Algorithms for Database Systems Seminar

May 29, 2015

Tell me what I need to know: Succinctly Summarizing Data with Itemsets

Written by

Sivaranjini Chithambaram

Supervised by

Prof. Peter Widmayer

Abstract

This is a report about the MTV algorithm, proposed in the paper “Tell me what I need to know: Succinctly Summarizing Data with Itemsets” written by Michael Mampaey, Nikolaj Tatti and Jilles Vreeken. The objective is to provide an insight into the topic of pattern mining, focusing on the probabilistic concepts used to develop the MTV algorithm.
Introduction and overview

A very interesting method of data mining is called pattern mining. This approach for data-analysis consists of finding existing patterns in a given dataset. It is a well-studied field and originates from analyzing a supermarket basket to examine customer behavior in terms of purchased products. A pattern in this context is understood as a set of items which is often sold together. Such a pattern can be directly derived from the data by computing the frequency of purchases where the corresponding items have been bought together.

The aim of the class of pattern mining algorithms we are looking at is to collect a set of patterns which exist in a dataset \( D \) such that the collection of patterns describes the most important information about the data \( D \).

To get an intuition for the problem we are trying solve let us take a look at an example: Consider a supermarket which sells the following 8 products: apples, balloons, champagne, candles, disposable dishes, eggs, flour and milk. The set of the 8 products is called \( A \). Now we have a pile of receipts from customers who have purchased items in this supermarket. The information contained on the receipts is represented in the dataset \( D \). By analyzing the receipts, we find out that eggs and flour are bought together with higher frequency, than the combination of eggs and champagne. These two subsets \( \{ \text{eggs, flour} \} \) and \( \{ \text{eggs, champagne} \} \) of \( A \) are patterns (or also called itemsets), where the latter occurs with lower frequency than the first one.

Typical pattern mining algorithms provide a ranking of such patterns by comparing them to a static model. Such methods measure the interestingness of a pattern \( P \) by measuring how much new information about \( D \) is gained by adding \( P \) to the collection. However the model is never updated with the information gained by newly added patterns. Therefore two similar patterns will score high ranks despite having redundant information and will both be added to the collection. This leads to the traditional problem of rediscovering patterns which contain similar information about the data over and over. As a consequence these static model methods result in a large collection of itemsets with high redundancy.

The method introduced in this paper however proposes a method which outputs a succinct and non-redundant collection of itemsets by taking into account what we already know about \( D \). The method introduces a dynamic approach which updates the model with knowledge gained by newly discovered patterns. This ensures that every newly added pattern only contains information about the data which is not described by previously discovered patterns.

This is not the first method which uses a dynamic approach. However, most of the other dynamic approaches require the user to set one or several parameters such as a maximum error threshold. The method proposed in this paper does not require any such parameters. Furthermore all other algorithms use a two-phase approach where the user has to
provide a collection of candidate patterns which must be completely mined and stored before actually running the program.

In this paper the Maximum Entropy Model is used to model the data. To grade the interestingness of an itemset the Bayesian Information Criterion is used. With these tools an algorithm for computing a summary of data iteratively is proposed. A summary of data is understood as a succinct and non-redundant collection of itemsets which contains the most important patterns found in the data.

**Motivating example**

Imagine you are in the supermarket which sells the 8 products from above and you see a person with a basket containing: flour, milk, eggs, balloons, champagne and candles. Intuitively you would assume this person is organizing a party. Now you observe the person adding disposable dishes to their basket. This is most likely not surprising to you, because you could predict that someone organizing a party might also need disposable dishes.

A bad representation of this data would be an itemset containing only flour, milk and eggs because these are items which are found in every other regular shopping cart. However the remaining items have a strong indication supporting our intuition that the person is organizing a party. The most indicative items in this example are the balloons, the candles, the disposable plates and the champagne. A possible representation of this data would be the itemset containing champagne, balloons, disposable dishes and candles. However if we already know, that balloons and champagne are purchased, we won’t be surprised that candles and disposable dishes are bought as well because these two items add only very little extra information to our prediction. A succinct representation of this basket may be an itemset containing champagne and balloons. Or in other words, knowing that balloons and champagne are purchased suffices to predict that a party is organized. The champagne indicates that adults are invited to the party, the balloons indicate towards a party and not just a fine dinner. And now it is easy to predict that the person might also buy other party items such as candles and disposable dishes. Therefore we may omit these items.

