frac{1 - v(X)}{v(X)} \times \frac{q}{1-q} \quad .$$

In other words we compare the ratio of $\frac{1-v(X)}{v(X)}$ against the expected value.

4.1.2 Beyond Single Measurements

Instead of comparing just a single statistic, like support or the violation rate, we can consider much richer information. One example is to compare the complete contingency table of an itemset X with an expectation [10].

Assume we are given a distribution p over items in $X = x_1 \cdots X_M$. That is, a distribution over $2^{|X|}$ entries. For convenience, let us write

$$p(X = t) = p(x_1 = t_1, \ldots, x_M = t_M),$$

where t is a binary vector of length M. We now consider two different distributions: the first is the *empirical distribution* computed from the dataset,

$$p_{emp}(X = t) = \frac{|\{u \in D \mid u_X = t\}|}{|D|} \quad ,$$

and the second, p_{ind}, is the independence model,

$$p_{ind}(X = t) = \prod_{i=1}^{M} p_{ind}(x_i = t_i) \quad ,$$

where the margins (item frequencies) are computed from the input data.

The standard way of comparing these two distributions is by doing a so-called G-test, which essentially is a log-likelihood ratio test,

$$2 \sum_{t \in D} \log p_{emp}(X = t_X) - 2 \sum_{t \in D} \log p_{ind}(X = t_X) \quad .$$

Under the assumption that the items of X are distributed independently (which we here do), this quantity approaches the χ^2 distribution with $2^{|X|} - 1 - |X|$ degrees of freedom. Interestingly, this quantity can also be seen as a (scaled) Kullback-Leibler divergence, $2|D|KL(p_{\text{emp}}||p_{\text{ind}})$.

Alternatively, we can also compare the two distributions with Pearson's χ^2 test,

$$|D| \sum_{t \in \{0,1\}^{|X|}} \frac{(p_{emp}(X = t) - p_{ind}(X = t)^2)}{p_{emp}(X = t)},$$

which has the same asymptotic behavior as the G-test.

Each of these tests can be used to determine the p-value, or likelihood, of a pattern under an assumed model. In practice these measurements are used to rank the patterns from most surprising (under the model) to least surprising, typically showing the user only the top-k of most surprising patterns.

Next we will now look into more elaborate models, which allow us to make more realistic assumptions about the data than complete independence.

4.2 Beyond Independence

While the independence model has many positive aspects, such as ease of computation, intuitive results, as well as interpretability, it is also fair to say it is overly simplistic: it is naive to assume all item occurrences are independent. In practice, we may want to take known interaction into account as background knowledge.

4.2.1 Partition Models

With the goal of mining interesting associations, Webb [73] discusses 6 principles for identifying itemsets that are *unlikely* to be interesting, and to this end proposes to check whether the frequency of an itemset X can either be closely determined by assuming independence between any of its partitions, or by the frequency of any of the supersets of X.

The so-called *partition* model which is needed to perform these tests is a natural generalization from the independence model. More specifically, if we are given an itemset X, consider a partition $\mathcal{P} = P_1, \ldots, P_M$ of X, with $\bigcup_{i=1}^{M} P_i = X$, and $P_i \cap P_j = \emptyset$ for $i \neq j$. Under this model, we expect the support of an itemset to be equal to the product of the frequencies of its parts, i.e., $\prod_{i=1}^{M} fr(P_i)$. It is easy to see that for the maximal partition, when the partition contains only blocks of size 1, the model becomes equal to the independence model.

We can now compare the expected values and the observations in the same way we compared when were dealing with the independence model. If the partition contains only 2 blocks, $M = 2$, we can use Fisher's exact test [19]. While not monotonic, Hamalainen [25] recently gave a practical bound that allows to prune large parts of the search space.

To use the partition model we need to choose a partition. To do so, we can either construct a global model, i.e., choose a fixed partition of $\mathcal{I}$, or we can construct a local model in which the actual partition depends on the itemset X. As an example of the latter case we can consider find the partition of size 2 that best fits the observed frequency [72].

4.2.2 Bayesian Networks

Another natural extension of the independence model are *Bayesian networks*, where dependencies between items are expressed by a directed acyclic graph. In general, computing an expected support from a global Bayesian network is NP-hard problem, however it is possible to use the network structure to your advantage [15]. Additional speed-ups are possible if we rank itemsets in one batch which allows us to use share some computations [34].

Clearly, the partition model is mostly a practical choice with regard to computability and allowing the Fisher test; it does not allow us to incorporate much more knowledge than the independence model. Bayesian networks are very powerful, on the other hand, but also notoriously hard to infer from data. More importantly, they can be very hard to read. Unless we use very simple networks, it is possible our model can make inferences that are far from the intuition of the analyst, and hence prune itemsets that are potentially interesting. Next we discuss a class of models that can circumvent these problems.

4.3 Maximum Entropy Models

In general, for any knowledge that we may have about the data, there are potentially infinitely many possible distributions that we can choose to test against: any distribution that satisfies our background knowledge goes. Clearly, however, not all of these are an equally good choice. For example, say that all we know about a certain row in the data is that it contains 10 ones out of a possible 100 items. Then, while not incorrect, a distribution that puts all probability mass on exactly one configuration (e.g., the first 10 items), and assigns probability 0 to all other configurations, does make a choice that intuitively seems unwarranted given what we know. This raises the question, how should we choose the distribution to test against?

The answer was given by Jaynes [35] who formulated the Maximum Entropy principle. Loosely speaking, the MaxEnt principle states that given some background knowledge, the best distribution is the one that (1) matches the background knowledge, and (2) is otherwise as random as possible. It is exactly this distribution that makes optimal use of the provided background knowledge, while making no further assumptions.

