

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 [7]. Unfortunately discretization always leads to information loss, and has other problems such as difficulty in selecting appropriate interval widths. In [8] 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 $H = \{X_1, \dots, X_n\}$. Elements of D are called *transactions*, following data-mining conventions. Sets of attributes will be denoted with uppercase letters I, J . Vectors of variable values from the domain of a set of attributes will be denoted with boldface letter $\mathbf{x} = (x_1, \dots, x_r)$. We further define $\mathbf{0} = (0, \dots, 0)$ as a vector of zeros.

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

We will make heavy use of the so called *multi-index notation*. A *multi-index* α is a sequence of integers $\alpha = (\alpha_1, \dots, \alpha_r)$. We define the following expressions:

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

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 are scaled appropriately, *i.e.* an attribute X_i is multiplied by a positive constant $c_i = (\max_{t \in D} |t.X_i|)^{-1}$. 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).

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

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

2 Support of real-valued functions, polynomial itemsets

In this section we 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 I . Define *support* and *absolute support* of f in dataset D respectively as $\text{supp}_D(f) = \sum_{t \in D} f(t[I])$, and $\text{supp}_D(|f|) = \sum_{t \in D} |f(t[I])|$. \square

DEFINITION 2.2. Let $I = \{X_{i_1}, \dots, X_{i_r}\} \subseteq H$ be a set of attributes. A *polynomial itemset of degree $|\alpha|$* is an expression of the form $I^\alpha = X_{i_1}^{\alpha_1} X_{i_2}^{\alpha_2} \dots X_{i_r}^{\alpha_r}$. \square

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

Let I^α, I^β , where $\alpha = (\alpha_1, \dots, \alpha_r)$, $\beta = (\beta_1, \dots, \beta_r)$, be two polynomial itemsets. We say that I^α is a subset of I^β , denoted $I^\alpha \sqsubseteq I^\beta$, if $\alpha_i \leq \beta_i$ for all $i \in \{1, \dots, r\}$. Similarly we say that I^α is a strict subset of I^β , denoted $I^\alpha \sqsubset I^\beta$ if in addition $\alpha_i < \beta_i$ for some $i \in \{1, \dots, r\}$. Neither support of polynomial itemsets, nor its absolute value are monotone w.r.t. inclusion. Fortunately, this is true for absolute support.

THEOREM 2.3. *Absolute support of polynomial itemsets is monotone w.r.t. inclusion, that is $I^\alpha \sqsubseteq I^\beta$ implies $\text{supp}_D(I^\alpha) \geq \text{supp}_D(I^\beta)$.*

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 based on the fact that $\text{supp}_D(f) \leq \text{supp}_D(|f|)$.

Several properties of absolute support for polynomial itemsets justifying the definition and giving it some intuitive interpretation have been given [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. 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

