Approximating Representations for Large Numerical Databases

Szymon Jaroszewicz[*] Marcin Korzeń [†]

Abstract

The paper introduces a notion of support for real-valued functions. It is shown how to approximate supports of a large class of functions based on supports of so called polynomial itemsets, which can efficiently be mined using an Apriori-style algorithm. An upper bound for the error of such an approximation can be reliably computed. The concept of an approximating representation was introduced, which extends the idea of concise representations to numerical data. It has been shown that many standard statistical modelling tasks such as nonlinear regression and least squares curve fitting can efficiently be solved using only the approximating representation, without accessing the original data at all. Since many of those methods traditionally require several passes over the data, our approach makes it possible to use such methods with huge datasets and data streams where several repeated scans are very costly or outright impossible.

1 Introduction and notation

Association rule mining [1] is primarily concerned with discrete data. The prevalent approach to numerical attributes is discretization, see for example [9]. Unfortunately discretization always leads to information loss, and has other problems such as difficulty in selecting appropriate interval widths. In [10] a definition of support capable of handling numerical data without discretization has been presented. Further work in this area includes [3] where rank methods have been used and [4] where the notion of support has been defined for polynomials.

This work significantly extends [4] by defining support of arbitrary real-valued functions. Furthermore, we show how to approximate support for a large class of functions using supports of so called polynomial itemsets introduced in [4], thus extending the notion of concise representations to numerical data. Approximation error can also be computed based on supports of polynomial itemsets.

We describe a modification of the well known Apriori algorithm [1] which is capable of finding a collection of frequent polynomial itemsets together with so called extended border, which is required to estimate approximation error.

We evaluate the performance and accuracy of our approach experimentally and show potential applications to nonlinear regression and curve fitting. We show that those tasks can often be expressed in terms of supports of functions on a given dataset, thus supports of a collection of polynomial itemsets can be used to build approximate models quickly without accessing the original dataset at all. All information needed to build the model is taken from the precomputed supports. The performance gain is especially visible when several models are built from a single dataset or when building a model requires several scans of the dataset.

Let D be a dataset with the set of attributes $H = \{X_1, \ldots, X_n\}$. Elements of D are called *transactions* following data-mining conventions. Sets of attributes will be denoted with uppercase letters I, J. We assume that sets of attributes are ordered by the indices of attributes, *e.g.* if $I = \{X_{i_1}, X_{i_2}, \ldots, X_{i_r}\}$, then $i_1 < i_2 < \ldots < i_r$. Vectors of variable values from the domain of a set of attributes I will be denoted with boldface letter $\mathbf{x} = (x_1, \ldots, x_r)$. We further define $\mathbf{0}_r = (0, \ldots, 0)$ as a sequence of r zeros. The index r will be omitted when clear from context.

Let $I = \{X_{i_1}, \ldots, X_{i_r}\} \subseteq H$ be a set of attributes and t a transaction. Define $t[I] = (t.X_{i_1}, \ldots, t.X_{i_r})$, where $t.X$ denotes the value of attribute X in t.

We will make heavy use of the so called *multi-index notation*, which makes it easy to concisely represent expressions involving several variables. Multi-indices will be denoted by lowercase boldface Greek letters $\boldsymbol{\alpha}, \boldsymbol{\beta}$. A *multi-index* $\boldsymbol{\alpha}$ is a sequence of integers $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$. We define the following expressions:

$$\begin{aligned}
|\boldsymbol{\alpha}| &= \alpha_1 + \alpha_2 + \ldots + \alpha_r \\
\boldsymbol{\alpha}! &= \alpha_1!\alpha_2!\cdots\alpha_r! \\
\mathbf{x}^{\boldsymbol{\alpha}} &= x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_r^{\alpha_r} \\
I^{\boldsymbol{\alpha}} &= X_{i_1}^{\alpha_1} X_{i_2}^{\alpha_2} \cdots X_{i_r}^{\alpha_r} \\
D_I^{\boldsymbol{\alpha}} &= \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial X_{i_1}^{\alpha_1} \partial X_{i_2}^{\alpha_2} \cdots \partial X_{i_r}^{\alpha_r}}
\end{aligned}$$

[*]National Telecommunications Institute, Warsaw, Poland, sj@cs.umb.edu

[†]Szczecin University of Technology, Poland, mkorzen@wi.ps.pl

We will now make a crucial assumption that all attributes in H are numerical (real-valued) and their values are always in the interval $[-1, 1]$. If the data does not meet this assumption, it is converted as follows: numerical attributes having values which fall outside of the $[-1, 1]$ range are scaled appropriately, *i.e.* an attribute X_i is multiplied by a positive constant $c_i = (\max_{t \in D} |t.X_i|)^{-1}$. An analysis of the influence of such scaling on concepts introduced later in the paper will be given in Section 6. Binary attributes are treated as real-valued attributes with domain $\{0, 1\}$. Categorical attributes are converted into a number of binary attributes (one binary attribute per category).

2 Support of real-valued functions, polynomial itemsets

In this section we will introduce key notions of the paper, the definition of support for arbitrary real valued functions and review the concept of polynomial itemsets from [4].

Definition 2.1. Let $I \subseteq H$ be a set of attributes, and let $f(I)$ be a function of attributes in I. We define *support* and *absolute support* of f in a dataset D respectively as

$$\begin{aligned} \operatorname{supp}_D(f) &= \sum_{t \in D} f(t[I]) \\ \operatorname{supp}_D(|f|) &= \sum_{t \in D} |f(t[I])|. \end{aligned}$$

□

It is easy to see that the following inequalities relating support with absolute support hold

$$\operatorname{supp}_D(f) \le |\operatorname{supp}_D(f)| \le \operatorname{supp}_D(|f|). \tag{2.1}$$

Definition 2.2. Let $I = \{X_{i_1}, \ldots, X_{i_r}\} \subseteq H$ be a set of attributes. A *polynomial itemset* is an expression of the form

$$I^{\boldsymbol{\alpha}} = X_{i_1}^{\alpha_1} X_{i_2}^{\alpha_2} \cdots X_{i_r}^{\alpha_r},$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_r)$ is a multi-index. I is called the *base* of the polynomial itemset and $\boldsymbol{\alpha}$ its exponent. Further, r is called the *length* of the polynomial itemset and $|\boldsymbol{\alpha}|$ its *degree.* □

Since a polynomial itemset is a real-valued function, the definitions of support and absolute support apply to it automatically.

Let $I^{\boldsymbol{\alpha}}, I^{\boldsymbol{\beta}}$, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_r)$, be two polynomial itemsets. We say that $I^{\boldsymbol{\alpha}}$ is a subset of $I^{\boldsymbol{\beta}}$, denoted $I^{\boldsymbol{\alpha}} \sqsubseteq I^{\boldsymbol{\beta}}$, if $\alpha_i \le \beta_i$ for all $i \in \{1, \ldots, r\}$. Similarly we say that $I^{\boldsymbol{\alpha}}$ is a strict subset of $I^{\boldsymbol{\beta}}$, denoted $I^{\boldsymbol{\alpha}} \sqsubset I^{\boldsymbol{\beta}}$ if in addition $\alpha_i < \beta_i$ for some $i \in \{1, \ldots, r\}$.

Notice that for a polynomial itemset $I^{\boldsymbol{\alpha}}$ we can form an equivalent itemset $J^{\boldsymbol{\beta}}$ based on any $H \supseteq J \supset I$. The coefficients in $\boldsymbol{\beta}$ corresponding to attributes not in I will be simply set to 0. For example a polynomial itemset $X_1^5 X_3^2$ based on $\{X_1, X_3\}$ can equivalently be written as $X_1^5 X_2^0 X_3^2 X_4^0$ based on $\{X_1, X_2, X_3, X_4\}$. This implies that the definition of polynomial itemset inclusion can be applied to arbitrary two polynomial itemsets regardless of their bases.

Neither support of polynomial itemsets, nor its absolute value are monotone w.r.t. inclusion. Fortunately, this property is true for absolute support.