4.3.1 MaxEnt Models for Transactions

As an example, let us discuss the MaxEnt model for binary data, in which we can incorporate frequencies of itemsets as background knowledge. Formally, let K be the number of items, and let Ω be the space of all possible transactions, that is, $\Omega = \{0, 1\}^K$ is a set of binary vectors of length K. In order to compute the expected support of an itemset, we need to infer a distribution, say p, over Ω.

Our next step is to put some constraints on what type of distributions we consider. More formally, we assume that we are given a set of functions $S_1, \dots, S_M$, $S_i : \Omega\mathbb{R}$, accompanied with desired values θ_i. Now let us consider a specific set of distributions, namely

$$Q = \{p \mid E_p[S_i] = \theta_i,\ i = 1, \dots, M\},$$

where $E_p[S_i]$ is the expected value of S_i w.r.t. p,

$$E_p[S_i] = \sum_{\omega \in \Omega} p(\omega) S_i(\omega) \quad .$$

In other words, Q consists of distributions for which the average value of S_i is equal to θ_i.

Apart from border cases, Q will typically be very large and may even contain infinitely many distributions. We need to have one distribution, and hence our next step is to choose one from Q. We do this by the Maximum Entropy principle. Formally, we identify this distribution by

$$p^* = \arg\max_{p \in Q} - \sum_{\omega \in \Omega} p(\omega) \log p(\omega),$$

where the standard convention $0 \times \log 0 = 0$ is used.

Besides nice information-theoretical properties, the maximum entropy distribution also has many other interesting and practical properties. For instance, it has a very useful regular form [16].

First consider that for some $\omega \in \Omega$, *every* distribution $p \in Q$ has $p(\omega) = 0$. Since $p^* \in \Omega$, this immediately implies that $p^*(\omega) = 0$. Let us define Z be the set of such vectors $Z = \{\omega \in \Omega \mid p(\omega) \text{for all} p \in Q\}$. The probability of the remaining points $\Omega \setminus Z$ can be expressed as follows: there is a set of numbers $r_0, \ldots, r_M$, such that

$$p^*(\omega) = \exp\left(r_0 + \sum_{i=1}^{M} r_i S_i(\omega)\right) \quad \text{for} \quad \omega \in \Omega \setminus Z. \tag{5.1}$$

The coefficient r_0 acts as a normalization constant. This form is the well-known *log-linear* model.

Now that we have established the general form of the maximum entropy model, let us look at some special cases of background information.

Assume that we do not provide any constraints, then the maximum entropy distribution will be the uniform distribution, $p(\omega) = 1/|\Omega|$. Consider now that we limit ourselves by setting K constraints, one for each item, $S_i(\omega) = \omega_i$. Then, $E[S_i]$ is the ith column margin, and we can show by simple manipulation of Eq. 5.1 that p^* corresponds to the independence model.

Consider now the other extreme, where we provide $2^K - 1$ constraints, one for each non-empty itemset, by setting $S_X(\omega)$ to be 1 if ω contains X, and 0 otherwise. We can show using inclusion-exclusion tricks that there is *only one* distribution in Q. If the corresponding targets θ_X were computed from a dataset D, then p^* is equal to the empirical distribution p_{emp} computed from D, $p_{emp}(\omega) = |\{t \in D \mid t = \omega\}|/|D|$.

Consider now that we do not use all itemset constraints. Instead we have a partition P and our itemset constraints consists only of itemsets that are subsets of blocks of P. In this case, p^* will have independent items belonging to different blocks. In other words, p^* is a partition model. Another example of non-trivial itemset constraints is a set consisting of all singleton itemsets and itemsets of size 2 such that these itemsets form a tree when viewed as edges over the items. In such case, p^* corresponds to the Chow-Liu tree model [13], a special case of Bayesian network where items may have only one parent. As an example of constraints not related to itemsets, consider $T_k(t)$, being equal to 1 if and only if t contains k 1 s, and 0 otherwise [65]. In such case, ET_k is the probability that a random transaction has k 1s.

All the cases we describe above have either closed form or can be computed efficiently. In general we can infer the MaxEnt distribution using iterative approaches, such as iterative scaling or a gradient descent [16]. The computational bottleneck in these methods is checking the constraints, namely computing the mean ES, a procedure that may take $O(|\Omega|) = O(2^K)$ time. As such, solving the MaxEnt problem in general for a given set of itemset constraints is computationally infeasible [63]. However, let us recall that in this section we are computing the distribution only to rank itemsets. Hence, we can limit ourselves by considering constraints defined only on items in X, effectively ignoring any item outside X [64, 46, 53]. This effectively brings down the computational complexity down to a much more accessible $O(2^{|X|})$.

4.3.2 MaxEnt and Derivability

Once inferred, we can compare the expectation to the observed supports using the same techniques as we developed above for the independence model. Moreover, there exists an interesting connection between the MaxEnt model and derivability of the support of an itemset. More specifically, for a given itemset X, the MaxEnt model derived using *proper* subsets of X shares a connection with the concept of non-derivable itemsets. An itemset is derivable if and only if its frequency can be deduced from the frequencies of its subsets. This is only possible when any distribution $p \in Q$ produces the same expectation $E_p[S_X]$ as the observed support. This immediately implies that the expected support according to p_{emp} is exactly the same as observed support. In summary, if an itemset is derivable, then a MaxEnt model derived from its subsets will produce the same expectation as the observed support.

