Mining surprising patterns using temporal description length

Soumen Chakrabarti Sunita Sarawagi Byron Dom

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
{soumen,sunita,dom}@almaden.ibm.com

Abstract

We propose a new notion of *surprising* temporal patterns in market basket data, and algorithms to find such patterns. This is distinct from finding *frequent* patterns as addressed in the common mining literature. We argue that once the analyst is already familiar with prevalent patterns in the data, the greatest incremental benefit is likely to be from *changes in the relationship* between item frequencies over time.

A simple measure of surprise is the extent of departure from a model, estimated using standard multivariate time series analysis. Unfortunately, such estimation involves models, smoothing windows and parameters whose optimal choices can vary dramatically from one application to another. In contrast, we propose a precise characterization of surprise based on the *number of bits* in which a basket sequence can be encoded under a carefully chosen coding scheme. In this scheme it is inexpensive to encode sequences of itemsets that have steady, hence likely to be well-known, correlation between items. Conversely, a sequence with large code length hints at a possibly surprising correlation.

Our notion of surprise also has the desirable property that the score of a set of items is offset by anything surprising that the user may already know from the marginal distribution of any proper subset. No parameters, such as support, confidence, or smoothing windows, need to be estimated or specified by the user.

We experimented with real-life market basket data. The algorithm successfully rejected a large number of frequent sets of items that bore obvious and steady complementary relations to each other, such as cereal and milk. Instead, our algorithm found itemsets that showed statistically strong fluctuations in correlation over time. These items had no obviously complementary roles.

1 Introduction

Data warehousing technology has enabled corporations to store huge amounts of data, and data mining has become a major motivating application. Large data sources suitable for mining are growing in number and size literally every passing moment.

For almost any such data source collected over years to decades, there will be *prevalent* patterns or broad regularities that are already known to domain experts, and *surprising* patterns that are novel, unexpected and non-trivial to explain. There may be patterns of both types that are statistically significant. There is broad consensus [15, 21, 22, 23] that the success of data mining will depend critically on the ability to go beyond *obvious* patterns and find *novel* and *useful* patterns [12]. Otherwise the results of mining will often be large and lack novelty, making it overwhelming and unrewarding for the analyst to sieve through them. A domain expert who is already familiar with the application domain is very unlikely to be satisfied with merely prevalent patterns, because (1) presumably the company is already exploiting them to the extent possible and (2) the competition knows about these patterns as well. The payoff from data mining lies in surprising second-order phenomena.

An ill-defined, vague notion of "domain knowledge" gets in the way of separating the novel patterns from the prevalent ones. In principle, one can propose various well-defined notions of domain knowledge. The analyst has a mental multivariate distribution over the attributes, and the system reports, from a certain class of patterns, those that reduce the distance between the mental distribution and the true distribution at the quickest rate. Of course, it is impossible to implement this in a real system.

1.1 Our contributions

This paper proposes and explores the notion that analysis of *variation of inter-item correlations along time* can approximate the role of domain knowledge in the search for interesting patterns. We concentrate on the problem of boolean market basket data [1, 2]. A set of k items is declared as "interesting" not necessarily because its absolute support exceeds a user-defined threshold, but because the *relationship* between the items changes over time. Furthermore, even if the support of the itemset changes over time, it is not considered interesting if the changes are totally *explained* by the changes in the support of smaller subsets of items.

We develop this notion of interest based on the number of bits needed to encode a itemset sequence using a specific coding scheme that we design. In this scheme it takes relatively few bits to encode sequences of itemsets that have steady correlation between items (which are likely to be well-known). Conversely, a sequence with large code length (relative to a baseline unconstrained coding scheme) hints at a possibly surprising correlation [19]. The surprise value of the itemset is related to the difference or ratio between the constrained and unconstrained code lengths. As a subroutine in this computation, our analysis produces, in a formal

Proceedings of the 24th VLDB Conference New York, USA, 1998

information-theoretic sense, the "best" segmentation of time for the interesting itemsets, based on how the relationship between items is changing. This segmentation technique could be of independent interest.

Our work potentially leads to improved accuracy and data understanding. Data collected over long intervals are generated by processes with *drifting* parameters. A single model over all time may lead to poor models and predictive accuracy. In this scenario, the large volume of data available for mining hurts rather than helps. Our analysis builds models for optimal segments of time, avoiding this problem. Other applications include the segmentation of data for supervised learning along time to discover interesting changes in decision boundaries as time passes.

1.2 Comparison with other approaches

We prefer this somewhat involved analysis to simple statistical tests. Any local estimate of support needs a time window: a small window leads to poor statistical confidence of estimated parameters and a large window may skip over a surprising segment and average it into a larger, uninteresting one. Window size, smoothing function, test statistic, surprise threshold, will all be critical choices best made by one highly experienced in the art. Similar statements hold for approaches based on frequency transformations. We wish to avoid any need for tuning and instead base everything on only one assumed property: that baskets are independently drawn from a possibly drifting distribution. The distinction from statistical approaches is discussed in detail in §7.

Compared to standard mining algorithms in which the core operations are pattern generation and tuple counting, our analyses involves more expensive computations. Part of the paper will describe techniques to greatly reduce the computation cost, but the complexity of the algorithms remains. We believe this stand is justified in view of the aforesaid urgency for mining tools to ignore mundane rules and discover novel ones [12, 15, 21, 22, 23]. Otherwise, the time saved in computation may well be spent by analysts discarding rules by inspection!

1.3 Organization of the paper

In §2 we review Information Theory basics and describe our model for sequences of market baskets. In §3 we present the components of our analysis to find surprising patterns. We give an overview of how these fit together in §4. In §5 we present techniques for improving the performance of our method. In §6 we report on our experience with two real-life market basket data sets. Related work and alternative approaches are reviewed in §7 and concluding remarks made in §8.

2 Modeling sequences of market baskets

2.1 Information theory basics

The underlying premise in this work is that data which can be described using very few bits is simple and predictable, and hence less likely to be found interesting or surprising. Information Theory gives us a general way to construct a model for given data while regarding it as a compression problem. Imagine a sender S, wishing to send some data $\vec{x}$ to a receiver R, using as few bits as possible. A suitable *model* M is first transmitted, consuming $L(M)$ bits. Typically M is chosen from a class of models and associated parameter space that S and R have agreed upon in advance. For example, S may transmit the mean and variance of a normal distribution to R, if there is reason to believe that the data follows a normal distribution.

Second, the data is encoded suitably and sent to R, using what we denote as $L(\vec{x}|M)$ bits of information. Because R knows M, data compression may be greatly enhanced, i.e., $L(\vec{x}|M)$ may be much smaller than $L(\vec{x})$. Specifically, suppose $\Pr[\vec{x}|M]$ is the probability of a specific data value $\vec{x}$ given the model M. Then Shannon's classical Information Theorem [10] states that the data can be encoded in $L(\vec{x}|M) = -\log \Pr[\vec{x}|M]$ bits. Note that S and R are motivated to pick a coding scheme that minimizes $L(M) + L(\vec{x}|M)$. The initial choice of the parameters values of M are usually made to maximize $\Pr[\vec{x}|M]$. The classical *Minimum Description Length* (MDL) principle [19] argues that this will generally lead to models that capture exactly the regularities in the data and avoid over-fitting to random deviations.