Theorem 2.3. *Absolute support of polynomial itemsets is monotone w.r.t. inclusion, that is $I^{\boldsymbol{\alpha}} \sqsubseteq J^{\boldsymbol{\beta}}$ implies $\operatorname{supp}_D(I^{\boldsymbol{\alpha}}) \ge \operatorname{supp}_D(J^{\boldsymbol{\beta}})$.*

Proof. Take two polynomial itemsets $I^{\boldsymbol{\alpha}} \sqsubseteq J^{\boldsymbol{\beta}}$. It is easy to see that there is a polynomial itemset $I_2^{\boldsymbol{\alpha}_2}$ such that $J^{\boldsymbol{\beta}} = I^{\boldsymbol{\alpha}} \cdot I_2^{\boldsymbol{\alpha}_2}$. Recall that all attributes in H are in the range $[-1, 1]$, consequently, in every transaction t, the absolute value of $I_2^{\boldsymbol{\alpha}_2}$ does not exceed 1, and the absolute value of $J^{\boldsymbol{\beta}}$ is not greater than that of $I^{\boldsymbol{\alpha}}$. Summing over all $t \in D$ the result follows. ∎

Thus, to find all itemsets with absolute value of support greater than or equal to ε, we can first use an Apriori style algorithm (see below) to find all polynomial itemsets whose absolute support is greater than or equal to ε. Next we compute the support of all itemsets found, and prune those whose support's absolute value is below ε (see Inequality 2.1).

In [4], the following properties of absolute support were presented. If a table contains only binary attributes, the definitions of support and absolute support of polynomial itemsets both reduce to the standard definition of support. If all attributes are independent and uniformly distributed on $[0, 1]$, absolute support behaves analogously to standard support of independent binary attributes with distribution $\left(\frac{1}{2}, \frac{1}{2}\right)$. These analogies show that polynomial itemsets are a natural extension of standard definition of support [1]. An interpretation of absolute support of polynomial itemsets as the number of records in which the value of the itemset is close to its maximum was also given in [4].

The algorithm for finding all polynomial itemsets with given minimum absolute support is given in Figure 1. This is a modified version of the algorithm presented in [4]. The algorithm is similar to the well known Apriori algorithm [1], but has some important modifications. Itemsets are generated in the order of increasing degree, at each iteration, the degree of itemsets consid-

Input: dataset D with set of attributes H, minimum support threshold ε
Output: all frequent polynomial itemsets with absolute support $\geq \varepsilon$

```
1: $k \leftarrow 1$; $C_0 \leftarrow \{1\}$; $F_0 \leftarrow \{1\}$; $C_1 \leftarrow \{X_i^1 : X_i \in H\}$
2: loop
3:     compute absolute support of all itemsets in $C_k$
4:     $F_k \leftarrow \{I^{\boldsymbol{\alpha}} \in C_k : \mathrm{supp}_D(|I^{\boldsymbol{\alpha}}|) \geq \varepsilon\}$
5:     compute support of all itemsets in $F_k$
6:     $C_{k+1} \leftarrow \emptyset$
7:     for all $I^{\boldsymbol{\alpha}} = X_{i_1}^{\alpha_1} \dots X_{i_r}^{\alpha_r} \in F_k$ do
8:         for all $j \geq i_r$ do
9:             $C_{k+1} \leftarrow C_{k+1} \cup \{I^{\boldsymbol{\alpha}} \cdot X_j\}$
10:        end for
11:    end for
12:    $k \leftarrow k + 1$
13: end loop
```

Figure 1: The `PolyApriori` algorithm

ered is increased by one. Support counting in steps 3, 5 is done by a simple database scan.

Candidate generation is done in steps 7 to 11. Note step 8, which increases the exponent of the last attribute in the itemset or adds a new attribute with exponent 1. Unlike the Apriori candidate generation, an attribute can be added to the same itemset more than once to allow for higher exponents.

Another difference is that instead of combining two frequent itemsets to produce a new candidate we simply add single attributes to them, and we don't prune itemsets with infrequent subsets. We thus compute absolute support for a larger collection of itemsets than it first seems necessary. However, as we shall see in the next section, those supports will be very useful for computing approximation accuracy. A possible optimization is to only use the exponent of 1 for attributes with values in $\{0, 1\}$ (*i.e.* converted binary attributes).

Let us now examine in more detail the collection of itemsets whose supports are computed by the `PolyApriori` algorithm.

Let $\mathcal{F}$ be a downward closed (*i.e.* $I^{\boldsymbol{\alpha}} \in \mathcal{F}$ implies $J^{\boldsymbol{\beta}} \in \mathcal{F}$ for all $J^{\boldsymbol{\beta}} \sqsubseteq I^{\boldsymbol{\alpha}}$) collection of polynomial itemsets and let I be a set of attributes. Define

$$\mathcal{F} \cap I = \{J^{\boldsymbol{\beta}} \in \mathcal{F} : J \subseteq I\}.$$

It is the set of those polynomial itemsets in $\mathcal{F}$ whose bases are subsets of I.

Let $J = \{X_{l_1}, \dots, X_{l_k}\}$ be a set of attributes. Define an *I-extension* of a polynomial itemset $J^{\boldsymbol{\beta}}$ as

$$\mathrm{ext}_I(J^{\boldsymbol{\beta}}) = \{J^{\boldsymbol{\beta}} \cdot X_{i_j} \text{ for all } X_{i_j} \in I \text{ such that } i_j \geq l_k\}.$$

For a downward closed collection of itemsets $\mathcal{F}$, define its *I-extended border* as

$$\mathcal{B}_I^{\mathrm{ext}}(\mathcal{F}) = \bigcup \{\mathrm{ext}_I(J^{\boldsymbol{\alpha}}) : J^{\boldsymbol{\alpha}} \in \mathcal{F}\} \setminus \mathcal{F}.$$

THEOREM 2.4. *Let $\mathcal{F} = \{I^{\boldsymbol{\alpha}} : \mathrm{supp}(|I^{\boldsymbol{\alpha}}|) \geq \varepsilon\}$. After the* `PolyApriori` *algorithm (with minimum support ε) terminates,*

$$F = \bigcup F_k = \mathcal{F}, \text{ and } \bigcup C_k = \mathcal{F} \cup \mathcal{B}_H^{\mathrm{ext}}(\mathcal{F}).$$

The proof is given in the Appendix.

3 Approximating supports of functions through polynomial expansions

In this section we restrict ourselves to a set of attributes $I = \{X_{i_1}, \dots, X_{i_r}\} \subseteq H$. We assume that multi-indices $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ have r elements.

We will show how to obtain approximations of support of a function based on supports of polynomial itemsets using the function's polynomial expansion. The idea is based on the following observation.

THEOREM 3.1. *Let $f(I)$ be a function which can be expressed by a power series*[1]

$$f(I) = \sum_{i=0}^{\infty} \sum_{\{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=i\}} c_{\boldsymbol{\alpha}} I^{\boldsymbol{\alpha}}. \tag{3.2}$$

Then

$$\mathrm{supp}_D(f) = \sum_{i=0}^{\infty} \sum_{\{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=i\}} c_{\boldsymbol{\alpha}} \mathrm{supp}(I^{\boldsymbol{\alpha}}).$$

Proof. The proof is obtained by summing both sides of (3.2) over all $t \in D$ and changing the order of summations.

$$\begin{aligned} \mathrm{supp}_D(f) &= \sum_{t \in D} f(t[I]) = \sum_{t \in D} \sum_{i=0}^{\infty} \sum_{\{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=i\}} c_{\boldsymbol{\alpha}} (t[I])^{\boldsymbol{\alpha}} \\ &= \sum_{i=0}^{\infty} \sum_{\{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=i\}} c_{\boldsymbol{\alpha}} \sum_{t \in D} (t[I])^{\boldsymbol{\alpha}} \\ &= \sum_{i=0}^{\infty} \sum_{\{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=i\}} c_{\boldsymbol{\alpha}} \mathrm{supp}(I^{\boldsymbol{\alpha}}). \end{aligned}$$

∎

Since, as we know, a large family of functions can be approximated by polynomials, the above theorem tells

[1] In fact the observation is true for any absolutely convergent series expansion, not only power series.

us that supports of those functions can be approximated using supports of polynomial itemsets.

We now present a theorem about computing support of a function based on its Taylor expansion and estimating the approximation error. We will use abbreviated notation $\sum_{\boldsymbol{\alpha}\in\mathcal{F}}$ for summing over all exponents of polynomial itemsets in $\mathcal{F}$.