4.3.3 MaxEnt Models for Whole Databases

So far we have considered only models on individual transactions. Alternatively, we can consider models on whole datasets, that is, instead of assigning probabilities to individual transactions, we assign probability to whole datasets. The space on which distribution is defined is now $\Omega = \{0, 1\}^{N \times K}$, where K is the number of items and N is the number of transactions, that is, Ω contains all possible binary datasets of size $N \times K$. Note that under this model N is fixed along with K, where as in transaction-based model only K is fixed and we consider dataset to be N i.i.d. samples. That is, while above we considered the data to be a bag of i.i.d. samples, we here assume the *whole* dataset to be one single sample. As such, different from the setting above, here *which* rows support an itemset are also considered to be of interest—and hence, the background knowledge should not just contain patterns, but also their row-sets.

In other words, we can use *tiles* as constraints. Given a tile T, let us write $S_T(D)$ for the number of 1s in entries of D corresponding to T. The mean $E[S_T]$ is then the expected number of 1 s in T. Note that we can easily model column margins using tiles, simply by creating K tiles, ith tile containing ith column and every row. We can similarly create row margins. The maximum entropy model derived from both rows and margins is known as *Rasch model* [57].

Unlike with transaction-based MaxEnt model and itemsets as constraints, discovering the MaxEnt for a set tiles can be done in polynomial time. The reason for this is that tile constraints allow us to factorize the model into a product of individual cells, which in turns allows us to compute the expectations $E[S_T]$ efficiently.

We can use the Rasch model to rank tiles based on the likelihood, that is, the probability that a random dataset will obtain the same values in T as the original dataset. We can show that this is equal to

$$\prod_{(i,j)\in T} p_{emp}(R_{ij} = D_{ij}),$$

where D_{ij} is the (i, j)th entry of the input dataset and R is the variable representing (i, j)th entry in a dataset. The smaller the probability, the more surprising is the tile according to the Rasch model. Unfortunately, this measure is monotonically decreasing. Consequently, the most interesting tile will contain the whole dataset. In order to remedy this problem, Kontonasios and De Bie [37] propose an normalization approach inspired by the Minimum Description Length principle [58], dividing the log-likelihood of the tile by its description length, roughly equal to the size of the transaction set plus the size of the itemset.

4.4 Randomization Approaches

So far, we compute expected frequencies by explicitly inferring the underlying distribution. However, we can avoid this by *sampling* datasets.

More formally, let Ω be the set of all possible binary datasets of size $N \times K$, $\Omega = \{0, 1\}^{N\times K}$. Many of these datasets are not realistic, for example Ω contains a dataset full of 1 s. Hence, we restrict our attention to datasets that have the same characteristics as the input dataset. One particular simple set of statistics is a set containing row and column margins. Assume that we have computed the number of 1s in each row and column. Let us write c_i for the number of 1 s in ith column, and r_j, the number of 1 s in jth row. Now consider, Ω' to be a subset Ω containing only the datasets that have the column margins corresponding to $\{c_i\}$ and row margins corresponding to $\{r_j\}$. Consider a uniform distribution over Ω'. We can now use this distribution to compute the expected value of a pattern. Such a distribution is closely related to the Rasch models explained previously. However, there are some technical differences. Datasets sampled from Ω' are forced to have certain row and column margins exactly while with Rasch models row and column margins are only forced on average. This, however, comes at a cost.

The uniform distribution over Ω' is very complex and, unlike the Rasch model, cannot be used directly. To remedy this problem we will have to sample datasets. Sampling Ω' from scratch is very difficult, hence we will use a MCMC approach [23, 29]. Given a dataset D from Ω', we first sample two columns, i and j, and two rows x and y. If it happens that out of four values D_{ix}, D_{jx}, D_{iy}, and D_{jy}, two are

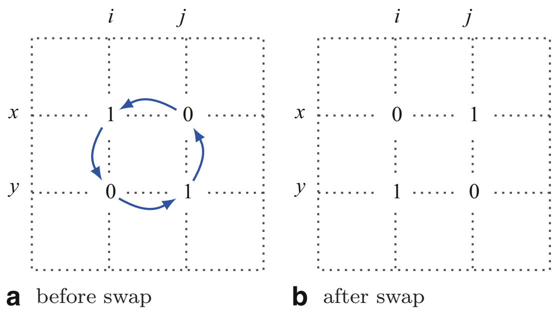

Fig. 5.2 An example of swap randomization

1 s and two are 0 s, and both ones are in opposite corners, see Fig. 5.2, then we swap D_{ix} with D_{iy} and D_{jx} with D_{jy}. It is easy to see that the new dataset will have the same row and column margins. By repeating this process many times, i.e., until the MCMC chain has converged, we can generate random datasets simply by starting the random walk from the input dataset. Note that in practice, however, we do not know when the MCMC chain has converged, and hence have to use a heuristic number of steps to reach a 'random' point in Ω'.

By sampling many random datasets, we can assess how significant a score obtained on the original data is in light of the maintained background knowledge by computing an empirical p-value—essentially the fraction of sampled datasets that produce higher support of an itemset X than the original support. The number of sampled datasets hence determines the resolution of the p-value.

In short, the (swap-)randomization and MaxEnt modelling approaches are very related. The former can be used to sample data that maintains the background knowledge exactly, while in the latter information is only maintained on expectation. The latter has the advantage that exact probabilities can be calculated. By sampling random data, whether by randomization or from a MaxEnt model, we can obtain empirical p-values—also for cases where by the nature of the score we're looking at (e.g., clustering error, classification accuracy, etc), it is impossible, or unknown, how to calculate exact values given a probabilistic model.