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:    $F_k \leftarrow \{I^\alpha \in C_k : \text{supp}_D(|I^\alpha|) \geq \varepsilon\}$ 
5:   compute support of all itemsets in  $F_k$ 
6:    $C_{k+1} \leftarrow \emptyset$ 
7:   for all  $I^\alpha = X_{i_1}^{\alpha_1} \dots X_{i_r}^{\alpha_r} \in F_k$  do
8:     for all  $j \geq i_r$  do
9:        $C_{k+1} \leftarrow C_{k+1} \cup \{I^\alpha \cdot X_j\}$ 
10:    end for
11:  end for
12:   $k \leftarrow k + 1$ 
13: end loop

```

Figure 1: The PolyApriori algorithm

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.

Let us now examine the collection of itemsets whose supports are computed by the PolyApriori algorithm. Let \mathcal{F} be a downward closed (*i.e.* $I^\alpha \in \mathcal{F}$ implies $J^\beta \in \mathcal{F}$ for all $J^\beta \sqsubseteq I^\alpha$) collection of polynomial itemsets and let I be a set of attributes. Define

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

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 J^β as

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

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

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

THEOREM 2.4. *Let $\mathcal{F} = \{I^\alpha : \text{supp}(|I^\alpha|) \geq \varepsilon\}$. After the PolyApriori algorithm (with minimum support ε) terminates, $F = \bigcup F_k = \mathcal{F}$, and $\bigcup C_k = \mathcal{F} \cup \mathcal{B}_H^{\text{ext}}(\mathcal{F})$.*

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 α, β 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.

THEOREM 3.1. *Let $f(I)$ be a function expressible by a power series $f(I) = \sum_{i=0}^{\infty} \sum_{\{\alpha:|\alpha|=i\}} c_{\alpha} I^{\alpha}$. Then*

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

Since a large family of functions can be approximated by polynomials, the Theorem tells 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. Denote by $\sum_{\alpha \in \mathcal{F}}$, the sum over all exponents of polynomial itemsets in \mathcal{F} .

THEOREM 3.2. *Let \mathcal{F} be a downward closed collection of polynomial itemsets, and $n = \max_{\alpha \in \mathcal{F}} |\alpha|$. 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*

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

where $|R| \leq \sum_{\alpha \in \mathcal{B}_I^{\text{ext}}(\mathcal{F} \cap I)} \frac{M^{\alpha}}{\alpha!} \text{supp}_D(|I^{\alpha}|)$, and

$$M^{\alpha} = \max_{\mathbf{x} \in \{0\}^{(j_{\alpha}-1)} \times [-1, 1]^{(r-j_{\alpha}+1)}} |D_I^{\alpha} f(\mathbf{x})|, \text{ and } j_{\alpha}, \text{ is the largest integer such that } \alpha_{j_{\alpha}} > 0.$$

The proof is omitted but below we give an example illustrating the way it proceeds.

EXAMPLE 3.3. Let $I = \{X, Y\}$, and the collection of frequent itemsets $\mathcal{F} \cap I = \{1, X^1, X^2, Y^1, X^1 Y^1\}$. 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 $f(X, Y) = f(0, Y) + \frac{X^1}{1!} \frac{\partial f}{\partial X^1} \Big|_{X=0} + \frac{X^2}{2!} \frac{\partial^2 f}{\partial X^2} \Big|_{X=0} + R_{(3,0)}$, where, for some $\xi \in (-1, 1)$, $R_{(3,0)} = \frac{X^3}{3!} \frac{\partial^3 f(X, Y)}{\partial X^3} \Big|_{X=\xi} \leq \frac{|I^{(3,0)}|}{(3,0)!} M^{(3,0)}$.

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, except $R_{(3,0)}$. Each time expand till the highest available power of Y : $f(X, Y) = f(0, 0) + \frac{Y^1}{1!} \frac{\partial f}{\partial Y^1} \Big|_{Y=0} + R_{(0,2)} + \frac{X^1}{1!} \frac{\partial f}{\partial X^1} \Big|_{Y=0} + \frac{X^1 Y^1}{1! 1!} \frac{\partial^2 f}{\partial X \partial Y} \Big|_{Y=0} + R_{(1,2)} + \frac{X^2}{2!} \frac{\partial^2 f}{\partial X^2} \Big|_{Y=0} + R_{(2,1)} + R_{(3,0)}$