THEOREM 3.2. *Let $\mathcal{F}$ be a downward closed collection of polynomial itemsets, and $n = \max_{\boldsymbol{\alpha}\in\mathcal{F}} |\boldsymbol{\alpha}|$. Let $f(I)$ be a function with has continuous partial derivatives of order up to $n+1$ at every point in $[-1,1]^r$. Then*

$$\mathrm{supp}_D(f) = \sum_{\boldsymbol{\alpha}\in\mathcal{F}\cap I} \frac{D_I^{\boldsymbol{\alpha}} f(I)|_{I=\mathbf{0}}}{\boldsymbol{\alpha}!}\mathrm{supp}_D(I^{\boldsymbol{\alpha}}) + R,$$

where

$$\begin{aligned} |R| &\leq \sum_{\boldsymbol{\alpha}\in\mathcal{B}_I^{\mathrm{ext}}(\mathcal{F}\cap I)} \frac{M^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\mathrm{supp}_D(|I^{\boldsymbol{\alpha}}|) \\ &\leq M\cdot\varepsilon\cdot \sum_{\boldsymbol{\alpha}\in\mathcal{B}_I^{\mathrm{ext}}(\mathcal{F}\cap I)} \frac{1}{\boldsymbol{\alpha}!} \end{aligned}$$

and

$$M^{\boldsymbol{\alpha}} = \max_{\mathbf{x}\in\{0\}^{(j_{\boldsymbol{\alpha}}-1)}\times[-1,1]^{(r-j_{\boldsymbol{\alpha}}+1)}} |D_I^{\boldsymbol{\alpha}} f(\mathbf{x})|,$$

and $j_{\boldsymbol{\alpha}}$, where $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_r)$, is the largest integer such that $\alpha_{j_{\boldsymbol{\alpha}}} > 0$. M is an absolute upper bound on $[-1,1]^r$ of all partial derivatives of f of order up to $n+1$, and ε is the minimum absolute support.

The proof can be found in the Appendix. At the end of this section we show an example illustrating the way it proceeds.

Let us now discuss the consequences of the above Theorem. It is possible to first find all polynomial itemsets with a given value of minimum absolute support ε, and then estimate the support of arbitrary (sufficiently differentiable) function from the supports of the polynomial itemsets *without* using the data at all. The estimation error may be large for some functions but it can be reliably estimated, without accessing the original dataset. Notice that the `PolyApriori` algorithm computes supports of all polynomial itemsets in the extended border of the collection of frequent itemsets. This is a superset of $\mathcal{B}_I^{\mathrm{ext}}(\mathcal{F}\cap I)$ needed to compute the error in Theorem 3.2. It is thus always possible to compute the Taylor sum over $\mathcal{F}\cap I$ and estimate the error by summing over $\mathcal{B}_I^{\mathrm{ext}}(\mathcal{F}\cap I)$.

Of course to find support of a function we need to obtain its Taylor coefficients and upper bound its derivatives. Those tasks may be time consuming for complicated functions of many variables, but do not require accessing the data, and thus will pay off for large databases. Moreover, as we show in the experimental section, finding supports of arbitrary functions of few variables is very useful and Taylor coefficients of such functions can be computed quickly.

A few remarks are now in place on how the coefficients of Taylor expansions can actually be computed automatically. One option is to symbolically compute function's derivatives and thus obtain the expansion. Symbolic derivation is fully automatic for all elementary functions and their combinations so obtaining the coefficients is straightforward. Another option is to obtain expansions for all elementary functions involved and combine them using operations on series. For example to obtain a series of a product of two functions, we can first expand each of them separately and then multiply the resulting series. Wikipedia's article on Taylor series [11] gives several examples of this procedure. We have tried both approaches and found the second to be much more efficient.

Let us now give an example which illustrates the proof of Theorem 3.2.

EXAMPLE 3.3. Let $I = \{X, Y\}$, and the collection of frequent itemsets $\mathcal{F}\cap I = \{1, X^1, X^2, Y^1, X^1Y^1\}$. We want to approximate the support of a function $f(X,Y)$. We first treat f as a function of X and apply Taylor's Theorem on X. For every $Y\in[-1,1]$ we have

$$\begin{aligned} f(X,Y) &= f(0,Y) + \frac{X^1}{1!}\left.\frac{\partial f(X,Y)}{\partial X^1}\right|_{X=0} \\ &\quad + \frac{X^2}{2!}\left.\frac{\partial^2 f(X,Y)}{\partial X^2}\right|_{X=0} + R_{(3,0)}, \end{aligned} \tag{3.3}$$

where $R_{(3,0)} = \frac{X^3}{3!}\left.\frac{\partial^3 f(X,Y)}{\partial X^3}\right|_{X=\xi}$ for some $\xi\in(-1,1)$, and

$$\begin{aligned} |R_{(3,0)}| &\leq \frac{|X^3|}{3!}\max_{(x,y)\in[-1,1]^2}\left|\left.\frac{\partial^3 f(X,Y)}{\partial X^3}\right|_{\substack{X=x\\Y=y}}\right| \\ &= \frac{|I^{(3,0)}|}{(3,0)!}M^{(3,0)} \end{aligned}$$

We stopped the expansion at X^2 because X^3 is not in $\mathcal{F}\cap I$. Apply now Taylor's Theorem to every term in (3.3), except $R_{(3,0)}$. Each time expand till the highest power of Y for which the corresponding polynomial itemset is still in $\mathcal{F} \cap I$:

$$
\begin{aligned}
f(X,Y) &= f(0,0)\\
&+ \frac{Y^1}{1!}\frac{\partial f(X,Y)}{\partial Y^1}\bigg|_{\substack{X=0\\Y=0}} + R_{(0,2)}\\
&+ \frac{X^1}{1!}\frac{\partial f(X,Y)}{\partial X^1}\bigg|_{\substack{X=0\\Y=0}}\\
&\qquad + \frac{X^1}{1!}\frac{Y^1}{1!}\frac{\partial^2 f(X,Y)}{\partial X \partial Y}\bigg|_{\substack{X=0\\Y=0}} + R_{(1,2)}\\
&+ \frac{X^2}{2!}\frac{\partial^2 f(X,Y)}{\partial X^2}\bigg|_{\substack{X=0\\Y=0}} + R_{(2,1)} + R_{(3,0)}\\
&= \sum_{\boldsymbol{\alpha} \in \mathcal{F} \cap I} \frac{I^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} D_I^{\boldsymbol{\alpha}} f(I)|_{I=\mathbf{0}} + R.
\end{aligned}
$$

The derivation of other remainders is analogous to the case of $R_{(3,0)}$ and is omitted. We have $R = R_{(3,0)} + R_{(0,2)} + R_{(1,2)} + R_{(2,1)}$, and $|R| \leq |R_{(3,0)}| + |R_{(0,2)}| + |R_{(1,2)}| + |R_{(2,1)}|$. Note that $\mathcal{B}_I^{\text{ext}}(\mathcal{F} \cap I) = \{X^3, Y^2, X^1Y^2, X^2Y^1\}$, so the remainders are indeed computed over the extended border of $\mathcal{F} \cap I$. Summing over all $t \in D$, as in the proof of Theorem 3.1, completes the derivation for this case.

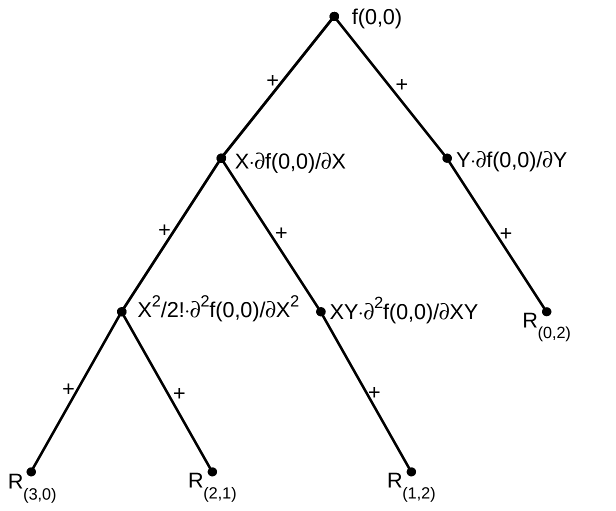


Figure 2: Graphical illustration of the Taylor expansion in Example 3.3

The example is depicted graphically in Figure 2. The figure shows the tree of polynomial itemsets as it would be visited by the `PolyApriori` algorithm. It can be seen that the tree coincides with the Taylor expansion presented above. Every node in the tree corresponds to one term of the expansion. The Taylor expansion first expands f on X thus following the leftmost path in the tree. Then every node on that path is in turn expanded on Y. Leaves of the tree are the remainder terms which form the extended border. ∎

Approximating representations. It follows from the above discussion, that we have obtained an analogue of concise representations [7] for numerical data. We prefer to use the name *approximating representation* since our aim is to approximate support of arbitrary functions. A formal definition is given below.