Example 1: Suppose we are given two bit strings of identical length and asked to identify the "more complex" one, without any further information. Then a reasonable approach would be to compress both using the "universal" Lempel-Ziv (LZ) compression scheme and select the larger of the two compressed strings. LZ yields asymptotically optimal compression, but does not produce a model-based explanation of the data. If the data were a finite sequence of floating point numbers drawn from a normal distribution, LZ could be a poor choice. □

2.2 Data and model for basket sequences

In our case, the raw data is a sequence (x_τ) of market baskets, each basket being a set of items, ordered by time τ. Let us focus on a fixed itemset with k items. Then each basket contains one of the 2^k possible subsets of the k items in it. Thus, if we are reasoning about k items, we can regard each basket to be the outcome of tossing a 2^k sided coin (better called a die) with the presence or absence of the i-th item encoded in the i-th bit of the toss outcome written as a k-bit number between 0 and $2^k - 1$.

First consider the case where a sequence of tosses (i.e., baskets) are generated from one coin. Our model M associates with each face of the coin a probability. For two items the model M, has four terms $p_{00}, p_{01}, p_{10}, p_{11}$ where p_{00} denotes the probability that both the items are absent, p_{01} denotes the probability that the first one is absent but the second is present and so on. This notation generalizes to k items in the obvious way.

To capture drift in the process underlying the baskets, we assume a random process that generates the data as follows: it has a set of coins with various face probabilities, unknown to us, the observers. It picks some coin arbitrarily and then tosses it for an arbitrary number of trials before moving on to another coin. Each coin thus defines a segment of time where the itemset distribution is *stationary*. We can observe only the sequence of outcomes. For simplicity and efficiency we pick stationary models within each segment, not ones whose parameters gradually drift with time. This is in the same spirit as the simple decision boundaries used by classification and regression trees. More complicated segment models can be used, but at larger complexity.

To summarize, in our model M we represent a k-itemset sequence as a set of segments where each segment is generated from a 2^k sided coin. The model cost $L(M)$ has two parts: the **parameter cost** which for each segment includes the cost of encoding the coin biases like p_{00} and p_{01} and the **segmentation cost** which includes the cost of encoding the number of segments and the boundary of each segment. The **data cost** is estimated by applying Shannon's theorem with each segment and summing up the log probabilities (since the segments are assumed to be independent).

3 Segmentation of basket sequences

We use the model introduced above as a basis for defining the interest measure of an itemset. We develop this definition in three steps. First we discuss how to find the best segmentation of an itemset using the model described above. We call this the **unconstrained segmentation** problem (§3.1). The unconstrained model does not provide the best compression because it ignores two factors. First, when modeling a k-itemset, user's knowledge of $k-1$ itemsets is not exploited. Second, even when the individual parameters of the model change from segment to segment, the relationship between the parameters may remain constant over time. In the second stage we incorporate these two factors into a **constrained segmentation** problem (§3.2). Finally, we explore means of comparing the complexity of diverse itemsets using a reference segmentation for each (§3.3). If the difference or ratio between the constrained and reference segmentation is large, we regard the itemset as surprising.

3.1 Unconstrained segmentation

We will first discuss the simpler case of segmenting a single item and then discuss generalizations to higher dimensions.

3.1.1 Single item

If a store has only one item, each basket can be thought of as the outcome of a single (2-sided) coin toss, based on whether the item was purchased or not. Given a sequence of T tosses, how do we find a segmentation? This is not as simple a question as it may seem at first glance. At one extreme, we can assume there are T coins, each with bias 0 or 1: given this model, the data fits perfectly; i.e., the probability of generating the given sequence is 1. At the other extreme we can posit only one coin: under this model the probability of generating the given sequence may be very low if indeed the sequence was generated from many coins with diverse biases. One could try estimating biases over windows of trials and then merge or split them. The danger is in picking these windows. Small windows will not give bias estimates with sufficient statistical confidence, and large windows may skip over a very interesting but small segment where the bias was remarkably different.

It turns out that there *is* a notion of the "correct" segmentation of the given sequence, defined in terms of MDL defined earlier and we next present a method for finding such a segmentation.

Claim 2 *The segmentation and coin parameters that minimizes $L(M) + L(\vec{x}|M)$ can be computed in $O(T^2)$ time.*

Proof. We construct a graph with $T+1$ nodes, and directed edges (i,j) for every $0 \le i < j \le T$. Let there be $t(i,j)$ trials between i and j, with $h_1(i,j)$ heads and $h_0(i,j)$ tails (these observed values are collectively called $\vec{x}$ above). Edge (i,j) is assigned a cost $c(i,j)$ which represents the model (sum of parameter and segmentation cost) and data cost for encoding the tosses between i (excluded) and j (included). Calculating this cost involves the following steps:

Estimating model parameters: First, we need to find the model parameters $p_1(i,j)$ and $p_0(i,j)$. The values of these parameters that optimize the data fit are their maximum likelihood (ML) estimates[1] calculated from the data as: $p_1(i,j) = h_1(i,j)/t(i,j)$, and $p_0(i,j) = 1 - p_1(i,j)$.

Finding data encoding cost: Then, we use the above parameters to calculate the data encoding cost for segment (i,j) (using Shannon's theorem) as

$$L(x|M) \quad = \quad -\log p_1^{h_1} p_0^{h_0} \quad = \quad -\textstyle\sum_r h_r \log p_r, \quad (1)$$

where each parameter is over the segment (i,j).

Finding parameter encoding cost: We need to transmit $p_1(i,j)$ (or $p_0(i,j)$; knowing one is enough). Note that the maximum likelihood estimate for $p_1(i,j)$ can take only one of $t(i,j)+1$ values, so we need to send only about $\log t(i,j)$ bits (although the parameters are real numbers).

Finding segmentation cost: This is just the encoding for the boundaries of each segment (i,j), or the number j itself, costing us $\log T$ bits for each of the m segments. (Actually, for m coins we just need $\log\binom{T-1}{m-1}$ bits, but this is very close to $m \log T$ for $m \ll T$.)

Finally we find the shortest path from node 0 to note T in $O(|E| + |V| \log |V|) = O(T^2)$ time. Each edge of the shortest path is a segment in the optimal segmentation. ■

The capability to find the exact optimum is important as a baseline even if the computation takes more

[1]To avoid problems with parameters approaching zero or one we use Laplace's rule [16], but we ignore this detail in our description.

than linear time. Also, later we will explore means to greatly reduce the computation cost in practice, while producing a segmentation of near-optimal quality.

3.1.2 Larger itemsets

For k-itemsets, our model is a sequence generated by a 2^k-sided coin. We can apply the same shortest path procedure as in the one-itemset case to find the best segmentation. The only difference is in the detail of computing the edge weights $c(i,j)$ corresponding to each segment.

Consider first the case of two items. Suppose we are given a set of $t > 0$ baskets over two items, among which h_{00} have neither of the items, h_{11} have both, h_{10} have the first item but not the second, and h_{01} have the second and not the first. These induce parameters p_{rs} as before. The data encoding cost is a direct extension of the one-item case, viz., $-\sum_{r,s} h_{rs} \log p_{rs}$. However, the way we estimate the model parameters p_{rs} and encode their costs changes because for the two item case there are two models to choose from. The first model corresponds to the case where the two items are *independent*, in which case only two parameters, $p_{1\cdot}$ and $p_{\cdot 1}$ are needed to specify the coin, the rest being calculated as $p_{11} = p_{1\cdot}p_{\cdot 1}$, $p_{10} = p_{1\cdot} - p_{11}$, etc. The parameter cost in this case is $\log(t+1)^2 \approx 2\log t$. The second model corresponds to when the items are *dependent*, in which case we need three parameters to specify the four-sided coin. The model cost is the logarithm of the number of ways in which t trials can be divided into four outcomes, $\log \binom{t-1}{3} \approx 3 \log t$. For both models we get the maximum likelihood estimates of the required parameters from the data and use the parameters to evaluate the data cost. The segmentation cost is also the same as in the one-item case. We evaluate the total cost for the edge (i,j) for both model types and take the smaller cost as $c(i,j)$.