Note that in a later stage of analyzing the basket, where you are very curious to know what kind of party the person is organizing, you might be more interested in knowing that candles were bought, because this indicates that the party involves a cake. And the disposable dishes might indicate how many people are attending the party, which may be of interest in a later stage of analyzing the data. The method described in this paper follows precisely this principle of iterative knowledge discovery.
**Terminology**

- We note the set of all items as \( A = \{a_1, a_2, \ldots, a_n\} \).
- A transaction \( t \in \{0,1\}^n = T \) is a vector of size \( n \) where each component of the vector corresponds to an item \( a_i \), \( 1 \leq i \leq n \).
- The input for our method is a dataset \( D = \{t_1, t_2, \ldots, t_m\} \) which contains \( m = |D| \) transactions.
- By “pattern” or “itemset” we denote a subset \( X \) of \( A \) (\( X \subseteq A \)).
- A collection \( C \) is a set of itemsets \( C = \{X_1, X_2, \ldots, X_k\} \).
- Each itemset \( X \) is associated with a frequency \( fr(X) \), this indicates how frequent the items in \( X \) are bought together in combination. The frequency of an itemset can be computed from dataset \( D \).

Let us apply this to our supermarket example from earlier:

We have the total number of items available \( n = 8 \) and the set of items
\( A = \{\text{apples, balloons, champagne, candles, disposable dishes, eggs, flour, milk}\} \).

A transaction \( t \) corresponds to a customer’s receipt, where \( t \) is an 8-component binary vector. The vector \( t \) has the entry 0 for all items which were not bought by this customer and 1 for the items that were bought. The \( t^\text{th} \) component, for \( i = 1, \ldots, 8 \) of the vector tells us whether item \( a_i \) was bought or not. Note that we do not care about the quantity of the items.

If somebody bought milk, flour and eggs the corresponding transaction \( t_1 \) would be: \( t_1 = (0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1) \). Another transaction \( t_2 = (0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0) \) translates to a purchase containing balloons, champagne, candles and disposable dishes. The input dataset \( D \) is a set of such transactions.

Example itemsets are: \( X_1 = \{\text{eggs, flour}\} \) or \( X_2 = \{\text{eggs, champagne}\} \). Basically all subsets \( X \subseteq A \) are an itemset.

An example collection could then be the following set with \( k = 2 \):
\[ C_{\text{example}} = \{X_1, X_2\} = \{\{\text{eggs, flour}\}, \{\text{eggs, champagne}\}\} \].

From the pile of receipts or the corresponding dataset \( D \) we know that eggs and flour are bought together in combination more often than the combination of eggs and champagne. In other words, the itemset \( X_1 = \{\text{eggs, flour}\} \) has a higher frequency than the itemset \( X_2 = \{\text{eggs, champagne}\} \). We denote the frequency of an itemset \( X \) by \( fr(X) \). Possible numerical values for our example could be:
\[ fr(X_1) = 0.9 \geq 0.2 = fr(X_2) \].

**Basic objective**

Given a dataset \( D \), our aim is to find a collection \( C \) of itemsets which contains the frequency information of all itemsets in \( A \). More precisely \( C \) should...

\[ (*) \quad \text{...enable correct frequency prediction for all itemsets } X \in C \text{ and reliable frequency prediction of other itemsets.} \]

\[ (**) \quad \text{...be a succinct and non-redundant set of itemsets.} \]
To preserve non-redundancy, any subset $C' \subset C$ should provide a significantly different description of $D$. In other words for every $X \in C$, $X$ should have a surprising frequency with respect to $C \setminus X$.

To find the best collection $C$ in terms of our criteria mentioned above, we introduce a scoring system. We grade the candidate collections with a score. The goal is to find the collection with the best score.

Before moving on, let us quickly recollect what we want from our algorithm.

**WHAT WE WANT**
- Input: dataset $D = \{t_1, \ldots, t_m\}$, where $m = |D|
- Output: collection of itemsets $C = \{X_1, \ldots, X_k\}$, where $|C| = k$ and $C$ fulfills criteria (*) and (**).