5 Dynamic Background Models

So far, we have covered only *static* scores. While within this class method have been proposed that are increasingly good at correctly identifying uninteresting patterns, they can only do so for individual patterns and with regard to a *static* background model. As such, when regarding the top-k result, we may still find patterns that are mostly variants of the same (significant) theme. In this chapter we turn our attention to models that explicitly take relationships between patterns into account.

Note, however, that some of the static models already do this to some extend. For example, the partition model studies subsets of the itemset under investigation. They do not, however, necessarily take all higher ranked itemsets into account.

In general, we can divide the dynamic approach into two categories. The first category is iterative pattern mining, where the goal is to greedily build a sequence of patterns, each pattern being the most surprising given the previous patterns and background knowledge. This is the class we discuss in this chapter. The second category is pattern set mining. There the goal is to produce a *set of patterns* that together optimize a given score over the whole set, as opposed to scoring patterns only individually. We will discuss pattern set mining in the next chapter.

Both categories have many technical similarities and share algorithmic approaches. In fact, some methods are difficult to pigeonhole as they can perform both tasks. The main difference we identify is that in pattern set mining we are looking for a set of patterns, that is, we need to control the number of patterns, whereas in iterative pattern ranking, we are 'simply' ranking patterns.

5.1 The General Idea

The main ingredient needed to perform iterative pattern mining, as opposed to static ranking, is a model that we can update. In particular, we need a model that can incorporate background knowledge in the same shape as what we're mining: patterns.

As such, the general approach here is that in the first iteration we infer the model p_1 according to the basic background knowledge $\mathcal{B}_1$ we may have about the data. We then rank all patterns accordingly, and select the top-k best scoring/most interesting patterns, $\mathcal{X}_1 = \{X_1, \ldots, X_k\}$. We assume the analyst will investigate these in detail, and hence that now onward we may regard these patterns *and* what can be derived from them as 'known'. As such, we update our background knowledge with $\mathcal{X}_1$, and hence for iteration 2 have $\mathcal{B}_2 = \mathcal{B}_1 \cup \mathcal{X}_1$, for which we infer model p_2. We then rank all patterns accordingly, etc, until we're done.

Next we will discuss three methods that allow for dynamic ranking.

5.2 Maximum Entropy Models

Maximum Entropy models, which we've met in the previous section, provide a natural way of constructing a probabilistic model from a given set of itemsets and their frequencies: essentially, each itemset is a constraint. As the technical details of how to infer a model under such constraints are beyond the scope of this chapter, and we refer the interested reader to, for example, Pavlov et al. [53].

Given a model that can incorporate itemsets and frequencies as background knowledge, we need to define a score for ranking candidates. We can use the statistical tests from Sect. 4, but a more intuitive approach is to use the likelihood of the data under the model. That is, the typical goal in dynamic ranking is to find a ranking that of which the top-k is the best explanation of the data in k terms.

To this end we construct a score in a post-hoc fashion. That is, we score a candidate on how much information we would gain *if* we would include it in the background knowledge. We do this as follows.

In general, given a set of itemsets $\mathcal{F}$, and a set of target frequencies θ_X for every $X \in \mathcal{F}$ as background knowledge, we can construct a MaxEnt model p^* such that $E[S_X] = \theta_X$ for every $X \in \mathcal{F}$. In turn, we can use the likelihood $p^*(D \mid \mathcal{F})$ to score the quality of $\mathcal{F}$. In other words, the better p^* can predict all frequencies the more likely is the data according to p^*, the better is the collection is $\mathcal{F}$. To be more precise, we know that $p^*(D \mid \mathcal{F} \cup \{Y\}) \geq p^*(D \mid \mathcal{F})$ and, as a special case, the equality holds whenever the observed frequency of Y is exactly equal to the expectation derived from $\mathcal{F}$. Moreover, the score increases as the observed frequency of Y becomes more distant from the expected value.

Given this likelihood score we can evaluate how informative a pattern Y is about the data in addition to our background knowledge. The question now is, how can find good rankings and pattern sets efficiently?

Wang and Parthasararthy [71] take a pre-mined collection of frequent itemsets as candidates, and consider these in level-wise batches. That is, they first consider the itemsets of size 1, then of size 2, and so on. Per batch they select all itemsets for which the predicted frequencies (L1 distance) deviates more than a given threshold, and add all of these to the background knowledge, after which they update the model and iterate to the next level. In order to make the ranking feasible, i.e., to get around the NP-hardness of inferring frequencies from the MaxEnt model, the authors sample frequencies instead of inferring them exactly.

Alternatively, Mampaey et al. [42] iteratively mine the top-most informative pattern, regardless of its cardinality. To do so efficiently, they propose an efficient convex bound which allows many candidate patterns to be pruned, as well as a method for more efficiently inferring the MaxEnt model using a quick inclusion/exclusion based approach. NP-hardness problems are here circumvented by partitioning the model, either explicitly such that only patterns of up to length k are allowed, or by allowing up to k overlapping itemsets per part.

5.3 Tile-based Techniques

While expressing redundancy with itemsets and generative models can be very complicated, as we have to somehow determine expected frequencies of itemsets given frequencies of other itemsets, tiles provide much more straightforward and natural ways of measuring redundancy. For instance, we can consider the overlap between tiles.