The number of possible models grows with the number of items. For 3-itemsets, depending on how the marginals are related, there are eight possibilities. The simplest case is when all two-way itemsets are independent, in which case three parameters are needed. The most general case is when all three-way marginals are correlated, in which case seven parameters are needed. Let the items be a, b, c. In between there are three cases of the form "(a,b) independent, (b,c) independent, (a,c) dependent" and three cases of the form "(a,b) dependent, (b,c) dependent, (a,c) independent." Four parameters suffice for all six cases. Similar enumerations can be generated for larger itemsets.

The general recipe for calculating edge weights for any k-itemset is as follows: for each possible model M:

1. Estimate the model parameters using the corresponding counts from the data. For example, for the independent model in the two-itemset case, we estimate $p_{1\cdot} = h_{1\cdot}/t$ and $p_{\cdot 1} = h_{\cdot 1}/t$.
2. Estimate all the k-dimensional parameters from the model parameter calculated above. For example, for independent model in two dimensions we estimate $p_{11} = p_{1\cdot}p_{\cdot 1}$, $p_{10} = p_{1\cdot} - p_{11}$, etc. In general, these parameters may not always have closed form solutions for estimating them from lower dimensional probabilities. In §3.1.3 we discuss the general procedure.
3. Find data encoding cost using above k-dimensional parameters. This is a straightforward generalization of the the two itemset case, $-\sum_{r,s} h_{rs} \log p_{rs}$.
4. Find the parameter encoding cost for the independent parameters of the model. If a model has ξ parameters, $\xi \log t$ is often a good estimate, although in some cases further refinements are possible, as discussed in §3.2.3.
5. Find the segmentation cost.
6. Find total cost as the sum of the data, parameter and segmentation cost.

Finally, select the model with the smallest total cost and assign its cost to the edge.

The number of models to be searched can increase exponentially with the number of dimensions. However, we can greatly reduce the number by using the following simple heuristic. If for a $k-1$ dimensional itemset, a particular model was found to be the best, then for a k dimensional itemset start from that model and only consider generalizations of that model. For example, if for the two-itemset (a,b), the dependent model was found better than the independent model then for the three-itemset (a,b,c) do not consider any model in which a and b are independent.

3.1.3 Estimating k dimensional probabilities from marginals of fewer dimensions

In general, it is not always possible to get closed form formulas for this estimation. Consider, for instance the three-itemset case. Of the eight cases discussed earlier, all but one yield-closed form solutions for $\hat{p}_{111}$. For instance, when one of the item pairs (say a and b) is dependent and the other two pairs are independent we can calculate the expected value as $\hat{p}_{111} = p_{1\cdot 1}p_{\cdot 1 \cdot}$. The problem case is when all three-way atomic probabilities are correlated. In this case, there is no explicit formula for computing the expected support $\hat{p}_{111}$ from the observed marginals. But there are simple iterative procedures [6, 8] that *converge* to the maximum likelihood (ML) estimate for $\hat{p}_{111}$. The iteration can be used even in cases where direct formulas exist; the iterative process will yield (in one iteration) the same answer as the closed form formulas when they exist [6, page 83]. We describe a classical algorithm called *Bartlett's method* for finding the probability of a k itemset given marginal probability of its subsets.

Bartlett's iterative procedure: For simplicity we discuss this process for three dimensions. The input consists of twelve 2-way marginals: four for each of three pairs of items namely: $p_{11\cdot}, p_{10\cdot}, p_{01\cdot}, p_{00\cdot}$ for item pair 1 and 2 and so on for the other two pairs. The process converges to values for all the eight 3-way probabilities $\hat{p}_{110}, \hat{p}_{101}, \ldots, \hat{p}_{000}$, so that these are the "least restrictive or maximal likelihood" estimates of the three dimensional probabilities while preserving the specified values of the 2-way marginal probabilities. That is, $p_{11\cdot} = \hat{p}_{111} + \hat{p}_{110}$ and so on.

The process starts by first assigning a starting value of 1 to each of the eight 3-way probabilities. Then in each step of the iteration it scales the 3-way probabilities so as to come closer to the observed marginals. It repeats this process until a suitable error threshold is reached.

```
Initialize $\hat{p}_{ijk} = 1$ for $i, j, k \in \{0, 1\}$
While error between iterations is high
    For each of the twelve two-way marginals update the
        3-way probabilities to fit that marginal better:
        e.g., for $p_{01\cdot}$ update as:
            $\hat{p}_{011} = \frac{p_{01\cdot}\hat{p}_{011}}{\hat{p}_{01\cdot}}$
            $\hat{p}_{010} = p_{01\cdot} - \hat{p}_{011}$
```

Claim 3 ([6]) *The iteration is guaranteed to converge for any k.*

3.2 A measure of surprise and the "single θ segmentation"

In this section we will propose some answers to this question: How do we detect that one sequence of baskets is more interesting than another? Intuitively, we want an answer to have the following properties:

- A k-itemset should not be found surprising simply because it has larger support than a pre-specified quantity, but because its support is significantly different from what is expected *given* the marginal supports of all proper subsets. Similar concerns have been raised by Brin and others [22].
- In order to produce a ranking by surprise measure, the measure should reflect, in a uniform way across different itemsets with diverse absolute support, the *complexity* of variation of correlation along time.

To satisfy the first requirement, we need a method of calculating expected support of an itemset from the support of its subsets. First we estimate expected support of the single items. Lacking prior knowledge of the data we can assume that all items are equally likely (prior knowledge can be easily integrated). Therefore, the expected support of each item in the data is the ratio of the average transaction length to the number of items.

For 2-itemsets, given the observed marginal supports $p_{1\cdot}$ and $p_{\cdot 1}$ of the individual items, the best (maximum likelihood) estimate of the 2-itemset is $\hat{p}_{11} = p_{1\cdot}p_{\cdot 1}$ derived using the assumption that the two itemsets are independent. If the actual support, $p_{11} \gg \hat{p}_{11}$ then this 2-itemset is interesting. Thus, our natural choice for the measure of surprise is

$$\theta \quad = \quad p_{1\cdots 1}/\hat{p}_{1\cdots 1}. \tag{2}$$

For larger itemsets, we use the method discussed in §3.1.3 to calculate the expected support from the lower dimensional marginals. For instance, for three-itemsets, if all two-way marginals are dependent we use the iterative procedure discussed earlier to calculate expected support.

The above notion generalizes previous work on estimating the expected value by Brin et al [22] that make the simplifying assumption that three-way itemsets are found interesting only when none of the three two-way marginals were dependent.

3.2.1 The single θ segmentation

Recall from §2.1 our scheme of compressing the data w.r.t. a model M chosen from a certain model class. In the unconstrained case in §3.1, M could be *any* sequence of coins.

Given two basket sequences, we can perform the unconstrained segmentation, but given these two coin sequences, which is more complex? A direct formal way of comparing the complexity of two coin sequences appears elusive. For example, a large number of unconstrained segments for a 2-itemset may not be surprising if the segmentation is always caused by one of the items and the items are always independent.

Since we assume that the analyst assumes "constant θ unless proved otherwise," we must encode the data w.r.t. a model M' from the *restricted model class* containing all possible sequences of coins in which all coins have the same θ value (say the value that is computed from data over all time). We call this the "single-θ segmentation." Roughly speaking, itemsets with large code length in this model class depart from the analyst's prior expectation and are therefore interesting.