### Maximum Entropy Model

To predict the frequency of an itemset we need a function which assigns a probability to an itemset. To assign a probability to each possible subset of the items in $A$ we need a probability distribution. To model the distribution the maximum entropy model (MEM) is used. An intuitive way to understand MEM is the following: For all itemsets which we already have in $C$ the probability distribution $p$ will predict the frequency of this itemset correctly. For all itemsets which are not added, it will assume the most random distribution. In other words, maximum entropy models make optimal use of provided information. They rely only on provided information and are fully unbiased otherwise.

We denote the $i^{th}$ component of a transaction $t$ as $t[i]$ for $i = 1, \ldots, n$ and $p(a_1 = t[1], \ldots, a_n = t[n])$ as $p(A = t)$.

The entropy of a distribution $p$ over $T = \{0,1\}^n$ is defined as:

$$H(p) = -\sum_{t \in T} p(A = t) \log p(A = t).$$

Now consider the set of distributions:

$$P = \{p|p(X_j = 1) = fr(X_j), j = 1, \ldots, k\}, where k = |C|.$$ 

This is a set of distributions which fulfill the property of estimating the frequency of every itemset $X \in C$ correctly. Among these $p \in P$ we are interested in $p^*$ which maximizes the entropy:

$$p^* = \arg \max_p \{H(p)|p \in P\}$$

With this maximum entropy distribution $p^*$ we can directly calculate estimations of frequencies of itemsets.
Approach

Among all candidate collections the best collection $C^*$ is the collection with the best score. We cannot iterate through all candidate collections to find $C^*$, since the search space is too large. To understand why the search space is too large, we take a closer look at the candidate collections. The set of potential candidate collections is the power set of all itemsets. The set of all itemsets is the power set of $A$. So, set of potential candidate collections is the power set of the power set of $A$ denoted as $\mathcal{P}(\mathcal{P}(A))$. Instead of searching for $C^* \in \mathcal{P}(\mathcal{P}(A))$ we go ahead by iteratively adding itemsets to $C$ and adapting our model $p^*$ in each round.

In every round we select the itemset $X$ which provides the highest amount of additional information about the dataset $D$ and add it to our output container $C$. Then we update our model and re-compute the probability distribution $p^*$ of our collection with the newly added itemset $X$. We keep adding new itemsets as long as the score of the collection with the new itemset is better.

**WHAT WE WANT**

Input: dataset $D = \{t_1, ..., t_m\}$, where $m = |D|$  
Output: collection of itemsets $C = \{X_1, ..., X_k\}$, where $|C| = k$

```plaintext
while (new score better than old score) {
    select next itemset $X_{next}$;
    $C = C \cup \{X_{next}\}$;
    compute $p^*$;
}
return $C$;
```

Bayesian Information Criterion

To measure how good our collection $C$ is we introduced a scoring system. Collections are associated with a score $s(C)$. We keep adding itemsets to $C$ as long as $s(C)$ gets better. If by adding an itemset the score gets worse, we don’t add the itemset and are finished. The scoring system proposed in this paper is based on the Bayesian Information Criterion (BIC).

The score of collection $C$ is defined as:

$$s(C) = -\log_2 p^*(D) + \frac{1}{2} |C| \log_2 |D|$$

The smaller the score $s(C)$, the better the model.

- The **first term** is the negative log-likelihood of the maximum entropy model. The log-likelihood measures the goodness of fit.
- The **second term** is a penalty for the number of itemsets: A higher number of itemsets leads to higher complexity. Therefore high complexity is penalized by the second term.
The best model is identified by high likelihood and low complexity. The key property of this scoring system is the following: Redundancy is automatically reduced because collections with redundant itemsets are penalized for being too complex without sufficiently improving the likelihood.

**WHAT WE WANT**

Input: $\text{dataset } D = \{t_1, \ldots, t_m\}, \text{where } m = |D|$
Output: collection of itemsets $C = \{X_1, \ldots, X_k\}, \text{where } |C| = k$

while $(s(C)\text{ decreases})$

- select next itemset $X_{\text{next}}$;
- $C = C \cup \{X_{\text{next}}\}$;
- compute $p^*$;
- compute $s(C)$;

return $C$;

The main advantage of using the BIC to implement a scoring system is that the BIC favors models that fit the data with few parameters.

As an alternative one might think of using the log-likelihood. This would be a bad choice because larger collections will provide more information and hence allow for better estimations than smaller collections. In that case we would end up with a large collection with high redundancy.