As a basic example of such problem, consider the large tile mining problem we encountered in Sect. 2, where the goal is to find all exact tiles covering at least *minarea* 1s. When we cast this in the dynamic ranking framework, we would want to find the sequence of tiles such that each top-k covers as many 1s as possible using k exact tiles. That is, every tile in the ranking has to cover as many uncovered 1s as possible: any 1s the k-th tile covers already covered by a tile of rank $\leq k$ are simply not counted—and hence redundancy is directly punished. Geerts et al. [21] call this the maximal tiling problem, and identify it as an instance of the set cover problem, which is known to be NP-hard [36].

For noisy tiles, overlap alone does not suffice to identify redundancy, as different tiles may explain areas of the data in more fine or coarse detail. We should therefore consider the *quality* of the tile, for example by punishing the noise, or favoring tiles that are surprising.

To this end we can re-use the Rasch model [18], now using it to discover surprising tile sets. Similar for ranking tiles based on area, we can also rank tilings by computing the likelihood of entries covered by the tile set [37]. Similarly, to rank individual tiles, we need to normalize this probability, as otherwise the best tiling is automatically one tile containing the whole dataset. Kontonasios and De Bie [37] do not explicitly update their model, but instead consider the case where the analyst would investigate every tile exactly. As such, the values (0 s and 1 s) of a processed tile can be assumed known, and hence the likelihoods of already covered cells set to 1. They further show this covering problem is an instance of Weighted Budgeted Maximum Set Cover, which is NP-hard, but for which the greedy approach is known to provide good solutions.

The MaxEnt model composed from tiles can be also used in a similar manner as the MaxEnt model from itemsets. For instance, given a set of tiles and their densities, that is, the fraction of 1 s inside each tile, we can construct the corresponding MaxEnt model and use the likelihood of the data as the goodness of the tile set [67]. Besides for ranking a set of candidate tiles, Tatti and Vreeken show that many (exploratory) data mining results on binary data can be translated into sets of noisy tiles. Hence, through the same machinery, we can dynamically rank results from different algorithms based on their relative informativeness [67].

Whereas the basic MaxEnt allows only frequencies of 1s within tiles as background knowledge, a recent paper by Kontonasios and De Bie [38], demonstrates how more complex information can be incorporated. Examples include frequencies of itemsets—unluckily, however, as we saw for the transaction based MaxEnt model, this does mean that inferring from the model becomes the NP-hard in general.

In the introduction we spoke about the impossibility of formalizing the inherently subjective notion of interestingness in general. Conceptually, however, dynamic ranking comes very close. As long as we can infer the Maximum Entropy model for the background knowledge an arbitrary user has, under the assumption that the user can optimally make all inferences from this knowledge, we know from information theory that our framework will correctly identify the most surprising result. De Bie [17] argues that this setup is one of the most promising for measuring subjective interestingness. The key challenges he identifies are in defining Maximum Entropy models for rich data and pattern types, as well as for efficiently mining those patterns that optimize the score.

5.4 Swap Randomization

As the last model we consider for dynamic ranking, we return to the swap randomization model we first described in Sect. 4. Recall that using this model we can sample random datasets of fixed row and column margins, and that we can use these samples to obtain empirical p-values.

We can extend this model by requiring that the sampled datasets must also have fixed frequencies for a certain given set of itemsets. Unsurprisingly, this makes sampling datasets very difficult, however. In fact, producing a a new dataset satisfying all the constraints is computationally intractable in general, even if we have original dataset at our disposal [29].

Instead of forcing hard constraints, we can relax these conditions and require that the probability of a random dataset R should decrease as the frequencies of given itemsets diverge from the target frequencies. Datasets that satisfy the given itemsets exactly will have the largest probability but other datasets are also possible. This relaxation allows us to use the same, well-understood, MCMC techniques as for standard swap randomization [29].

6 Pattern Sets

Pattern set mining is the fourth and final approach to discovering interesting patterns that we cover, and is also the most recent. It is strongly related to the dynamic modeling approach we met in the previous section, but has a slightly different twist and implications.

The general idea in pattern set mining is simple: instead of measuring the interestingness of each pattern X individually, i.e., through $q(X)$, we now define q over *sets of patterns* $\mathcal{X}$. That is, instead of evaluating a pattern X only locally, e.g., checking whether it describes significant local structure of the data, the goal is now defined globally. As such, we aim to find that set of patterns $\mathcal{X}$ that is optimal with regard to q. For example, we can now say we want to find that set of patterns that together describes the structure of the data best, i.e., that models the full joint distribution of the data best.

By measuring quality over sets instead of individual patterns, we face a combinatorial problem over an exponential space of candidate elements. That is, to find the optimal solution naively we would have to consider every possible subset out of the space of $2^{|\mathcal{I}|}$ possible patterns in the data. Sadly, none of the proposed quality measures for pattern set mining exhibit monotonicity, and hence we have no efficient strategy to obtain the optimal pattern set. Moreover, while for some measures we know that even approximating the optimum within any constant factor is NP-hard [49], for most measures the score landscape is so erratic we so far have no results at all on the complexity of the optimization problem.

In light of the search space and the difficulty of the problem, most pattern set mining methods employ heuristics. In particular, the locally optimal greedy strategy is a popular choice. This means that in practice, pattern set mining methods and dynamic modeling have a lot in common; although in iterative ranking there is no explicit global goal defined, in order to find a ranking, we iteratively greedily find the locally most informative pattern, and update the model. A key difference to pattern set mining is that here we explicitly take the complexity of the pattern set in check; we consider both the gain in quality, as well as the cost in complexity over the full set.

(Though pattern set mining is a combinatorial problem, and intuitively optimizing most instances seems very complex, so far theoretical hardness results have only been found for a handful of cases, including [49, 48, 75].)