3.2.2 Approximate solution for general k

Consider a 2-itemset. First we take the entire data and find which of the independent or the dependent model fits the data better using the procedure in §3.1.2. Call this the global model M_g. If M_g is the independent model, the global value of θ is 1 (costing just one bit) otherwise estimate the value as:

$$\theta(0,T) \quad = \quad \frac{h_{11}(0,T)/T}{(h_{1\cdot}(0,T)/T)(h_{\cdot 1}(0,T)/T)}. \tag{3}$$

For setting up the shortest path instance, we must then assign the coding cost to edge (i, j). The segmentation and parameter costs are computed as before. To compute the data cost, we must estimate the coin face probabilities given the observed record counts $h_{00}, h_{01}, h_{10}, h_{11}$ (over the t records in this time range (i, j)). We calculate from these the observed marginals $\bar{p}_{\cdot 1} = h_{\cdot 1}/t$ and $\bar{p}_{1\cdot} = h_{1\cdot}/t$, which then give all the parameters of our coin for segment (i, j) via $\hat{p}_{11} = \theta\bar{p}_{\cdot 1}\bar{p}_{1\cdot}$, $\hat{p}_{10} = \bar{p}_{1\cdot} - \hat{p}_{11}$, etc. If the resulting coin is inconsistent (i.e., some face probability is not between zero and one) we declare the edge (i, j) as having infinite cost. It is easy to see that there is at least one feasible path in the shortest path problem, viz., the $(0, T)$ path, that remains unaffected by this approximation. In §3.2.3 we discuss a more elaborate exact procedure for $k = 2$. Given this coin we estimate the probability of the observed data in this segment as $\Pr[\vec{h}|\vec{p}] = \sum_{r,s} h_{r,s} \log \hat{p}_{r,s}$ as before. We add the data cost to the model cost as evaluated in the various cases described in §3.1.2 and take the minimum for the edge cost $c(i, j)$.

The procedure remains essentially unchanged for larger itemsets. For a k-itemset, we find over all time the best model M_g. If M_g is the complete model involving k dimensional probabilities as parameters, the expected value $\hat{p}_{1\cdots 1}(0,T)$ is computed by Bartlett's iteration using the $k-1$ dimensional marginals, and the global θ is then calculated as

$$\theta(0,T) = \frac{h_{1\cdots 1}(0,T)/T}{\hat{p}_{1\cdots 1}(0,T)}.$$

For segment (i,j), we compute observed $k-1$ and marginals $\bar{p}_{\dots}(i,j)$ over only the records in interval (i,j). Next we invoke the iterative algorithm on these observed marginals to obtain $\hat{p}_{1\dots 1}$, and compute all the other $\bar{p}$'s as discussed for the two-item case. Again, if the coin becomes inconsistent we mark the edge as infeasible; this can be looked at as a kind of approximation. If M_g is not the complete model, we apply the same model M_g on all the segments and compute the edge cost as in the unconstrained case.

3.2.3 Exact solution for 2-itemsets[2]

For the important special case of 2-itemsets, we do not need the approximate above; we have an exact characterization of the code length. We envision a system with precise analysis for the $k=2$ case and approximations beyond; in §3.1.2 we noted that the most interesting patterns we found were for $k=2$, and indeed all patterns for $k=3$ were explained by 2-way marginals.

The potential difficulty with summarily discarding inconsistent coins as above is that there may exist a consistent coin satisfying the global θ constraint which does not exactly fit the observed marginals but still has a reasonably high data probability.

Therefore, we cannot compute the $\bar{p}$'s directly from observed data, but must cast this as a constrained optimization problem. Thus we want to assign consistent values to variables $p_{r,s} : r,s \in \{0,1\}$ is as to obey the constraints and maximize the data probability (minimize data cost). Let the observed numbers of "heads" of the four types be $h_{r,s}$. Then we must solve the following constrained non-linear optimization problem over unknowns $p_{r,s}$ (all other quantities are numerically known):

$$\begin{array}{ll} \max & \sum_{r,s\in\{0,1\}} h_{r,s} \log p_{r,s} \\ \text{subject to} & \left\{ \begin{array}{l} \sum_{r,s\in\{0,1\}} p_{rs} = 1 \\ p_{11} = \theta(p_{10}+p_{11})(p_{01}+p_{11}). \end{array} \right. \end{array} \quad (4)$$

This can be solved using a variety of iterative techniques [14]. We use a simple steepest ascent algorithm.

Finally we turn to the question of the optimal way of encoding the parameters. To do this we produce a particular coding scheme and argue that nothing can be better. Here is the scheme, we omit the argument for lack of space. Recall that θ is transmitted once for all segments; this is a negligible cost. For all segments, this fixed value of θ will be used. For segment (i,j), we will send two parameters. As we have noted earlier, although these are interpreted as real numbers, the model can only assume a discrete number of configurations, and we can thus send the parameters as integers. We will pick two integers $\phi_{1\cdot}$ and $\phi_{\cdot 1}$ to send. The intent is that the receiver will compute the model as $p_{1\cdot} = \phi_{1\cdot}/t$, $p_{\cdot 1} = \phi_{\cdot 1}/t$, $p_{11} = \theta\phi_{1\cdot}\phi_{\cdot 1}$, and thus know all p_{rs}. We have to ensure that the ϕ's are such that all p_{rs}'s are consistent, specifically, that the following hold:

$$p_{11} \le p_{1\cdot}, \quad p_{11} \le p_{\cdot 1}, \quad \text{and} \quad p_{1\cdot} + p_{\cdot 1} \le 1 + p_{11}.$$

Replacing by ϕ's and simplifying, we get:

$$\phi_{1\cdot} \le t/\theta, \quad \phi_{\cdot 1} \le t/\theta, \quad \text{and} \quad \phi_{1\cdot} + \phi_{\cdot 1} \le t + \tfrac{\theta}{t}\phi_{1\cdot}\phi_{\cdot 1},$$

apart from the standard constraints $\phi \le t$. The number of bits required to encode $\phi_{1\cdot}$ and $\phi_{\cdot 1}$ is the log of the number of ways in which they can be chosen subject to the constraints. Because the boundaries are smooth, we can use continuous analysis and approximate the number of choices by the *volume* within the feasibility region of the constraints.

There are two cases, $\theta \ge 1$ and $\theta < 1$. In the former case, it is easy to verify that the constraints $\phi_{1\cdot} \le t/\theta$ and $\phi_{\cdot 1} \le t/\theta$ are sufficient to yield a consistent 4-sided coin. The last constraint kicks in when $\theta < 1$. Simplifying notation, we need to find the area within the region bounded by

$$x, y \ge 0 \quad \text{and} \quad x + y \le t + \tfrac{\theta}{t}xy,$$

which evaluates to

$$\int_0^t \frac{t-x}{1-(\theta/t)x}dx = \frac{t^2}{\theta}\left[1 + \frac{1-\theta}{\theta}\ln(1-\theta)\right]. (5)$$

We briefly discuss the computational problem of carrying the exact analysis over to larger itemsets. For $k \ge 3$, the θ-constraint in equation (4) or the volume evaluation in equation (5) cannot be written in closed form in general. Numerical or Monte-Carlo integration may be too expensive.

3.3 Piecewise constant θ segmentation

Given files of diverse length, it is not reasonable to compress them, compare the absolute sizes of the compressed files, and pick the largest one as most complex. The compression *ratio* would be a better indicator of complexity (large ratio of original to final size implies less complexity). Similarly, the code-length of various itemsets using the constant-θ segmentation are not directly comparable; a baseline code-length is needed for each itemset.