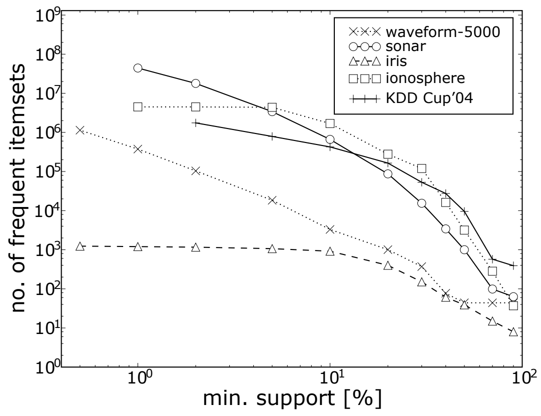
Definition 3.4. An *approximating representation* of a dataset D for a class of functions $\mathcal{G}$ is a set of statistics computed from D such that support in D of every function from $\mathcal{G}$ can be approximated together with upper bound on the approximation error. ∎

Any downward closed collection of polynomial itemsets together with its extended border is an approximating representation for the class of functions having Taylor expansions. Theorem 3.1 can however be applied to arbitrary polynomial expansions, and Weierstrass Theorem [8] states that every numerical function has a polynomial approximation of arbitrary accuracy, so the class of functions can be considerably extended. Our approach is different from [2, 5] which focus on estimating support of unknown itemsets.

4 Experimental Evaluation and Applications

In this section we evaluate our approximating representation experimentally. For performance reasons, it is often necessary to limit the maximum number of attributes in polynomial itemsets (denoted $\max_r$) and their maximum degree (denoted $\max_k$). The value of $\max_r$ depends on the functions we want to approximate in the future. Elementary functions of l arguments require $\max_r$ of at least l. To approximate a sum of functions, $\max_r$ needs to be at least the largest $\max_r$ needed for any of the summed functions. For a product of functions we need $\max_r$ of at least the sum of their required $\max_r$'s. To get useful approximations $\max_k$ needs to be greater than $\max_r$.

Performance evaluation of the `PolyApriori` algorithm. The `PolyApriori` algorithm was implemented in Python and ran on a 1.7GHz Pentium machine. Figure 3 shows computation time and the number of frequent polynomial itemsets (including the extended border) for various datasets and support thresholds. The $\max_k$ of 5 was used, except for the `KDD Cup'04 physics` dataset where $\max_k = 4$, no $\max_r$ limit was used.



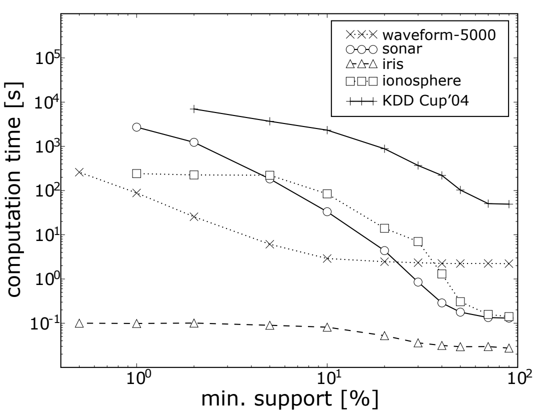


Figure 3: Runtime and number of frequent polynomial itemsets counted for the `PolyApriori` algorithm for various datasets and minimum support thresholds. $\max_k = 5, \max_r = \infty$

In a more realistic setting, Figure 4 shows algorithm performance for $\max_r = 4$ and $\max_k = 9$. We found that such a high value of $\max_k$ allows for very accurate approximations (see below).

It can be seen from the figures, that the algorithm allows for mining large, realistic datasets, especially if $\max_r$ threshold is used. Large number of frequent polynomial itemsets can be a problem in some cases.

Accuracy of approximation. We will now show estimated approximation accuracy for various minimum support thresholds for `waveform-5000` and `KDD Cup'04 physics` datasets. The charts show upper bounds on relative support approximation accuracy obtained using Theorem 3.2; the actual errors were in fact much smaller. We computed the upper bounds for functions of 1, 2 and 3 variables, for each possible attribute, pair and triple of attributes respectively for various levels of minimum support. Figure 5 shows box plots for each function and level of minimum support. The plots show the minimum, maximum, median and the first and third quartile of the error bounds taken over all single attributes, pairs or triples of attributes respectively. For example the upper left figure shows that for `waveform-5000` dataset with 2% minimum support that the median of the upper bound of the relative approximation error of support of $\exp(X_i)$ is about 0.08% and the maximum 0.2%. Logarithmic scale is used on the y axis.

It should be noted that the error estimates show relative upper bounds on errors, which are bound to be high for functions with low support. This explains high error rates for some sets of attributes. Also note, that the charts show an upper bound and the true errors are in fact much lower.

Despite those difficulties, it can be seen that the estimation error decreases rapidly with the decrease of minimum support, and very accurate approximations are possible in most cases. This depends on the properties of the function, for example $\sin^2$, whose higher order derivatives are high, is more difficult to estimate than exp. Also, support of functions of larger number of variables is more difficult to approximate.

Also note that the KDD Cup'04 physics dataset has very skewed distributions on some variables. When going from min. support 5% to 2% this causes a jump in accuracy for one variable functions, since a large number of new frequent polynomial itemsets suddenly become available.

We will now present some illustrative examples of applications of our representation to standard statistical modelling tasks. These are not meant to be industrial quality statistical applications.

Nonlinear regression. In [4] an application of polynomial itemsets to picking terms of polynomial regression has been presented, but regression coefficients were still calculated from the data. In this work, regression coefficients are calculated from the approximating representation without accessing the data at all.

Suppose we want to construct a nonlinear regression model $Y = w_0 + \sum_{i=1}^{n} w_i f_i(X_i)$, where f_i are arbitrary functions, and the weights vector $\mathbf{w} = (w_0, w_1, \ldots, w_n)$ is found using the least squares method. Let $f_D(X_i)$ denote the vector of values of function $f(X_i)$ in each transaction of D, *i.e.* $f_D(X_i) = (f(t.X_i))_{t\in D}$, and f_{0D} a vector of $|D|$ ones. The sought vector $\mathbf{w}$ satisfies the matrix equation $\mathbf{Aw} = \mathbf{b}$, where $\mathbf{A}$ is a matrix s.t. $(\mathbf{A})_{ij} = {f_i}_D(X_i)^T \cdot {f_j}_D(X_j)$, and $\mathbf{b}$ is a column vector with $\mathbf{b}_i = {f_i}_D(X_i)^T \cdot Y$. Notice that $(\mathbf{A})_{ij} = \mathrm{supp}_D(f_i(X_i) \cdot f_j(X_j))$ and $\mathbf{b}_i = \mathrm{supp}_D(f_i(X_i) \cdot Y)$. We