 $R_{(3,0)} = \sum_{\alpha \in \mathcal{F} \cap I} \frac{I^{\alpha}}{\alpha!} D_I^{\alpha} f(I)|_{I=0} + R.$

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 $\mathcal{B}_I^{\text{ext}}(\mathcal{F} \cap I) = \{X^3, Y^2, X^1 Y^2, X^2 Y^1\}$, so the remainders are indeed computed over the extended border of $\mathcal{F} \cap I$. To go from variables to supports we sum over all $t \in D$.

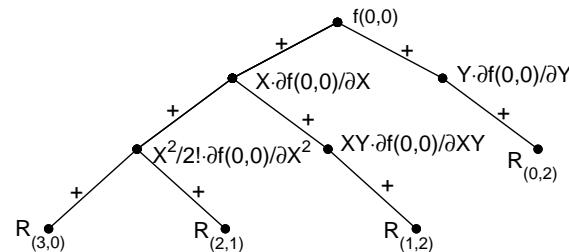


Figure 2: Graphical illustration of the Taylor expansion in Example 3.3

The example is depicted graphically in Figure 2, which 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. Leaves are the remainder terms which form the extended border $\mathcal{B}_I^{\text{ext}}(\mathcal{F} \cap I)$. \square

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 including $\mathcal{B}_I^{\text{ext}}(\mathcal{F} \cap I)$ 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 $\mathcal{B}_I^{\text{ext}}(\mathcal{F} \cap I)$.

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, but do not require accessing the data, and thus will pay off for large databases. One option is to compute function's derivatives symbolically 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. We

found the second approach to be much more efficient, so we adopted it in the experiments.

Approximating representations. We have thus obtained an analogue of concise representations [2, 6, 5] for numerical data. We prefer to use the name *approximating representation* since our aim is to approximate supports of arbitrary functions, not just of unknown itemsets.

4 Experimental Evaluation and Applications

In this section we evaluate our method 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. Elementary functions of l arguments require $\max_r \geq l$. 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. Useful results require \max_k to be greater than \max_r .

Performance evaluation of the PolyApriori algorithm. The PolyApriori algorithm was implemented in Python on a 1.7GHz Pentium machine. Figure 3 shows computation time and the number of frequent itemsets (including the extended border) for various datasets and support thresholds. The value of \max_k was 9 and of \max_r 4. We found that such value of \max_k allows for very accurate approximations (see below).

It can be seen that the algorithm allows for mining large, realistic datasets. 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 the KDD Cup'04 physics dataset. 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 4 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 2% minimum support the median of the upper bound of the relative approximation error of support of $\exp(X_i)$ is about $3.16 \cdot 10^{-5}\%$.

It should be noted that the error estimates show relative upper bounds on errors, which are naturally high when support is low. This explains high errors for

some sets of attributes. Also note, that the charts show an upper bound and the true errors are in fact much lower. Also note that the KDD Cup'04 physics dataset has very skewed distributions on some variables. When going from minimum support 5% to 2% this causes a jump in accuracy for one variable functions, since a number of new frequent polynomial itemsets become available.

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.

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 $\mathbf{w} = (w_0, w_1, \dots, w_n)$ is found using the least squares method. The sought vector \mathbf{w} satisfies the matrix equation $\mathbf{A}\mathbf{w} = \mathbf{b}$, where \mathbf{A} is a matrix s.t. $(\mathbf{A})_{ij} = \text{supp}_D(f_i(X_i) \cdot f_j(X_j))$, and \mathbf{b} is a column vector with $\mathbf{b}_i = \text{supp}_D(f_i(X_i) \cdot Y)$. We 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, \dots, 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. 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. The table below shows the root mean square errors of the model fitted directly from data and based on polynomial approximations for different values of minimum support.