One option is to use the unconstrained code-length. it may be either smaller or larger than the constrained code-length. It can be larger in a 2-item situation, for example, when the unconstrained segmentation is induced mostly by sudden changes in only one marginal probability, but the value of θ is the same in all segments, so that the joint probability tracks these

[2]This section can be skipped on first reading.

changes. The unconstrained segmentation will assign a new three-parameter coin to each of these segments. In contrast, the single-θ segmentation lucks out, paying only for two parameters per segment, and paying for θ only once. Conversely, the unconstrained code-length can be smaller if the constrained model was a poor fit to the data. Thus positive differences are interesting, and negative differences are not (but zero is not a very significant notch on this scale).

Another option for the baseline would be a *piecewise constant* θ segmentation. This is a relaxation of the single-θ model class. A piecewise constant θ model first specifies an *outer* segmentation of $(0, T)$. Over each outer segment, it specifies a single value of θ. Then, for each outer segment, it specifies an *inner* segmentation, each assigned to one coin. All inner coins assigned to an outer segment have the same value of θ, the one associated with the outer segment. Finally, other necessary parameters for inner coins are specified as in the single θ case.

Finding a piecewise constant θ segmentation is simple given the previous building blocks. We set up a shortest path problem again. To assign the cost for edge (i, j), we run the constant θ segmentation algorithm on the (i, j) segment of the data. To avoid too many invocations of the constant θ procedure in practice, we can heuristically restrict potential segmentation points to those of the unconstrained segmentation. For efficiency and simplicity we use the unconstrained baseline in our experiments.

4 Algorithm summary

In the previous section we described the building blocks of our technique. Here we put them together to show an overall picture of our system. We proceed from small to large itemsets as usual. However, because our surprise measure is conditioned on the marginals and on variation along time, we have to design pruning strategies different from the simple fixed support filtering commonly used in bottom-up itemset search algorithms (although any such minimum support can also be gainfully used). Thus the outline of our algorithm takes the following form.

One item:

1. Select only those items that can possibly have more than one segment. This is the *pruning criterion* discussed in §5.1.
2. Find unconstrained segments for single items. Also find the code length using only one global coin from start to finish. This is also the (trivial) *constrained* segmentation problem for single items.
3. Order items by the difference, ratio, or relative difference of these code lengths. Display items to user.

Two or more items:

1. From marginals of proper subsets, compute the maximum support the itemset could have along time. Call this the *support envelope.*
2. From the envelope estimate if the itemset could possibly have more than one segment. If not eliminate the itemset. This property of prunability is monotonic with respect to itemset containment, i.e., if an itemset can have at most one segment, no superset can have more. Thus we can use subset-based pruning when it helps.
3. Compute two segmentations of the itemset: the piecewise constant θ segmentation, and the single θ segmentation.
4. Use the difference, ratio, or relative difference of code-lengths to order the itemsets by surprise value.

5 Computational issues

We discuss two important performance issues in this section: how to control itemset expansion via pruning, and how to compute near-optimal segmentations in near-linear time.

5.1 Pruning criteria

We prune an itemset when its support sequence along time is so close to zero that it will not be possible to get more than one segment for any sequence that is enveloped by this sequence. This is superior to an absolute aggregated support based pruning because it can differentiate between the following two sequences both of which have the same aggregated support: One of the sequences has all the ones concentrated in one or two contiguous segments and another has the ones spread uniformly over the entire time base. Clearly, the former is more interesting than the latter. However, if for some economic reasons one is interested only in itemsets with support above a value, we can always include that as an additional filtering step.

This property, like the absolute support based pruning is also monotonic, meaning if an itemset is pruned all its supersets must also be pruned. Therefore, we can apply the "apriori" technique [2] of pruning any itemsets with at least one subsets already pruned, during the candidate generation phase. In addition, we can also prune by looking at the estimated upper envelope on the support sequence of an itemset. This upper envelope is calculated during candidate generation by taking a minimum of the supports of each of the immediate subsets at each point in time. If this upper envelope meets the pruning criteria below, we drop the itemset.

Claim 4 *A sequence $\vec{x}$ of length n cannot be pruned if there is at least a fraction f of the sequence for which the estimate of coin bias p is large enough to satisfy*

$$\begin{aligned} p\log(1/f) \;>\; & \frac{\log n}{nf} - (1-p)\log(1-p) \qquad (6) \\ & +((1/f)-p)\log(1-pf). \end{aligned}$$

Proof. (Sketch) Consider a sequence $\vec{y}$ that is enveloped by $\vec{x}$. The best case in which $\vec{y}$ can have two segments is when it is identical to $\vec{x}$ for some segment (i, j) in time and zero everywhere else. For such a sequence, find two coding costs: C_1 assuming the entire sequence is a single segment and C_2 assuming a separate segment for (i, j). C_1 has higher data encoding cost whereas C_2 has at least one extra segment and thus pays higher parameter and segmentation cost (see §3.1). These costs can be expressed in terms of f, p, n where f, p refer to the parameters of the sequence (i, j). We then use the inequality $C_1 > C_2$ to find the above condition. ■

5.2 Fast shortest path approximations

The second important performance issue concerns the shortest path computation. This is a key subroutine in all our algorithms. For data mining applications with millions of transactions, it is clearly unacceptable to have quadratic complexity along time. In this section we will study how to arrest this quadratic growth and still get a good segmentation.

One immediate way to cut down running time is to pre-aggregate the data. In developing the algorithms, we have been dealing with each market basket as one reading of a random variable, but many data sets are pre-aggregated at daily or weekly levels. Given such data, it clearly makes no sense to segment finer than a day or week. In fact, if the input data is not aggregated at all, i.e. it is a 0-1 sequence, optimality is preserved in first taking run lengths of zeroes and ones. However, care may be needed to aggregate at coarser levels. Larger chunks will cut down running time but may gloss over narrow, interesting segments. Thus, a simple fixed size chunking and pre-aggregation, followed by a single shortest path computation, will not always achieve our goal.

The heuristic: Suppose there are T transactions and thus $T+1$ nodes in the graph. Fix a constant ϵ between 0 and 1 to be decided later and break up the graph into $T^{1-\epsilon}$ chunks, each having T^{ϵ} nodes in it. Now we solve $T^{1-\epsilon}$ instances of shortest path over the chunks, each taking $O(T^{2\epsilon})$ time, for a total time of $O(T^{1+\epsilon})$. Let the nodes on the shortest path in some chunk be called *chosen*. Heuristically, we hope that most chunks are completely contained within some global optimal segment, so that there are very few edges in the local shortest path for each chunk. If this is not the case, the chunk size is too large.

We now construct a sparsified version of the original graph. All $T+1$ nodes are there, but the only edges are those that go from one chosen node to another. If our heuristic assumption above holds, this graph has only $O(T^{1-\epsilon})$ chosen nodes and a final shortest path run on it takes $O(T^{2-2\epsilon})$ time. The total time is thus $O(T^{1+\epsilon} + T^{2-2\epsilon})$, which has the smallest value, $O(T^{4/3})$, for $\epsilon = \frac{1}{3}$, assuming, of course, that this choice yields a small number of chosen nodes. We finally report the shortest path found in the sparse graph.

Analysis. How about the quality of the approximate segmentation compared to the optimal?