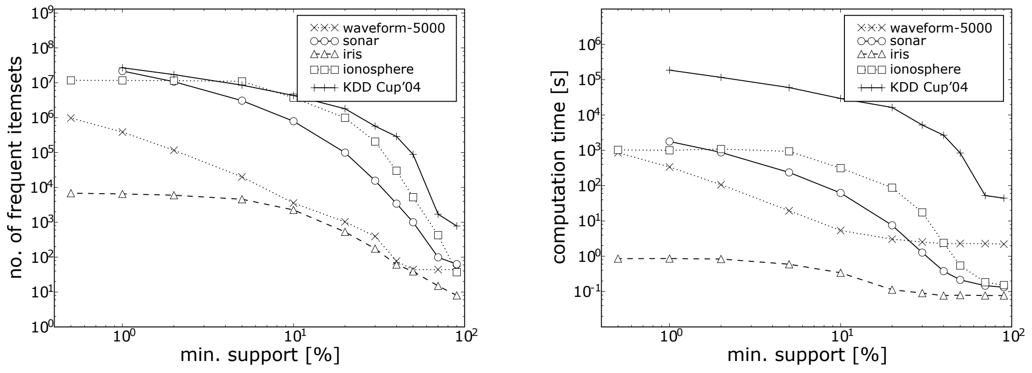
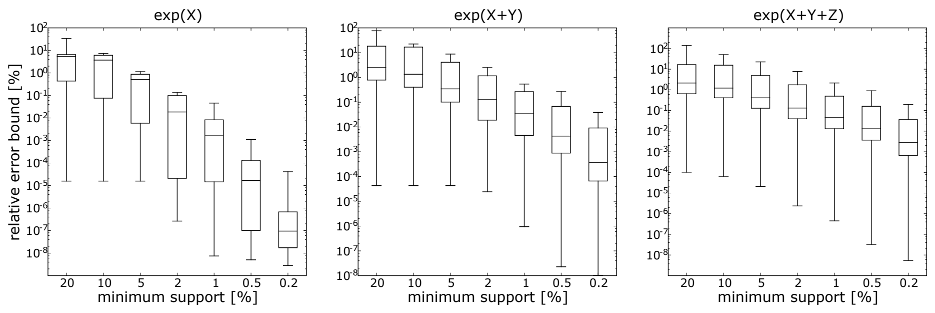
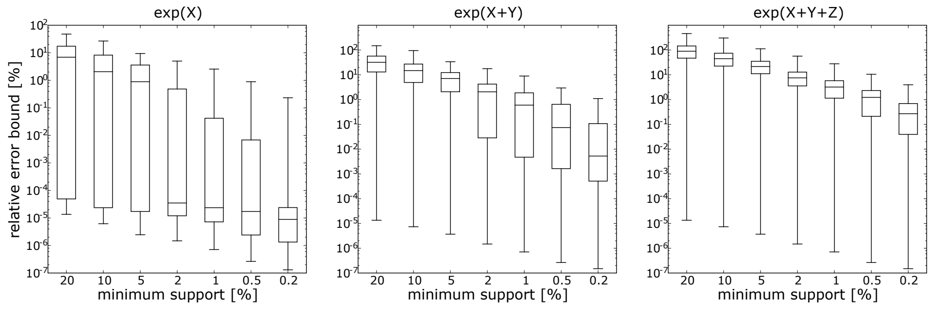


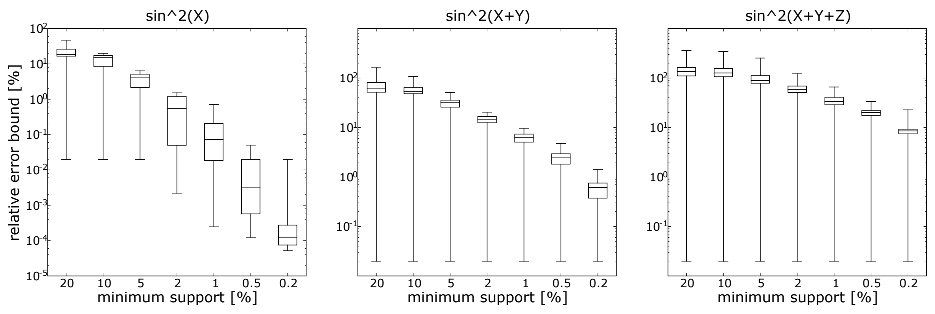
Figure 4: Runtime and number of frequent polynomial itemsets counted for the `PolyApriori` algorithm for various datasets and minimum support thresholds. $\max_k = 9, \max_r = 4$



`waveform-5000`, $\max_k = 9$



`KDD Cup'04 physics dataset`, $\max_k = 9$, $\max_r = 3$



`waveform-5000`, $\max_k = 9$

Figure 5: Upper bounds on relative estimation error for functions of 1, 2 and 3 variables for `waveform-5000` and `KDD Cup'04 physics` datasets at different levels of minimum support.

can thus find Taylor expansions of $f_i(X_i) \cdot f_j(X_j)$ and $f_i(X_i) \cdot Y$, use them to approximate the matrix $\mathbf{A}$ and vector $\mathbf{b}$ from a collection of polynomial itemsets, and solve the matrix equation to find approximate $\mathbf{w}$.

In our experiment we took $f_1 = \sin(2.5X_1)$, $f_2 = \cos(0.5X_2)$, $f_3 = \sin^2(1.1X_3)$, $f_4 = \exp(0.7X_4)$, $f_5 = \sin(3.1X_5)$. We generated an artificial dataset with attributes $X_1, \ldots, X_5$ and an attribute $Y = 0 - 1.5f_1(X_1) + 1.0f_2(X_2) - 2.5f_3(X_3) + 1.0f_4(X_4) - 1.5f_5(X_5) + N(0, 0.03)$, where $N(0, 0.03)$ is a normally distributed noise term with standard deviation 0.03. The data values were generated independently for each variable from a normal distribution $N(0, 1)$. There were 200000 records.

We built a nonlinear regression model $Y = w_0 + \sum_{i=1}^{5} w_i f_i(X_i)$ based on the whole dataset and the approximation procedure described above. Table 1 shows the root mean square errors of the model fitted directly from data and based on polynomial approximations for different values of minimum support.

It can be seen that for higher values of minimum support the error of approximated model can be very high, but as the minimum support decreases, the error converges to that of the model built on full data. High error values for low supports resulted in this case from the fact that large errors caused the regression matrix to become ill-conditioned.

Also note, that since upper bound on approximation error is known, we are able to detect cases when the error is too high so the procedure is reliable.

Notice that if we decide to build a new model for different functions f_i we can do it based on the approximating representation without accessing the data at all. When a large number of models is built this can give huge savings.

Curve fitting We can extend the approach to arbitrary nonlinear curve fitting, where we approximate the value of an attribute Y using a function $f_{\mathbf{w}}(I)$ of attributes $I = \{X_{i_1}, \ldots, X_{i_r}\}$ with parameters $\mathbf{w}$, using the least squares criterion. More precisely, we want to find $\mathbf{w}$ such that $E(\mathbf{w}) = \sum_{t \in D}(t.Y - f_{\mathbf{w}}(t[I]))^2 = \mathrm{supp}_D\big((Y - f_{\mathbf{w}}(I))^2\big)$ is minimized. The matrix equation method is not helpful here, and $E(\mathbf{w})$ has to be minimized directly. Notice that $E(\mathbf{w})$ can be estimated based on a collection of polynomial itemsets (as long as it has a Taylor expansion), and we can apply a minimization algorithm to this approximation.

Unfortunately the Taylor coefficients need to be recomputed for every value of $\mathbf{w}$ used during the optimization, but we can symbolically compute the coefficients as functions of $\mathbf{w}$, and simply recompute them after $\mathbf{w}$ changes.

We generated an artificial dataset with attributes X_1, X_2 and an attribute $Y = 0.2\exp(1.3X_1 - 0.7X_2) - 0.8 + N(0, 0.004)$. The data points were generated from a uniform distribution on $[-0.9, 0.9] \times [-0.9, 0.9]$.

We approximated Y using the function $f_{\mathbf{w}}(X_1, X_2) = w_3 \exp(w_1 X_1 + w_2 X_2) + w_4$. We used Matlab to symbolically compute the Taylor expansion of $E(\mathbf{w})$ on X_1, X_2, Y where each coefficient was a function of w_1, w_2, w_3, w_4. We limited each of the w_i's to the range $[-2, 2]$ to avoid the loss of accuracy of the Taylor expansion for large values of w_i's, and used Matlab's constrained minimization procedure `fmincon` to minimize $E(\mathbf{w})$. More research is needed to remove such limitations on the values of w_i parameters.

We compared our approach with Matlab's `lsqcurvefit` least-squares curve fitting function, which had to perform a data scan at each point of the minimization. The results are shown in Figure 6. The times of running the `PolyApriori` algorithm (minimum support of 1.5%) and performing the Taylor expansion *are* included. It can be seen that using the approximation gives significant performance gains for databases of 100000 records or more, while the accuracy of an approximated fit is comparable to scanning the whole dataset. The accuracy obtained by both methods is high, the error is significantly lower than 1%. The running time of our method is dominated by relatively slow symbolic computation routines.