6.1 Itemsets

Krimp is among the most well-known pattern set mining algorithms. Siebes et al. [62] define the best set of itemsets by the Minimum Description Length principle as the set that provides the best lossless compression. Each itemset is assigned a code word, the length of which depends on how frequently the itemset is used when greedily covering the data without overlap. The pattern set is then scored on the number of bits necessary to lossless describe the data. That is, the sum over the number of bits to encode the dictionary, the itemsets and their code words, and number of bits to encode the data using these code words.

The Krimp algorithm heuristically finds good sets by first mining frequent itemsets up to a given $minsup$ threshold, and greedily selecting from these in a fixed order. The resulting pattern sets are typically small (100 s) and have been shown to be useful in many data mining tasks [70] (see also Chap. 8). There exist a number of variants of Krimp for other data types, including for low-entropy sets [31]. Siebes and Kersten [61] investigated to structure functions, and proposed the Groei algorithm for approximating the optimal k-pattern set.

Alternative to a descriptive model, we can aim to find a good generative model. The Mini algorithm proposed by Gallo et al. [20] employs a probability distribution based on itemsets and frequencies, and aims finding the set of itemsets that predict the data best.

While intuitively appealing, using likelihood to score pattern sets, however, is not enough since the score increases w.r.t. the inclusion of pattern set, that is, the set containing all itemsets will have the highest likelihood. To control the size of the set we need to exert some control over the output set. For example, we can either ask the user to give a number of patterns k, or automatically control the number of the itemsets by BIC [59] or MDL [58], in which case the improvement of adding a new itemset into a result set must be significant.

The MaxEnt model employed by the mtv algorithm, which we met in the previous section, does not only lend itself for iterative ranking, but can also be straightforwardly used with either of test two model selection criteria [42]. In practice, (partly due to the partitioning constraints necessary to keep computation feasible) MTV typically finds pattern sets in the order of tens of patterns.

6.2 Tiles

The k maximal tiling problem, as defined by Geerts et al. [21] is perhaps the earliest example of pattern set mining. Xiang et al. [74, 75] expanded on the problem setting and aim to cover as many of the 1s of the data with k noisy tiles. Both problem settings are NP-hard, and are related to Set Cover. As such, the greedy strategy is known to find a solution within $O(\log n)$ of the optimum.

Boolean Matrix Factorization Strongly related to tiling is the problem of Boolean Matrix Factorization (BMF) [49]. The goal in matrix factorization is to find a low-rank approximation of the data. In case of BMF, each factor can be regarded as a noisy tile, and a factorization hence as a set of tiles. BMF is known to be NP-hard, as well as NP-hard to approximate [49]. The Asso algorithm is a heuristic for finding good k-factorizations, and can be coupled with an MDL strategy to automatically determine the best model order [47]. Lucchese et al. [40] gave the much faster PandA algorithm, which optimizes a more loose global objective that weighs the number of covered 1s with the number of 1s of the factors.

Geometric Tiles So far we have only considered unordered binary data in this chapter. When we fix the order of the rows and columns, however, for instance because the data is ordered spatially, it may only make sense to consider tiles that are consecutive under this ordering. This problem setting gives rise to interesting problem settings, as well as algorithmic solutions.

Gionis et al. [22] propose to mine dense *geometric* tiles that stand out from the background distribution, and to do so iteratively in order to construct a tree of tiles. To determine whether a tile is significant, they propose a simple MDL based score. Finding the optimal tile under this score is $O(n^2\ m^2)$ and hence infeasible for non-trivial data. To circumvent this problem, they propose a randomized approach. Tatti and Vreeken [68] recently proposed an improved algorithm that can find the optimal sub-tile in only $O(mn\min(\mathrm{m}, \mathrm{n}))$. The tile trees discovered by these methods typically contain in the order of 10 s to 100 s of tiles.

Gionis et al. [22] also showed that by the same strategy, by first applying spectral ordering, meaningful *combinatorial* tiles can be discovered from unordered binary data.

6.3 Swap Randomization

The final pattern set mining approach we consider is based on swap randomization. Lijffijt et al. [39] aim to find the smallest set of patterns that explains most about the data in terms of a global p-value. By employing the swap randomization framework, and in particular by using empirical p-values they can consider a rich range of patterns, including frequent itemsets as well as clusterings. The goal of finding the smallest set that statistically explains the data is NP-hard in its general form, and there exists no efficient algorithm with finite approximation ratio.

As many random datasets need to be sampled in order to determine significance this approach is not as computationally efficient as some of the methods covered above, however, it should be noted that the framework is highly general. In principle it can be applied to any data and pattern type for which a (swap-)randomization variant has been defined, which includes, among others, real-valued data [51], graphs [28], and sequences [32].

7 Conclusions

With the goal of mining interesting patterns we face two main problems: first, we need to be able to identify whether a pattern is potentially interesting, and second, we do not want the results to be redundant. Both these concepts are subjective, and hence there is no single correct answer to either of the two goals. Consequently, we provide an overview of myriad of different techniques for mining interesting patterns. These techniques range from classic reduction approaches, such as, closed itemsets and non-derivable itemsets to statistical methods, where we either are looking patterns that deviate most from the expectation or looking for a compact set that models the data well.

Unlike when mining frequent itemsets, for more advanced interestingness measures we rarely have monotonicity to our advantage. This means that we cannot prune the search space easily, and mining algorithms are hence significantly more complex. In particular algorithms for discovering pattern sets are often heuristic. Better understanding of these combinatorial problems and their score, for example, by providing theoretical guarantees, is both a promising and necessary line of work toward developing better and faster algorithms.