Claim 5 *The approximate shortest path has corresponding code length at most twice the optimal.*

Proof. (Can be skipped.) Consider a chunk c having nodes ℓ_c, m_c, r_c, where the global optimal shortest path has edges (ℓ_g, m_c) and (m_c, r_g), whereas the chunk's local shortest path skips from ℓ_c to r_c avoiding

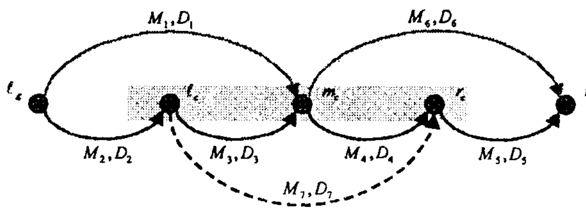

Figure 1: Illustration of approximate shortest paths.

m_c. (Note that ℓ_c, r_c are not necessarily the leftmost or rightmost nodes of c, ℓ_g, r_g could also belong to c, and some of these nodes could be one and the same.) Note the parameter and data encoding costs M_1 and D_1 through M_7 and D_7 in Figure 1. Consider the total cost of the path $(\ell_g, \ell_c, r_c, r_g)$, which is less than the cost of the path $(\ell_g, \ell_c, m_c, r_c, r_g)$, because (ℓ_c, r_c) was preferred by the local shortest path to (ℓ_c, m_c, r_c). We will now compare (ℓ_g, m_c) in optimal with (ℓ_g, ℓ_c, m_c), and (m_c, r_g) in optimal with (m_c, r_c, r_g).

For all edges, the global segmentation cost is the same, $S = \log T$. Let M_1, D_1 be the parameter and data costs for (ℓ_g, m_c) in optimal, and $M_2, D_2; M_3, D_3$ be the maximum likelihood model and data costs for (ℓ_g, ℓ_c) and (ℓ_c, m_c) respectively. Then observe that $D_2 + D_3 \leq D_1$, and $M_2 + M_3 \leq 2M_1$. This is because D_2 and D_3 are ML estimates. Similarly, $D_4 + D_5 \leq D_6$ and $M_4 + M_5 \leq 2M_6$. Thus we have to compare between $2S + (M_1 + D_1) + (M_6 + D_6)$ (cost of the optimal) and $4S + (M_2 + D_2 + M_3 + D_3) + (M_4 + D_4 + M_5 + D_5)$ (cost of approximate). The approximate cost is at most $2S + M_1 + M_6$ larger. Since optimal is at least $2S + M_1 + M_6$ large, this is a factor of at most 2. ■

Consider how the optimal path may interact with a given chunk c. It may never touch any node in c, in which case there is nothing to do. If it does, it enters c at some node and leaves at another (perhaps the same) node. If these nodes are not chosen, we can make the adjustment above to construct a path passing through only chosen nodes that has at most two times the optimal cost. In practice, if ϵ is small enough, the optimal path avoids most chunks and so the cost increase is quite negligible, less than 0.2% in our experiments. It is roughly proportional to the number of hops in the optimal path.

Exploiting marginal segmentations: Another source of information that can be exploited for faster segmentation is the segmentation of marginals. The advantage of using marginals as against a fixed width chunking is that statistically, most swings in the joint distribution are explained by marginals. Note that the heuristic and analysis does not depend on how the initial chunks are derived.

6 Experiments and experience

We developed a prototype and studied real-life market basket data over several years. `Dataset-A` consisted of 2.8 million transactions spanning over seven years from

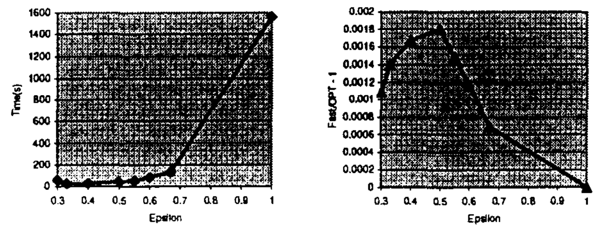

Figure 2: Effect of ϵ on the fast shortest path heuristic. One line shows running time and the other shows quality of the result. If optimal uses b^* bits and the heuristic uses b bits the quantity plotted is $b/b^* - 1$.

1987 to 1993. The time stamps with each transaction was recorded to a granularity of a day. Therefore, each sequence was of length at most 2590. There were 15.8 thousand total number of items and the average length of a transaction was 2.62. Dataset-B consisted of 1.65 million transactions spanning over a period of three years from 1991 to 1993. Here again, the recording granularity was a day yielding a total of 1092 time points for each item sequence. There were a total of 69 thousand items and the average length of a transaction was 1.6.

We describe the performance of our system in §6.1, and in §6.2 we evaluate the quality of the output of our system, using anecdotal evidence as well as quantitative measures.

6.1 Performance

The apparent complexity of our data analyses may evoke questions about practicality. Actually, our method works within very reasonable time. The main potential complexity was in the shortest path computation. This was significantly mitigated by our fast shortest path heuristic. Figure 2 shows how the fast approximation can cut down the time for shortest path computation. Specifically we study the effect of ϵ between $\frac{1}{3}$ and 1. First consider the plot of running time against ϵ. It is evident that for a broad range of ϵ, the time taken by the approximate algorithm is smaller than that of the optimal algorithm by more than an order of magnitude. It is not monotonic: there is a small valley near .4. At small ϵ, there are many chunks but almost never more than two chosen nodes per chunk, i.e. their endpoints. Most of the work is in the final phase. At larger ϵ, there are larger, fewer chunks; there is more work in the first phase, and perhaps more chosen nodes in the second phase.

Next consider the plot of error against ϵ. We plot the ratio of approximate bits to optimal bits minus one. The big message is that the error is tiny, less than .2% typically. Error also shows non-monotonicity. At small ϵ, MDL is extremely unwilling to posit more than one coin per chunk, but this is OK since tiny chunks are unlikely to straddle optimal segment boundaries. As chunk size increases, MDL maintains this stand for some time even as errors accumulate because of straddling. Eventually, MDL gives in and increases chosen nodes, at which point the final phase picks up good paths again. Summarizing, the fast heuristic enables our analysis to execute within reasonable time without losing accuracy.

6.2 Quality of results

In this section we compare the quality of the output of our algorithm with that of simpler or previously known alternatives.

MDL: Our approach as discussed in this paper.

Stat: This is a standard statistical approach where data is first aggregated to some level of granularity (a week by default in our experiments). For each such regular segment, we measure the θ value and find the spread (ratio of standard deviation and mean) of θ over all the segments. We order itemsets based on this measure. We experimented with four other measures of interest: standard deviation of θ, spread of chi-squared values, spread and standard deviation of p-values. We report comparisons with only the spread of θ measure since it showed the best overall results.

MDL-Stat: A shortcoming of the above method is that users need to specify a fixed window size to segment the sequence. The MDL-based method can find the best segmentation automatically. Therefore, we test a hybrid of the two methods above where we first use the MDL approach to get the best segmentation (by solving the unconstrained segmentation problem). Next, we order itemsets on the spread of the θ between segments as in the statistical approach.

Correlation: In this approach, we ignore time and simply calculate the θ value over the entire sequence aggregated to a single point.

We present three kinds of evaluation. First we present anecdotal evidence (§6.2.1) that the top sets output by our method are more interesting than those from the alternatives, and explain why this is the case. Next we undertake a more quantitative evaluation. We consider the top items found interesting by our method and find their positions in the itemset orderings of the alternative methods. We call this the *rank-order* evaluation (§6.2.2). Then, we evaluate the sharpness or selectivity of the different approaches. We consider what fraction of the total itemsets have "high" values of the interest measure. Intuitively, a method where a very small number of itemsets have high values of the interest measure is better than another where the high interest values are spread over large number of itemsets making it harder to define a sharp cut-off. We call this the *selectivity evaluation* (§6.2.3). For these comparisons we consider itemsets of size two; in our experiments with these large datasets we have not found any 3-itemset which is surprising given the marginals of item subsets.