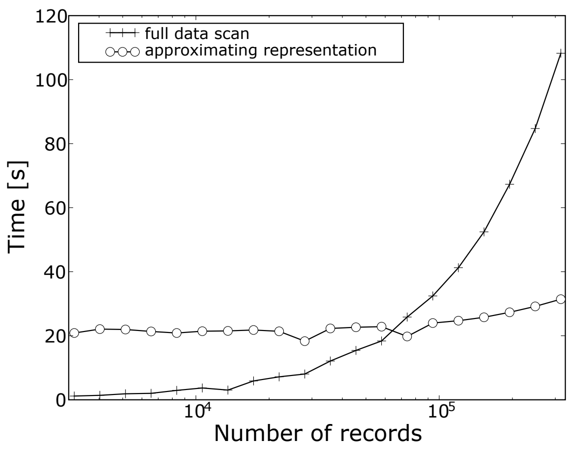
It has to be noted that both methods are sensitive to the choice of the starting values for the weights. Choosing a wrong starting point resulted in high approximation error for either or both methods. This problem is inherent to minimization techniques used, not the result of polynomial approximation. We chose $w_1 = 0.5, w_2 = -0.5, w_3 = 0.1, w_4 = 0.1$ as starting weights, which gave good results for both approaches. Repeating the fit for a number of random points would be another good solution. For 300000 records, our method gave weights $w_1 = 1.3423, w_2 = -0.7168, w_3 = 0.1912, w_4 = -0.7927$, very close to correct values. Matlab's `lsqcurvefit` found weights $w_1 = 1.3018, w_2 = -0.7007, w_3 = 0.1996, w_4 = -0.7996$ which are a better approximation, which however took more than three times as long to compute. Note that our algorithm depended on just 146 numbers (supports of frequent itemsets), while `lsqcurvefit` used the whole table with 300000 records.

In this case our approach is faster even when fitting a single model and taking the time of the `PolyApriori` algorithm into account.

To test how the approach handles large datasets, we repeated the experiment for the KDD Cup'04 physics dataset (training part). To this end we added an extra

min. support [%]	1.58	1.17	0.86	0.63	0.46	0.34	0.25	0.18	0.14	0.1
no. of frequent itemsets	40	44	64	92	94	167	157	199	313	336
approximation	0.094	3.1e9	8.7e2	0.042	0.095	0.088	0.034	0.033	0.023	0.027
full data set	0.027	0.033	0.028	0.020	0.028	0.032	0.015	0.015	0.017	0.026

Table 1: Root mean square errors of nonlinear regression fitted directly from data and based on polynomial approximations for different values of minimum support.



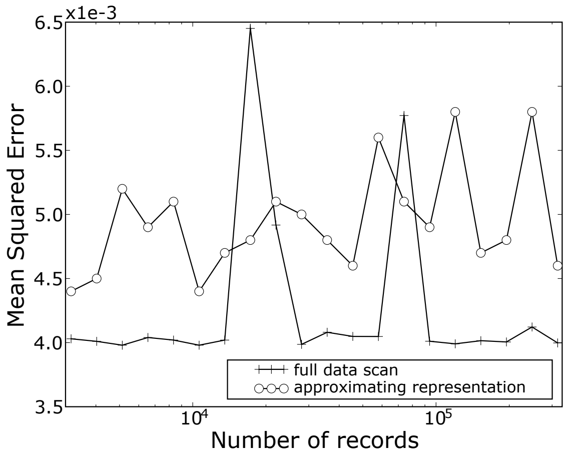


Figure 6: Performance and accuracy of nonlinear least-squares curve fitting of an artificial dataset using full dataset and approximation.

attribute Y computed as (after scaling all X's to the interval $[-1, 1]$):

$$Y = \sum_{i=1}^{78} 0.3 \exp(X_i).$$

We then fitted a function $Y = a + \sum_{i=1}^{78} b_i \exp(c_i X_i)$ to the data using the least squares criterion.

We used $max_k = 9$ and minimum support of 0.5%. Notice that in for this particular problem the series expansion of the squared error only contains terms of up to two variables. We took advantage of this, setting $\max_r = 2$ which allowed us to raise $\max_k$ to 12 and use no minimum support at all.

We found symbolic computations of Taylor coefficients too slow here so we used the other approach, applying operations on series to construct the coefficients of the full series from coefficients of expansions of each exponential function. This approach proved to be much faster. In fact it was fast enough to allow for estimation of the gradient of the error function, and as a result to use a more efficient minimization routine based on conjugate gradient descent.

Performance and accuracy of approximate curve fitting for $\max_k$ of 9 and 12 as well as analogous results for Matlab's curve fitting are shown in Figure 7.

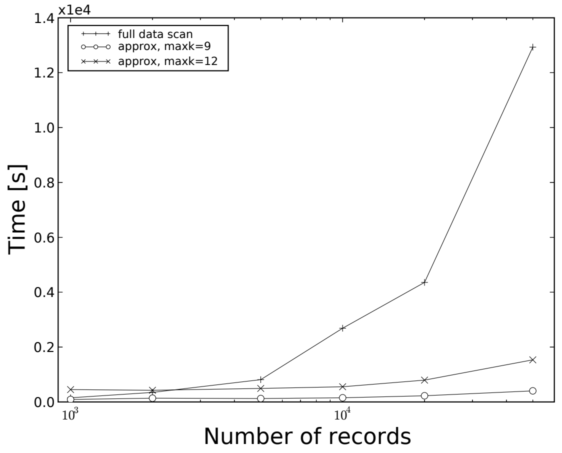
It can be seen that the Mean Squared Error of approximated curve fitting is higher than Matlab's. It should however be noted that the variance of Y is 7.3, which is much higher than the MSE of around 0.3 of our fit for $\max_k = 9$ and which dwarfs the 0.0049 MSE for $\max_k = 12$. We thus managed to obtain a successful fit at a fraction of the cost of using traditional methods.

Notice that in Figure 7 the time of mining frequent itemsets *is* included. In practice frequent polynomial itemsets would have been mined beforehand, and the speed advantage would have been even greater.

5 Application to data streams

A potential application area for polynomial approximating representations are data streams. Since data in a data stream can be looked at only once, algorithms requiring multiple passes over the data cannot be applied. A solution would be to continuously update a collection of polynomial itemsets based on the data stream. The collection would serve as the approximating representation. Since the collection embodies information about all data seen so far in the data stream, approximate models could be built based on the polynomial itemsets approximating (hopefully well) models built on the whole stream.

The problem of course lies in maintaining a collec-



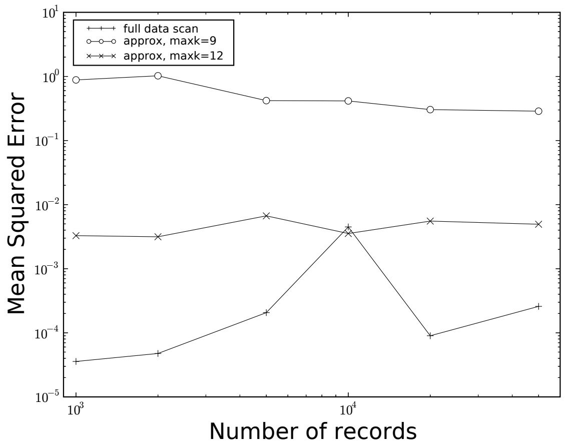


Figure 7: Performance and accuracy of nonlinear least-squares curve fitting on the `physics` dataset using standard curve-fitting and polynomial approximations.

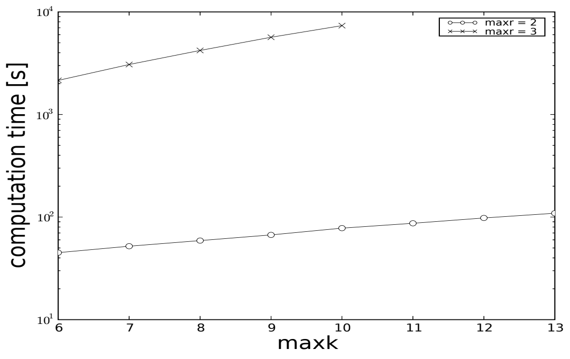


Figure 8: Finding a full collection of polynomial itemsets for given $\max_k$ and $\max_r$ in the `physics` data.

tion of polynomial itemsets while reading stream data. Here we will adopt the simplest possible solution of keeping all itemsets with given $\max_k$ and $\max_r$. This is equivalent to setting the minimum support to 0, but since a full collection is being mined, this can be done by simple counting without the overhead of `PolyApriori`.

Figure 8 shows the timing for finding such a collection of polynomial itemsets from the KDD Cup'04 physics database. It can be seen that the approach is feasible for $\max_r$ of 2 and 3.