In this chapter we covered techniques meant only for binary data. In comparison, discovering and defining interesting patterns and pattern sets from other types of data, such as, sequences, graphs, or real-valued datasets is still strongly underdeveloped. Both in the definition of useful interestingness measures, as well as in the development of efficient algorithms for extracting such patterns directly from data there exist many opportunities for exciting future work.

Regardless of data type, the key idea for future work is developing algorithms for mining *small* and *non-redundant* sets of only the most interesting patterns. Or, to quote Toon Calders at the 2012 ECMLPKDD most-influential paper award: “please, please stop making new algorithms for mining *all* patterns”.

Acknowledgments Jilles Vreeken is supported by the Cluster of Excellence "Multimodal Computing and Interaction" within the Excellence Initiative of the German Federal Government. Nikolaj Tatti is supported by Academy of Finland grant 118653 (ALGODAN).

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, pages 487–499, 1994.
2. C. C. Aggarwal and P. S. Yu. A new framework for itemset generation. In *PODS*, pages 18–24. ACM, 1998.
3. R. Agrawal, T. Imielinksi, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216. ACM, 1993.
4. M. Al Hasan and M. J. Zaki. Output space sampling for graph patterns. *PVLDB*, 2(1):730–741, 2009.
5. M. Al Hasan, V. Chaoji, S. Salem, J. Besson, and M. J. Zaki. Origami: Mining representative orthogonal graph patterns. In *ICDM*, pages 153–162. IEEE, 2007.
6. R. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD*, pages 85–93, 1998.
7. M. Boley, C. Lucchese, D. Paurat, and T. Gärtner. Direct local pattern sampling by efficient two-step random procedures. In *KDD*, pages 582–590. ACM, 2011.
8. M. Boley, S. Moens, and T. Gärtner. Linear space direct pattern sampling using coupling from the past. In *KDD*, pages 69–77. ACM, 2012.
9. J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Disc.*, 7(1):5–22, 2003.
10. S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *SIGMOD*, pages 265–276. ACM, 1997.
11. D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu. MAFIA: A maximal frequent itemset algorithm. *IEEE TKDE*, 17(11):1490–1504, 2005.
12. T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *PKDD*, pages 74–85, 2002.
13. C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE TIT*, 14(3):462–467, 1968.
14. E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *IEEE TKDE*, 13(1):64–78, 2001.
15. R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. Probabilistic networks and expert systems. In *Statistics for Engineering and Information Science*. Springer-Verlag, 1999.
16. I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *Annals Prob.*, 3(1):146–158, 1975.
17. T. De Bie. An information theoretic framework for data mining. In *KDD*, pages 564–572. ACM, 2011.
18. T. De Bie. Maximum entropy models and subjective interestingness: an application to tiles in binary databases. *Data Min. Knowl. Disc.*, 23(3):407–446, 2011.
19. R. A. Fisher. On the interpretation of χ^2 from contingency tables, and the calculation of P. *J. R. Statist. Soc.*, 85(1):87–94, 1922.
20. A. Gallo, N. Cristianini, and T. De Bie. MINI: Mining informative non-redundant itemsets. In *ECML PKDD*, pages 438–445. Springer, 2007.
21. F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *DS*, pages 278–289, 2004.
22. A. Gionis, H. Mannila, and J. K. Seppänen. Geometric and combinatorial tiles in 0-1 data. In *PKDD*, pages 173–184. Springer, 2004.
23. A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *TKDD*, 1(3):167–176, 2007.
24. B. Goethals and M. Zaki. Frequent itemset mining dataset repository (FIMI). http://fimi.ua.ac.be/, 2004.

25. W. Hämäläinen. Kingfisher: an efficient algorithm for searching for both positive and negative dependency rules with statistical significance measures. *Knowl. Inf. Sys.*, 32(2):383–414, 2012.
26. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12. ACM, 2000.
27. D. Hand, N. Adams, and R. Bolton, editors. *Pattern Detection and Discovery*. Springer-Verlag, 2002.
28. S. Hanhijärvi, G. C. Garriga, and K. Puolamäki. Randomization techniques for graphs. In *SDM*, pages 780–791. SIAM, 2009.
29. S. Hanhijärvi, M. Ojala, N. Vuokko, K. Puolamäki, N. Tatti, and H. Mannila. Tell me something I don't know: randomization strategies for iterative data mining. In *KDD*, pages 379–388. ACM, 2009.
30. H. Heikinheimo, J. K. Seppänen, E. Hinkkanen, H. Mannila, and T. Mielikäinen. Finding low-entropy sets and trees from binary data. In *KDD*, pages 350–359, 2007.
31. H. Heikinheimo, J. Vreeken, A. Siebes, and H. Mannila. Low-entropy set selection. In *SDM*, pages 569–580, 2009.
32. A. Henelius, J. Korpela, and K. Puolamäki. Explaining interval sequences by randomization. In *ECML PKDD*, pages 337–352. Springer, 2013.
33. IBM. *IBM Intelligent Miner User's Guide, Version 1, Release 1*, 1996.
34. S. Jaroszewicz and D. A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. In *KDD*, pages 178–186. ACM, 2004.
35. E. Jaynes. On the rationale of maximum-entropy methods. *Proc. IEEE*, 70(9):939–952, 1982.
36. R. M. Karp. Reducibility among combinatorial problems. In *Proc. Compl. Comp. Comput.*, pages 85–103, New York, USA, 1972.
37. K.-N. Kontonasios and T. De Bie. An information-theoretic approach to finding noisy tiles in binary databases. In *SDM*, pages 153–164. SIAM, 2010.
38. K.-N. Kontonasios and T. De Bie. Formalizing complex prior information to quantify subjective interestingness of frequent pattern sets. In *IDA*, pages 161–171, 2012.
39. J. Lijffijt, P. Papapetrou, and K. Puolamäki. A statistical significance testing approach to mining the most informative set of patterns. *Data Min. Knowl. Disc.*, pages 1–26, 2012.
40. C. Lucchese, S. Orlando, and R. Perego. Mining top-k patterns from binary datasets in presence of noise. In *SDM*, pages 165–176, 2010.
41. M. Mampaey. Mining non-redundant information-theoretic dependencies between itemsets. In *DaWaK*, pages 130–141, 2010.
42. M. Mampaey, J. Vreeken, and N. Tatti. Summarizing data succinctly with the most informative itemsets. *TKDD*, 6:1–44, 2012.
43. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *KDD*, pages 189–194, 1996.
44. H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *KDD*, pages 181–192, 1994.
45. H. Mannila, H. Toivonen, and A. I. Verkamo. Levelwise search and borders of theories in knowledge discovery. *Data Min. Knowl. Disc.*, 1(3):241–258, 1997.
46. R. Meo. Theory of dependence values. *ACM Trans. Database Syst.*, 25(3):380–406, 2000.
47. P. Miettinen and J. Vreeken. Model order selection for Boolean matrix factorization. In *KDD*, pages 51–59. ACM, 2011.
48. P. Miettinen and J. Vreeken. MDL4BMF: Minimum description length for Boolean matrix factorization. Technical Report MPI-I-2012-5-001, Max Planck Institute for Informatics, 2012.
49. P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The discrete basis problem. *IEEE TKDE*, 20(10):1348–1362, 2008.
50. F. Moerchen, M. Thies, and A. Ultsch. Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. *Knowl. Inf. Sys.*, 29(1):55–80, 2011.
51. M. Ojala. Assessing data mining results on matrices with randomization. In *ICDM*, pages 959–964, 2010.

52. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416. ACM, 1999.
53. D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. *IEEE TKDE*, 15(6):1409–1421, 2003.
54. J. Pei, A. K. Tung, and J. Han. Fault-tolerant frequent pattern mining: Problems and challenges. *Data Min. Knowl. Disc.*, 1:42, 2001.
55. R. G. Pensa, C. Robardet, and J.-F. Boulicaut. A bi-clustering framework for categorical data. In *PKDD*, pages 643–650. Springer, 2005.
56. A. K. Poernomo and V. Gopalkrishnan. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *KDD*, pages 697–706, New York, NY, USA, 2009. ACM.
57. G. Rasch. *Probabilistic Models for Some Intelligence and Attainnment Tests*. Danmarks paedagogiske Institut, 1960.
58. J. Rissanen. Modeling by shortest data description. *Automatica*, 14(1):465–471, 1978.
59. G. Schwarz. Estimating the dimension of a model. *Annals Stat.*, 6(2):461–464, 1978.
60. J. K. Seppanen and H. Mannila. Dense itemsets. In *KDD*, pages 683–688, 2004.
61. A. Siebes and R. Kersten. A structure function for transaction data. In *SDM*, pages 558–569. SIAM, 2011.
62. A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *SDM*, pages 393–404. SIAM, 2006.
63. N. Tatti. Computational complexity of queries based on itemsets. *Inf. Process. Lett.*, 98(5):183–187, 2006.
64. N. Tatti. Maximum entropy based significance of itemsets. *Knowl. Inf. Sys.*, 17(1):57–77, 2008.
65. N. Tatti and M. Mampaey. Using background knowledge to rank itemsets. *Data Min. Knowl. Disc.*, 21(2):293–309, 2010.
66. N. Tatti and F. Moerchen. Finding robust itemsets under subsampling. In *ICDM*, pages 705–714. IEEE, 2011.
67. N. Tatti and J. Vreeken. Comparing apples and oranges - measuring differences between exploratory data mining results. *Data Min. Knowl. Disc.*, 25(2):173–207, 2012.
68. N. Tatti and J. Vreeken. Discovering descriptive tile trees by fast mining of optimal geometric subtiles. In *ECML PKDD*. Springer, 2012.
69. C. Tew, C. Giraud-Carrier, K. Tanner, and S. Burton. Behavior-based clustering and analysis of interestingness measures for association rule mining. *Data Min. Knowl. Disc.*, pages 1–42, 2013.
70. J. Vreeken, M. van Leeuwen, and A. Siebes. KRIMP: Mining itemsets that compress. *Data Min. Knowl. Disc.*, 23(1):169–214, 2011.
71. C. Wang and S. Parthasarathy. Summarizing itemset patterns using probabilistic models. In *KDD*, pages 730–735, 2006.
72. G. I. Webb. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *TKDD*, 4(1):1–20, 2010.
73. G. I. Webb. Filtered-top-k association discovery. *WIREs DMKD*, 1(3):183–192, 2011.
74. Y. Xiang, R. Jin, D. Fuhry, and F. F. Dragan. Succinct summarization of transactional databases: an overlapped hyperrectangle scheme. In *KDD*, pages 758–766, 2008.
75. Y. Xiang, R. Jin, D. Fuhry, and F. Dragan. Summarizing transactional databases with overlapped hyperrectangles. *Data Min. Knowl. Disc.*, 2010.
76. M. J. Zaki. Scalable algorithms for association mining. *IEEE TKDE*, 12(3):372–390, 2000.
77. M. J. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. In *SDM*, pages 457–473. SIAM, 2002.
78. M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *KDD*, Aug 1997.