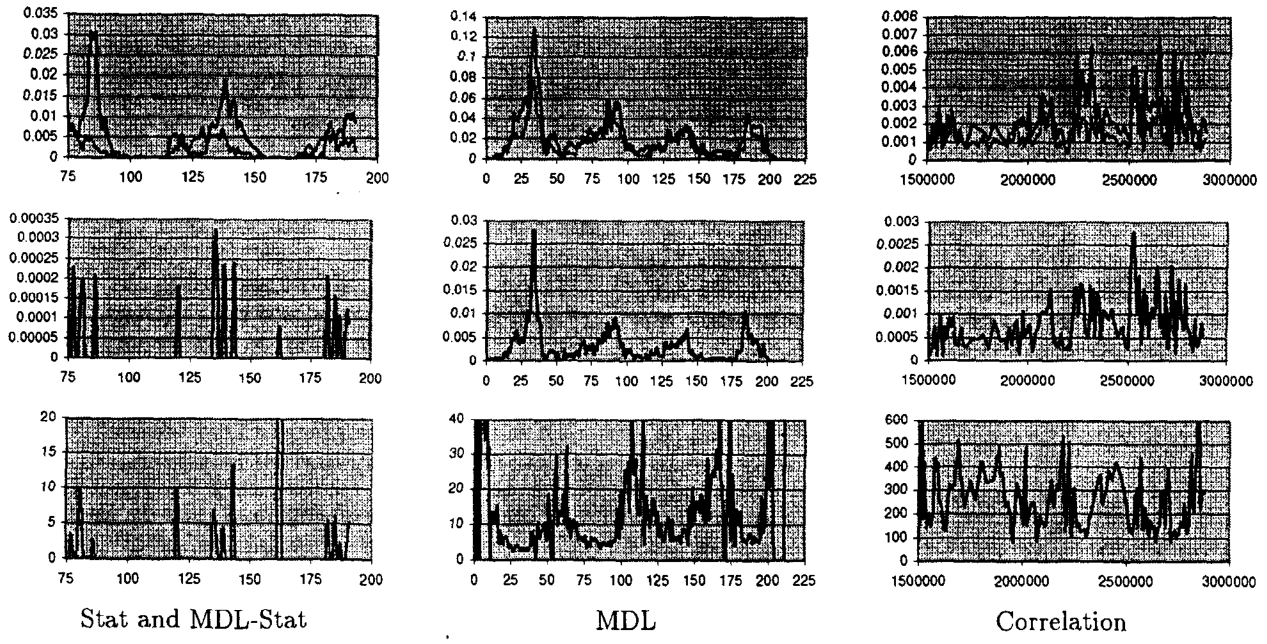

Figure 3: Sequences ranked top by each of the four methods. The x-axis shows time. The top row shows marginals $p_{1\cdot}$ and $p_{\cdot 1}$ averaged over a weekly window. The middle row shows the support p_{11} of both items in a basket and the lowest row shows the θ values.

6.2.1 Anecdotes

In Figure 3 we show various sequences for three pairs of itemsets. The first itemset (first column) was ranked very high by the Stat method and the MDL-Stat methods but was not found interesting by MDL. The second item pair was ranked high by MDL but was ranked very low by all of the other approaches (did not appear in the top several hundred, or 10%, of their ranked list) and the final itemset (third column) was ranked high by Correlation but was not found interesting by MDL.

A quick look shows that θ fluctuates a fair amount for each of the three cases. That is normal. We want to separate statistically significant fluctuations from random noise. A closer look at these sequences shows these differences.

Consider the itemset that was found near the top of the Stat and the MDL-Stat list but was found uninteresting by MDL (first column). The spread of the θ values is high for this sequence because of the high peaks of θ caused due to small support of the marginals. The MDL-based approach is robust to such small marginals since these end up having small code lengths. The items turned out to be complementary: polo shirt and shorts.

Now, consider the itemset that was picked as interesting by MDL but was ranked low by the other two methods (second column). Once we ignore short range noise in θ, we notice that it increases from 2 (almost independent) to around 17 over a steady range. The change happens gradually with the result that the deviation-based measures do not find this interesting. The items were men's and women's shorts. These do not have complementary roles and there is no obvious reason why dependence between these two items should increase with time, and yet the pattern is statistically significant. We are therefore justified in regarding this as a surprising pattern.

For the itemset that was ranked high based on Correlation but not by MDL, we observe that θ fluctuates around a large constant mean of 250, but the fluctuation is small relative to 250, compared to the MDL topper. The item pair turned out to be bedsheets and pillow-cases, which people routinely buy together.

6.2.2 Rank order

As far as the user is concerned, an exact measure of complexity of itemset sequences is less important than the *ordering* it imposes on itemsets. In a typical application, we envision that the user will be shown ranked lists of the most surprising itemsets (by some measure) from single to double to larger itemsets which are not explained by smaller itemsets shown earlier in the list. Accordingly, in this section, we will compare the rankings computed using the various suggested approaches.

6.2.2.1 Comparing rankings: In the graphs in Figure 4 we plot the ranks assigned by MDL (y-axis) against the rank assigned by the Stat and MDL-Stat methods. (To reduce clutter a random collection of 500 itemset pairs are plotted.) First note that there is no strong correlation between the MDL and the Stat method. The correlation coefficient (over all itemset pairs) is 0.02 which is practically close to zero. With better segmentation using the MDL-Stat method the correlation improves from 0.02 to 0.08 as shown by the second figure but the correlation is still not strong enough for te method to be a substitute for MDL. For instance, on zooming to the 100 by 100 grid near the origin we find that only 11 of the top 100 itemsets in the MDL list occur in the top-100 MDL-Stat list.

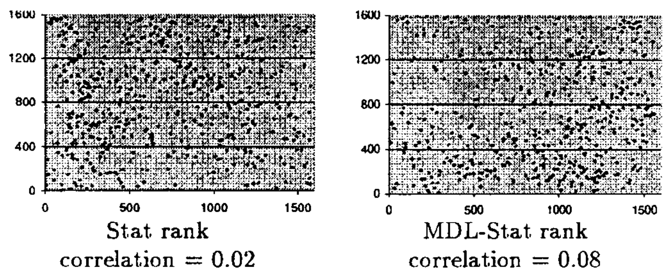

Figure 4: Scatter plots comparing ranks of itemsets in different methods. In both figures the y axis is the rank assigned by MDL.

6.2.2.2 Sensitivity to window sizes of the Stat method.

Another potential problem with the Stat approach is coming up with a good window size over which to compute θ. To show the sensitivity of the result to this parameter in Figure 5 we show the correlation between the Stat method for different values of periodicity with the MDL method. As we increase the window size from 1 week to 4 weeks the correlation with MDL increases but then drop slightly as we increase the window size further. The best correlation is achieved when we use MDL to find the best segmentation as indicated by the extreme point marked "Opt".

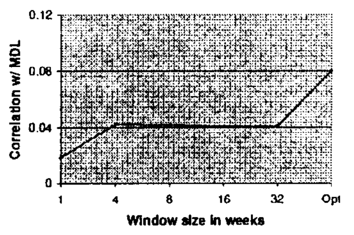

Figure 5: Change in correlation with the MDL approach with difference window sizes.

6.2.3 Selectivity

We give a measure of the selectivity of the different methods in filtering interesting itemsets from the rest. This is related to the sharpness with which the count of the number of itemsets with interest above a threshold falls. In Figure 6 we compare this measure for different methods. The x-axis shows the interest measure used by the method and the y axis shows the count of the number of itemsets above a given interest value. Note that the sharpest gradient is obtained by the MDL method. Out of a total of 1600 itemsets, less than 50 have interest measure greater than zero. The number is significantly higher for the other three methods.

7 Related work

We review related work spanning Statistics, Machine Learning, and Data Mining and discuss how our work compares with existing approaches.