To prove that this scoring system indeed favors itemsets with low redundancy, we recall the following theorem and its proof from the paper we are discussing:

**Theorem**[1]: Let $C$ be a collection of itemsets and let $p^*$ be the corresponding maximum entropy model. Let $X \notin C$ be an itemset such that $f r(X) = p^*(X = 1)$. Then $s(C \cup \{X\}) > s(C)$.

Before proving the Theorem let us understand its meaning: If you take an itemset $X$ which is not in $C$, such that the frequency of this itemset is predicted correctly by our model $p^*$ then the score gets worse by adding this itemset to $C$. If $s(C \cup \{X\})$ is greater than $s(C)$, then the score of $C$ is better $C \cup \{X\}$, since we prefer lower scores.

**Proof:**

Let $C_1 := C \cup \{X\}$,

- $P_1 = \{p | p(X = 1) = fr(X) \forall X \in C_1\}$
- $p_1^* = \arg \max_p \{H(p) | p \in P_1\}$.

Because $C$ is a proper subset of $C_1$, $P_1 \subseteq P$ holds. This is obvious, because among all distributions $p \in P$ only those distributions are in $P_1$ which also fulfill the condition $fr(X) = p(X = 1)$ for the newly added $X$.

Now we look for the maximum entropy distribution in $P_1$ as well as in $P$. Since $P_1 \subseteq P$, the distribution $p_1^*$ in $P_1$ which has maximal entropy is also in $P$. Hence the maximum entropy model $p^* \in P$ has a greater or equal entropy than $p_1^*$.

So we get $H(p_1^*) \leq H(p^*)$. 
Now we show $H(p^1_1) \geq H(p^*)$ to obtain $H(p^1_1) = H(p^*)$:

The assumption $fr(X) = p^*(X = 1)$ in the theorem implies, that $p^* \in P_1$. From that we get that the entropy of $p^*_1$, which has maximal entropy in $P_1$ is greater or equal than the entropy of $p^*$ and we get $H(p^*_1) \geq H(p^*)$.

As desired we combine the two equations and obtain $H(p^*_1) = H(p^*)$.

Because of the uniqueness property of the maximum entropy [2], we have $p^* = p^*_1$. Now we know that the first term of the scores $s(C)$ and $s(C_1)$ are equal. We also know that the second term is larger in $s(C_1)$ because $C_1$ is a larger collection. This means that $s(C_1) > s(C)$, which concludes the proof. ■

So, indeed the scoring system above based on BIC, is a good way of grading candidate collections, because among collections with similar information value, it favors the one with lower redundancy.

Alternative criteria to end the loop: We ask the user for an input parameter $z$, which defines how big the output collection $C$ should be. We stop adding itemsets, once we have found the first $z$ itemsets.

**WHAT WE WANT**

**Input:** dataset $D = \{t_1, \ldots, t_m\}, \text{where } m = |D|, \text{ int } z$

**Output:** collection of itemsets $C = \{X_1, \ldots, X_k\}, \text{where } |C| = k \leq z$

while $(s(C) \text{ decreases and } |C| < z)$ {
    select next itemset $X_{\text{next}}$;
    $C = C \cup \{X_{\text{next}}\}$;
    compute $p^*$;
    compute $s(C)$;
}

return $C$;

We are left with two open questions:

1) How to find out which itemset to add next?

2) How to compute $p^*$?

**Selecting $X_{\text{next}}$ and computing the maximum entropy model**

Ideally we would like to select the itemset which reduces the BIC score most. This would require us to iterate through all possible itemsets. The set of all possible itemsets is the power set of $A$ which has $2^n$ elements. Then, for every $X \in \mathcal{P}(A)$ we would have to compute the maximum entropy model for $X \cup C$.

Instead, we proceed by introducing a heuristic to determine the next itemset $X_{\text{next}}$. We choose the itemset $X_{\text{next}}$ for which the term $|p^*(X_{\text{next}}) - fr(X_{\text{next}})|$ is maximized. I.e. we select the itemset for which the difference between the actual frequency and the prediction by our model is highest. The method FindBestItemset() [3] returns the itemset $X_{\text{next}}$ which has a frequency $fr(X_{\text{next}})$ that is most off, compared to the
estimated value by our model. We use this method as a black-box method; the details of FindBestItemset() can be found in the paper.