| min _{supp} [%] | 1.58 | 0.86 | 0.46 | 0.25 | 0.14 |
|-------------------------|-------|-------|-------|-------|-------|
| # itemsets | 40 | 64 | 94 | 157 | 313 |
| approx. | 0.094 | 8.7e2 | 0.095 | 0.034 | 0.023 |
| full data | 0.027 | 0.028 | 0.028 | 0.015 | 0.017 |

It can be seen that for higher values of minimum

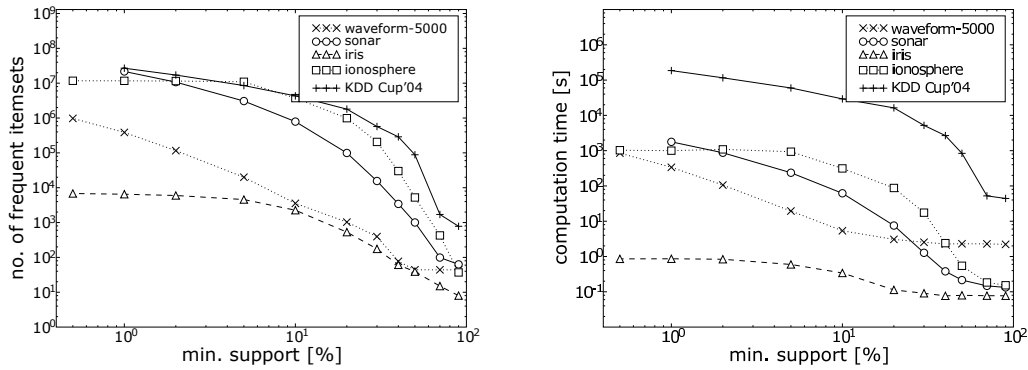


Figure 3: Runtime and number of frequent polynomial itemsets counted for the PolyApriori algorithm for various datasets and minimum support thresholds. $\max_k = 9, \max_r = 4$

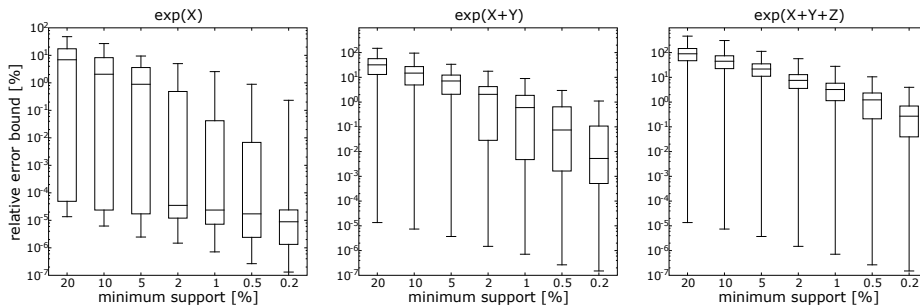


Figure 4: Upper bounds on relative estimation error for functions of 1, 2 and 3 variables for KDD Cup'04 physics dataset at different levels of minimum support and $\max_k = 9$.

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 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 $I = \{X_{i_1}, \dots, X_{i_r}\}$ with parameters \mathbf{w} , using the least squares criterion. We want to find \mathbf{w} such that $E(\mathbf{w}) = \sum_{t \in D} (t.Y - f_{\mathbf{w}}(t[I]))^2 = \text{supp}_D((Y - f_{\mathbf{w}}(I))^2)$ is minimized. In this case $E(\mathbf{w})$ has to be minimized directly. Notice that $E(\mathbf{w})$ can be estimated based on a collection of polynomial itemsets, and we can apply any

minimization algorithm to this approximation.

Our approximation was fast enough to allow for estimation of the gradient of the error function, and as a result for using a more efficient minimization routine based on conjugate gradient descent.

We tested the approach on the KDD Cup'04 physics dataset (training part). To this end we added an extra 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 that data.

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 performed the experiments for various sizes of samples drawn from the dataset to illustrate how algorithms' performance depends on the number of records.

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 5.

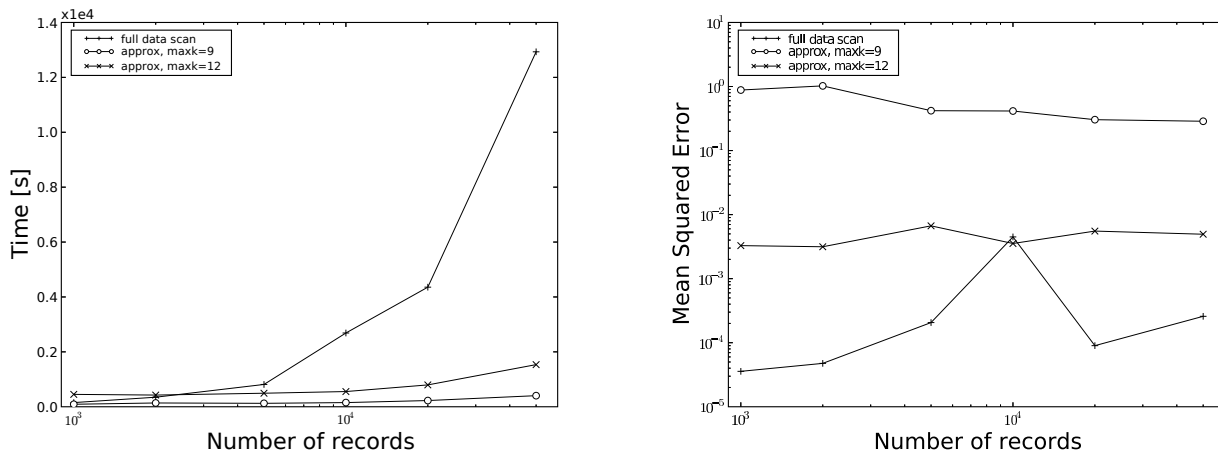


Figure 5: Performance and accuracy of nonlinear least-squares curve fitting on the `physics` dataset using standard curve-fitting and polynomial approximations.

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 5 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.

Application to data streams. A potential application area are data streams. Data in a data stream can be looked at only once so multi-pass algorithms 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. Some early results can be found in the full version of the paper.

5 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.

Future work includes trying other approximation methods, such as Chebyshev expansion, convergence analysis of the expansion with decreasing minimum support and 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*, pages 207–216, 1993.
- [2] T. Calders and B. Goethals. Depth-first non-derivable itemset mining. In *SIAM Int. Conf. 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] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *KDD'96*, pages 189–194, Portland, OR, August 1996. AAAI Press.
- [7] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD*, pages 1–12, 1996.
- [8] M. Steinbach, P.-N. Tan, H. Xiong, and V. Kumar. Generalizing the notion of support. In *KDD'04*, pages 689–694, Seattle, WA, August 2004.