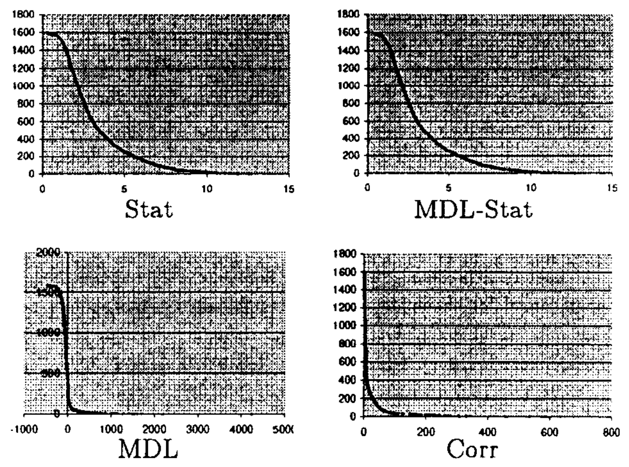

Figure 6: Comparing sharpness of selectivity for the four methods. The x-axis shows the interest value used by that method. The y axis shows the count of the number of itemsets above a given interest value.

Statistics. In principle, our problem does permit a standard statistical approach involving the following steps:

1. Decide on a *model M* of the time series, usually using deep domain knowledge of the process.
2. Choose a *suitable* smoothing window w_1 and estimate the model parameter over many windows where the process can be assumed to be stationary. Also devise a confidence test. Various simplifying assumptions such as normal approximations may be made at this stage.
3. To judge if another (recent) window w_2 shows a shift, estimate its parameters and apply the confidence test to it, reporting deviations more than a *threshold a*.

This approach requires the user to make at least four critical choices: M, w_1, w_2 and a. Significant tuning and domain expertize may be entailed. An approximation that may be valid in one application may behave poorly in another setting. For example, we already saw in Section 6 how different window sizes can give different interest rankings. Such issues are echoed even in textbooks on the subject [5, Page 54]:

> It is difficult to formulate [smoothing] and give a mathematical statistical solution. The practitioner, thus, must proceed on the basis of general experience and intuition ... Smoothing leads to an estimated trend that is descriptive rather than analytic or explanatory. Because it is not based on an explicit probabilistic model, the method cannot be treated fully and rigorously in terms of mathematical statistics.

Our work is an attempt to address this very issue. A system which needs no tuning is closer to the needs of mining systems that must deal with diverse data.

Machine Learning. Our segmentation problem is in some sense a one-dimensional unsupervised clustering scenario. Similar segmentation problems have been addressed by Dom [11] in the context of image segmentation, Rissanen and Shedler [20] in the context of identifying stretches of production or short-lived items in a factory, and Ron and Freund [13] and Blum and Chalasani [7] in the context of learning from a set of distributions. Most of the proposed algorithms are worse than quadratic, and none deal with identifying segments based only on the drift of the *relationship* between variables, i.e., potentially ignoring drifts that are well-explained by drifts in the marginals.

Data Mining. The issue of efficiently updating mining results incrementally is relatively well-studied [9, 24, 3] in the data mining literature. A few recent papers have also addressed the issue of discovering interesting patterns along time for market basket data. Ozden et al present an algorithm for discovering "cyclic" associations rules provided the user *specifies* period(s) and segment size(s) of interest [18]. Lent et al. in [17] discuss how a plot of support versus time for frequent itemsets can be queried to find interesting trends along time. Their methodology is to first partition the data into a fixed number of segments, find support in each of these segments and then provide a query interface for the resulting timeseries as discussed in [4]; querying based on shapes of the time series can be used as a good user interface in front of our system.

8 Conclusion and future work

We have proposed and explored a new approach to extracting *temporally surprising* patterns, as against just *prevalent* patterns, from market basket databases. This is an attempt to substitute the user's domain knowledge and hence effectively eliminate patterns that are already well-known. We used the minimum description length principle together with an appropriate encoding scheme and model class to achieve this. No domain expertise, model selection, or parameter tuning is needed from the user. Experiments with market basket data showed that our method is effective in eliminating prevalent and obvious itemsets (such as milk and cereal), while extracting itemsets with no obvious complementary relationship showing statistically strong variation of dependence along time.

Seasonal variations: Our work opens up many avenues for future exploration. In our experiments, the top ranking itemsets had seasonal marginals, but seasonality of θ was not a major reason for high ranks. Nevertheless, it will be interesting to handle predictable seasonal variation. The current coding scheme has to be extended to recognize simplicity in the regular reuse of coin parameters, perhaps by transmitting an *index* to a known coin from the past, not all its parameters from scratch.

Incremental mining: The attention to time provides a very natural framework for doing incremental mining. Rather than fold new transactions into global estimates of support [9], one can maintain incremental shortest paths and integrate the new segment of data into the existing segmentation.

Acknowledgement. Thanks to Rakesh Agrawal, Trang Nguyen, and Ramakrishnan Srikant for helpful discussions and Martin van den Berg for comments on the manuscript. We used Andrew Goldberg's shortest path library as a starting point for our system.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, December 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 12, pages 307–328. AAAI/MIT Press, 1996.

[3] R. Agrawal and G. Psaila. Active data mining. In *Proc. of the 1st Int'l Conference on Knowledge Discovery in Databases and Data Mining*, Montreal, Canada, August 1995.

[4] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In *Proc. of the 21st Int'l Conference on Very Large Databases*, Zurich, Switzerland, September 1995.

[5] T. W. Anderson. *The statistical analysis of time series*. John Wiley & Sons, Inc, 1971.

[6] Y. Bishop, S. Fienberg, and P. Holland. *Discrete Multivariate Analysis theory and practice*. The MIT Press, 1975.

[7] A. Blum and P. Chalasani. Learning switching concepts. In *Proc. fifth annual workshop on cmputational learning theory*, 1992.

[8] B.S.Everitt. *The analysis of contingency tables*. Monographs on statistics and applied probability 45. Chapman & Hall, second edition, 1992.

[9] D. Cheung, J. Han, V. Ng, and C. Wong. Maintenance of discovered association rules in large databases: An incremental updating techniques. In *Proc. of 1996 Int'l Conference on Data Engineering*, New Orleans, USA, February 1996.

[10] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[11] B. Dom. MDL estimation with small sample sizes including an application to the problem of segmenting binary strings using Bernoulli models. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 1997. longer version: IBM Research Report RJ 9997 (89085).

[12] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discover: and overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthuruswamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[13] Y. Freund and D. Ron. Learning to model sequences generated by switching distributions. In *Proceedings of the Eighth Annual ACM Conference on Computational Learning Theory (COLT)*, 1995.

[14] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.

[15] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Third International Conference on Information and Knowledge Management*, pages 401–407, 1994.

[16] P.-S. Laplace. *Philosophical Essays on Probabilities*. Springer-Verlag, New York, 1995. Translated by A. I. Dale from the 5th French edition of 1825.

[17] B. Lent, R. Agrawal, and R. Srikant. Discovering Trends in Text Databases. In *Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining*, Newport Beach, California, August 1997.

[18] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proc. Int'l Conference on Data Engineering*, 1998.

[19] J. Rissanen. Stochastic complexity in statistical inquiry. *World scientific seies in computer science*, 15, 1989.

[20] J. Rissanen and G. Shedler. Failure-time prediction. Technical Report RJ 9745, IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose CA 95120-6099, 1994.

[21] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):970–974, 1996. Special issue on Data Mining.

[22] C. Silverstein, R. Motwani, and S. Brin. Beyond market baskets: Generalizing association rules to correlations. In *SIGMOD*, 1997.

[23] C. Stedman. Data mining for fool's gold. *Computerworld*, 31(48), Dec. 1997.

[24] P. Utgoff, N. Berkman, and J. Clouse. Decision tree induction based on efficient tree restructuring. *Machine learning journal*, Oct 1997.