To compute the distribution \( p \) in \( P = \{ p|p(X = 1) = f r(X), \ \forall X \in C \} \) which has maximal entropy, we use a method called IterativeScaling()[4]. This algorithm converges to the maximum entropy distribution \( p^* \) and is discussed further in the paper. Again, we use this method as a black-box method.

**WHAT WE WANT**

Input: \( \{ t_1, ..., t_m \}, \text{where } m = |D|, \text{ int } z \)

Output: collection of itemsets \( C = \{ X_1, ..., X_k \}, \text{where } |C| = k \leq z \)

while (\( s(C) \) decreases and \( |C| < z \)) {
    \( X_{\text{next}} = \text{FindBestItemset()} \);
    \( C = C \cup \{ X_{\text{next}} \} \);
    \( p^* = \text{IterativeScaling()} \);
    compute \( s(C) \);
}

return \( C \);

Now we've also solved the last two mysteries and have a greedy algorithm. From an abstract perspective we can say, that our greedy algorithm keeps adding the most “surprising” information to the summary.

**MTV algorithm**

With a few more minor optimizations, we get the final algorithm. There is a very easy way to include background information of the users and thus prevent rediscovering what the user already knows. We simply instantiate \( C \) with a collection of itemsets \( B \), which represents the user's background information. Then, we look for the collection \( C' \) of size at most \( z \), such that \( s(B \cup C') = s(C) \) is minimal.

If we do not want to constrain the size of \( C \), we can simply set \( z = \infty \).

By adding the itemset \( X \) which reduces the BIC score the most, an incremental construction of the desired collection \( C \) is given. The algorithm terminates when the size of the collection reaches \( z \) or when the BIC score no longer decreases. This algorithm mines Maximally informaTiVe summaries and is called the MTV algorithm.

**MTV Algorithm**

Input: \( \{ t_1, ..., t_m \}, \text{where } m = |D|, \text{ int } z \), background knowledge itemsets in \( B \)

Output: collection of itemsets \( C = \{ X_1, ..., X_k \}, \text{where } |C| = k \leq z \)
Key aspects of the MTV algorithm:

- Each round of the while loop adds an itemset to the collection \( C \), which leads to an incremental construction of \( C \) and an iterative data-analysis procedure.
- By calling the method FindBestItemset(), the algorithm selects the itemset which contributes most effectively to the current stage of the process.
- Re-computing \( p^* \) in each iteration ensures that the model is updated with newly added itemsets \( X \) which is crucial for the intended dynamic approach.
- As a result of assuring that the BIC score \( s(C) \) is always lower than the old BIC score, the non-redundancy of the collection of itemsets is preserved.

**Potential Optimization**

The scoring system \( s(C) \) only penalizes \( C \) for the number of itemsets but it does not take into account the size of the itemsets. This means that a collection containing a large itemset is penalized for complexity only as much as an itemset containing just one item.

Taking a final look at our example from the beginning, let

\[ C_1 = \{ \{ \text{balloons, champagne} \} \} \] be a collection of itemsets with a corresponding maximum entropy distribution \( p_1^* \) and

\[ C_2 = \{ \{ \text{balloons, champagne, candles, disposable dishes} \} \} \] a collection of itemsets with corresponding maximum entropy distribution \( p_2^* \). Assume that \( \log_2 p_1^*(D) = \log_2 p_2^*(D) \).

By using this assumption and \( |C_1| = 1 = |C_2| \) we get:

\[
s(C_1) = -\log_2 p_1^*(D) + \frac{1}{2} |C_1| \log_2 |D| = -\log_2 p_2^*(D) + \frac{1}{2} |C_2| \log_2 |D| = s(C_2)
\]

Thus, in the current scoring system both collections are preferred equally. According to our assumption however, the extra two items candles and disposable dishes in \( C_2 \) do not provide any further information. Regarding succinctness we should prefer \( C_1 \) over \( C_2 \) because larger itemsets require more storage.

(Recall that you were not surprised that a person buying balloons and champagne adds candles and disposable dishes to his or her supermarket basket and it was enough to know about the balloons and champagne to predict that he or she is organizing a party).
Conclusion

The MTV algorithm successfully computes the collection of itemsets which describes the input data best with respect to succinctness and non-redundancy. The approach proposed does not require any parameters such as maximum error threshold or significance level. The BIC determines the best model. Alternatively it is possible to mine the top-z most interesting itemsets. All in all, the MTV algorithm fulfills the initially desired properties.

References