Another approach would be to generalize an approach such as [6] to maintain a collection of frequent polynomial itemsets with given minimum support. We leave it as a topic of future research.

6 Discussion and Future Research

In the paper, a definition of support for arbitrary real-valued functions has been presented, and shown how supports of large a family of functions can effectively be approximated based on supports of polynomial itemsets. Experiments have shown that polynomial itemsets with given minimum absolute support can efficiently be mined by an Apriori like procedure, and that accurate approximations of supports of many important functions can be obtained.

We have shown that a number of real-life tasks such as nonlinear regression and curve fitting can be performed based only on the collection of frequent polynomial itemsets without accessing the data at all. In many cases significant performance gains over standard procedures have been demonstrated. In fact we have moved the difficulty related to numerical computations on large datasets to the domain of symbolic computations related to computing series expansions of functions, which is independent of database size.

The main problem of the presented concise representation is accuracy of Taylor approximation. While very accurate for values close to the point around which the series was expanded ($\mathbf{0}$ in our case), the approximation error increases rapidly for arguments far from the expansion center. Consider for example an attribute T representing time and an attribute X depending periodically on T, say $X = \sin(aT)$ for some constant a. If the number of periods included in data is large we will not be able to give a good approximation of the relationship between T and X unless itemsets involving very high exponents of T are known, which is practically impossible. Addressing such issues is a topic for future research, and will involve examination of other function expansions such as Fourier series and Chebyshev approximation, which offer good accuracy over the whole interval (but are less accurate around $\mathbf{0}$).

In Section 1 we limited the domain of all attributes to the range $[-1, 1]$. Every attribute X_i with values outside this range was scaled by $c_i = (\max_{t \in D} |t.X_i|)^{-1}$.

This assumption was necessary to guarantee the monotonicity property for absolute support. The interpretation of absolute support of polynomial itemsets given in [4] as the number of records in which the value of an itemset is close to its maximum remains valid after such scaling.

It is possible to undo the effects of this scaling during support estimation. For example if we want to compute support of $f(X_1, X_2, X_3)$, where X_1, X_2, X_3 are original (unscaled) attributes, we can instead compute the support of a function $f'(X_1', X_2', X_3') = f(\frac{X_1'}{c_1}, \frac{X_2'}{c_2}, \frac{X_3'}{c_3})$, where X_1', X_2', X_3' are scaled versions of X_1, X_2, X_3 respectively. Unfortunately this will usually add extra multiplicative terms to function derivatives, and can cause a loss of estimation accuracy.

Alternative error estimation methods need to be examined for certain cases, such as the function $\sqrt{1+x}$ which has infinite derivatives around -1. For example, if we know that a polynomial approximates a function with accuracy ϵ on an interval, the support estimation error is bounded by $\epsilon \cdot |D|$. Unfortunately estimating ϵ, although can be done without accessing the table, is more difficult than upper-bounding the derivatives, especially for functions of many variables.

Future work also includes convergence analysis of the expansion with decreasing minimum support, some early results were obtained but are omitted due to lack of space. Another topic of future research is detailed analysis of how approximation error influences accuracy of tasks such as nonlinear regression and curve fitting.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conf. on Management of Data*, pages 207–216, 1993.

[2] T. Calders and B. Goethals. Depth-first non-derivable itemset mining. In *Proc. of the SIAM International Conference on Data Mining (SDM'05)*, 2005.

[3] T. Calders, B. Goethals, and S. Jaroszewicz. Mining rank-correlated sets of numerical attributes. In *KDD'06*, 2006.

[4] S. Jaroszewicz. Polynomial association rules with applications to logistic regression. In *KDD'06*, 2006.

[5] M. Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In *Proc. ICDM*, pages 305–312, 2001.

[6] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. of the 28th VLDB Conference*, 2002.

[7] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 189–194, Portland, OR, August 1996. AAAI Press.

[8] K. Saxe. *Beginning Functional Analysis.* Springer, 2001.

[9] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD Conf. on Management of Data*, pages 1–12, 1996.

[10] M. Steinbach, P.-N. Tan, H. Xiong, and V. Kumar. Generalizing the notion of support. In *Proc. of the 10th International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 689–694, Seattle, WA, August 2004.

[11] `http://en.wikipedia.org/wiki/Taylor_series`. retrieved on 12 Oct, 2006.

A Proof of Theorem 2.4

Proof. Since only itemsets with absolute support greater than or equal to ε are included in F, the inclusion $F \subseteq \mathcal{F}$ is obvious. We prove the converse by induction on k. Take any element of $\mathcal{F}$ of degree 1, it is obviously included in F_1 since C_1 is initialized to all polynomial itemsets of degree 1 and it is frequent. Suppose now that all itemsets of degree k from $\mathcal{F}$ are included in F, and take any polynomial itemset $J^{\boldsymbol{\beta}}$ of degree $k+1$ from $\mathcal{F}$. By induction hypotheses and the downward closed property of absolute support, all its subsets ($\sqsubseteq$) of degree k are in F_k. This implies that $J^{\boldsymbol{\beta}}$ will be generated in step 9, and will become a member of F_{k+1} since it is frequent.

Let us now prove the result for $\bigcup C_k$. The inclusion $\bigcup C_k \subseteq \mathcal{F} \cup \mathcal{B}_H^{\text{ext}}(\mathcal{F})$ follows easily from the candidate generation procedure in step 9. The inclusion $\mathcal{F} = F \subseteq \bigcup C_k$ is also trivial. Take now any $I^{\boldsymbol{\alpha}} = X_{i_1}^{\alpha_1} \dots X_{i_r}^{\alpha_r} \in \mathcal{B}_H^{\text{ext}}(\mathcal{F}) = \mathcal{B}_H^{\text{ext}}(F)$. From the definition of $\mathcal{B}_H^{\text{ext}}(F)$, $X_{i_1}^{\alpha_1} \dots X_{i_r}^{\alpha_r - 1} \in F$, and by the candidate generation procedure in step 9, $I^{\boldsymbol{\alpha}} \in \bigcup C_k$. ∎

B Proof of Theorem 3.2

We begin by proving a lemma.

Lemma B.1. *(Taylor's theorem for downward closed collections) Consider a function $f(I)$ where $I = \{X_{i_1}, \dots, X_{i_r}\}$, which has continuous partial derivatives up to order $n+1$ at every point of $[-1,1]^r$. For every $\mathbf{x} \in [-1,1]^r$ we have*

$$f(\mathbf{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{F} \cap I} \frac{D_I^{\boldsymbol{\alpha}} f(\mathbf{x})|_{\mathbf{x}=\mathbf{0}}}{\boldsymbol{\alpha}!} \mathbf{x}^{\boldsymbol{\alpha}} + R,$$

where

$$|R| \leq \sum_{\boldsymbol{\alpha} \in \mathcal{B}_I^{\text{ext}}(\mathcal{F} \cap I)} \frac{M^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} |\mathbf{x}^{\boldsymbol{\alpha}}|,$$

and

$$M^{\boldsymbol{\alpha}} = \sup_{\mathbf{x}\in\{0\}^{(j-1)}\times[-1,1]^{(r-j+1)}} |D_I^{\boldsymbol{\alpha}} f(\mathbf{x})|,$$

where j is the largest integer such that $\alpha_j > 0$.

Proof. Let $I_j = \{X_{i_1}, \ldots, X_{i_j}\}$ for $0 \le j \le r$. We will prove a stronger result, that for every $0 \le j \le r$,

$$f(\mathbf{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{F}\cap I_j} \frac{D_I^{\boldsymbol{\alpha}} f(\mathbf{x})|_{I_j=\mathbf{0}}}{\boldsymbol{\alpha}!}\mathbf{x}^{\boldsymbol{\alpha}} + R_j, \tag{2.4}$$

where

$$|R_j| \le \sum_{\boldsymbol{\alpha}\in\mathcal{B}_{I_j}^{\text{ext}}(\mathcal{F}\cap I_j)} \frac{M^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}|\mathbf{x}^{\boldsymbol{\alpha}}|.$$

The proof is by induction on j. The base case $j = 0$ is trivially true since the only possible value of $\boldsymbol{\alpha}$ is $\mathbf{0}$ (corresponding to the constant 1 polynomial in no variables) and $\mathcal{B}_{I_j}^{\text{ext}}(\mathcal{F}\cap I_j)$ is empty, so the right-hand-side becomes simply $f(\mathbf{x})$.

In the inductive step, assume the lemma is true for some $j < r$. Denote by $(\boldsymbol{\alpha}, q)$ the multi-index obtained from $\boldsymbol{\alpha}$ by replacing α_{j+1} with q.

We apply Taylor's theorem for one variable to every term of the sum in (2.4). For a term with index $\boldsymbol{\alpha} \in \mathcal{F}\cap I_j$ we get that for some $\xi \in (-1,1)$

$$\begin{aligned}
&\frac{\mathbf{x}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} D_I^{\boldsymbol{\alpha}} f(\mathbf{x})|_{I_j=\mathbf{0}} \\
&= \frac{\mathbf{x}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\sum_{q=0}^{m_{\boldsymbol{\alpha}}} \frac{X_{i_{j+1}}^q}{q!}\frac{\partial^q}{\partial X_{i_{j+1}}^q} D_I^{\boldsymbol{\alpha}} f(\mathbf{x})\Big|_{\substack{I_j=\mathbf{0}\\ X_{i_{j+1}}=0}} \\
&\quad + \frac{\mathbf{x}^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\frac{X_{i_{j+1}}^{m_{\boldsymbol{\alpha}}+1}}{(m_{\boldsymbol{\alpha}}+1)!}\frac{\partial^{m_{\boldsymbol{\alpha}}+1}}{\partial X_{i_{j+1}}^{m_{\boldsymbol{\alpha}}+1}} D_I^{\boldsymbol{\alpha}} f(\mathbf{x})\Big|_{\substack{I_j=\mathbf{0}\\ X_{i_{j+1}}=\xi}} \\
&= \sum_{q=0}^{m_{\boldsymbol{\alpha}}} \frac{D_I^{(\boldsymbol{\alpha},q)} f(\mathbf{x})\Big|_{I_{j+1}=\mathbf{0}}}{(\boldsymbol{\alpha},q)!}\mathbf{x}^{(\boldsymbol{\alpha},q)} + R_{\boldsymbol{\alpha}}, \qquad (2.5)
\end{aligned}$$

where $m_{\boldsymbol{\alpha}}$ is the highest number such that $\mathbf{x}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}})}$ belongs to $\mathcal{F}\cap I_{j+1}$, and

$$|R_{\boldsymbol{\alpha}}| \le \frac{M^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)}}{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)!}|\mathbf{x}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)}|.$$

Applying (2.5) to every term of the sum in the inductive hypothesis and we get

$$\begin{aligned}
f(\mathbf{x}) &= \sum_{\boldsymbol{\alpha}\in\mathcal{F}\cap I_j}\sum_{q=0}^{m_{\boldsymbol{\alpha}}} \frac{D_I^{(\boldsymbol{\alpha},q)} f(\mathbf{x})\Big|_{I_{j+1}=\mathbf{0}}}{(\boldsymbol{\alpha},q)!}\mathbf{x}^{(\boldsymbol{\alpha},q)} \\
&\quad + R_j + \sum_{\boldsymbol{\alpha}\in\mathcal{F}\cap I_j} R_{\boldsymbol{\alpha}}. \qquad (2.6)
\end{aligned}$$

We will now show that the double summation above is equivalent to summing over $\mathcal{F}\cap I_{j+1}$. It is easy to see that all indices $(\boldsymbol{\alpha}, q)$ in the double sum are distinct. As a direct consequence of the way $m_{\boldsymbol{\alpha}}$ is chosen, $(\boldsymbol{\alpha}, q) \in \mathcal{F}\cap I_{j+1}$ for every $\boldsymbol{\alpha}$ and $0 \le q \le m_{\boldsymbol{\alpha}}$.

Conversely, take any $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{j+1}, 0, \ldots, 0) \in \mathcal{F}\cap I_{j+1}$. $\boldsymbol{\beta}$ can be rewritten as $(\boldsymbol{\beta}', \beta_{j+1})$ for some multi-index $\boldsymbol{\beta}' = (\beta_1, \ldots, \beta_j, 0, \ldots, 0)$. Since $\mathcal{F}$ is downward closed, $\boldsymbol{\beta}' \in \mathcal{F}\cap I_j$, and since $\boldsymbol{\beta} = (\boldsymbol{\beta}', \beta_{j+1}) \in \mathcal{F}$, it must be that $0 \le \beta_{j+1} \le m_{\boldsymbol{\beta}'}$, and the term corresponding to $\boldsymbol{\beta}$ is included in the double summation in (2.6).

Notice now that

$$\begin{aligned}
|R_{j+1}| &= \Big|R_j + \sum_{\boldsymbol{\alpha}\in\mathcal{F}\cap I_j} R_{\boldsymbol{\alpha}}\Big| \le |R_j| + \sum_{\boldsymbol{\alpha}\in\mathcal{F}\cap I_j}|R_{\boldsymbol{\alpha}}| \\
&\le \sum_{\boldsymbol{\alpha}\in\mathcal{B}_{I_j}^{\text{ext}}(\mathcal{F}\cap I_j)} \frac{M^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}|\mathbf{x}^{\boldsymbol{\alpha}}| \qquad (2.7) \\
&\quad + \sum_{\boldsymbol{\alpha}\in\mathcal{F}\cap I_j} \frac{M^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)}}{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)!}|\mathbf{x}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)}|. \qquad (2.8)
\end{aligned}$$

It remains to show that the sums (2.7) and (2.8) are equivalent to summing over $\mathcal{B}_{I_{j+1}}^{\text{ext}}(\mathcal{F}\cap I_{j+1})$.

By inductive hypothesis all terms present in R_j have their respective polynomial itemsets in $\mathcal{B}_{I_j}^{\text{ext}}(\mathcal{F}\cap I_j)$ and thus in $\mathcal{B}_{I_{j+1}}^{\text{ext}}(\mathcal{F}\cap I_{j+1})$. Let us look at new remainder terms added in the induction step. Take any $\boldsymbol{\alpha} \in \mathcal{F}\cap I_j$. It 'generates' one remainder term $R_{\boldsymbol{\alpha}}$ involving a polynomial itemset ${I_{j+1}}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)}$. By definition of $m_{\boldsymbol{\alpha}}$ we have ${I_{j+1}}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}})} \in \mathcal{F}\cap I_{j+1}$ and ${I_{j+1}}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)} \notin \mathcal{F}\cap I_{j+1}$, and by definition of $\mathcal{B}_{I_{j+1}}^{\text{ext}}(\mathcal{F}\cap I_{j+1})$, ${I_{j+1}}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}}+1)} = {I_{j+1}}^{(\boldsymbol{\alpha},m_{\boldsymbol{\alpha}})}\cdot X_{i_{j+1}} \in \mathcal{B}_{I_{j+1}}^{\text{ext}}(\mathcal{F}\cap I_{j+1})$.

Conversely take any $\boldsymbol{\beta} \in \mathcal{B}_{I_{j+1}}^{\text{ext}}(\mathcal{F}\cap I_{j+1})$, and let X_{i_l} be the last variable with a nonzero exponent in $\boldsymbol{\beta}$. If $i_l < i_{j+1}$ then it is also true that $\boldsymbol{\beta} \in \mathcal{B}_{I_j}^{\text{ext}}(\mathcal{F}\cap I_j)$ and by inductive hypothesis, the term corresponding to it is included in R_j. If $i_l = i_{j+1}$ then there must be exactly one multi-index $\boldsymbol{\alpha}$ such that ${I_{j+1}}^{\boldsymbol{\beta}} = {I_{j+1}}^{\boldsymbol{\alpha}}\cdot X_{i_{j+1}}$. By definition of $\mathcal{B}_{I_{j+1}}^{\text{ext}}(\mathcal{F}\cap I_{j+1})$, ${I_{j+1}}^{\boldsymbol{\alpha}} \in \mathcal{F}\cap I_{j+1}$ and ${I_{j+1}}^{\boldsymbol{\beta}} \notin \mathcal{F}\cap I_{j+1}$, so the last nonzero exponent of $\boldsymbol{\beta}$ must be $m_{\boldsymbol{\alpha}}+1$ and thus the term corresponding to it is present in the remainder sum in (2.8) exactly once.

The result follows by proceeding with the induction until $j = r$. ∎

Proof. (of Theorem 3.2) The theorem follows by applying Lemma B.1 to f in the definition of support and rearranging the sums. ∎