Point-Distribution Algorithm for Mining Vector-Item Patterns

Anne M. Denton, Jianfei Wu
Department of Computer Science
North Dakota State University
Fargo, ND
{anne.denton,jianfei.wu}@ndsu.edu

Dietmar H. Dorr
Research and Development
Thomson Reuters
St. Paul, MN
dietmar.dorr@thomsonreuters.com

1. INTRODUCTION

Advances in storage technology have long been driving the need for new data mining techniques. Not only are typical data sets becoming larger. The diversity of available attributes is increasing in many problem domains. This work addresses an aspect of diversity that is often overlooked: Data mining problems commonly involve more than a single set of attributes that fits a vector space model, and may in fact be characterized by multiple separate sets. Those sets provide an additional level of abstraction, and may either derive from separate sets of experiments, or they may have been the result of transformations from different original data types such text, by means of the the bag-of-words model [12], or graphs, through vector space embeddings [24, 23].

The presented work recognizes the cohesiveness of multiple sets of continuous attributes and uses the combinations for further processing. As is commonly done in data mining, machine learning, and statistics, we consider those attributes as dimensions in a vector space. The search for patterns allows focusing on those vector attribute – item set combinations for which a relationship can be shown to exist. The pattern mining can be used as a preprocessing step towards classification with the respective item or item set as class label. Our work differs from conventional approaches in that multiple vector attributes are considered. In contrast to conventional feature selection, our approach determines the usefulness of entire vector attributes, each of which consists of multiple continuous attributes. We focus on data that are already provided in vector format, such as gene expression data, and search for patterns that relate the vector attributes to item attributes, such as those considered in market basket research [1], i.e. attributes that are conceptually binary and represent presence and absence of a particular item with presence typically being less frequent then absence. In the evaluation we show that the pattern mining process finds existing relationships with high accuracy, even when the underlying data has explicitly been chosen to be noisy.

Figure 1 provides an overview of the problem. For simplicity, a one-dimensional setting is chosen, but the concepts of distributions equally apply in higher dimensions. In this figure, the (one-dimensional) vector information determines the horizontal position of an object. Objects are assumed to also have item data. The presence of item 1 is indicated through a filled red circle and the presence of item 2 through a filled green rectangle. Objects that do not show a particular item have an empty circle or rectangle respectively. The
Objects having items

Point Distribution of all data points
Distribution of points with item 1 (strong pattern)
Distribution of points with item 2 (no strong pattern)

Objects
Item 1
Item 2

Figure 1: Schematic of a 1-dimensional data set of points together with two example items. Smoothed point distributions for the overall data set as well as for points having the individual items are shown. It can be seen that the distribution of points with item 1 shows two peaks that are much higher than what would be expected based on the overall distribution (strong pattern exists). The distribution of points with item 2 follows the overall distribution much more closely (no strong pattern exists).

d = 0. One-nearest-neighbor approaches such as \cite{37} can, therefore, not be used.

We use a robust kernel-density-estimation-based technique \cite{27} that makes no assumptions about the relationship between the sampling of the two distributions. Similar techniques are known from kernel-density estimation \cite{27} and kernel-density-based clustering \cite{13, 17}. Instead of a Gaussian kernel, which does not work well in high dimensions \cite{36}, we use a uniform kernel, for which the diameter is directly tied to the properties of the data, leaving no free parameters that a user would have to choose.

2. RELATED WORK

The concept of vector item patterns was first introduced in \cite{38} and has been applied to different data sets and similarity measures in \cite{8}. A related algorithm based on pairwise similarity has been shown to assist in determining relevant functional groups in *Escherichia coli* \cite{9}.

From the data mining perspective, the problem of relating vector to item data can be considered either from the frequent pattern mining or from the classification perspective. Frequent pattern mining algorithms, which were originally developed for item data alone \cite{1}, have been generalized to continuous data \cite{4} and combinations of item and continuous data \cite{33, 29, 5}. These algorithms do not, however, allow considering sets of continuous attributes jointly in the pattern mining process.

From the classification perspective, the concept of a vector-item pattern can be compared with the question of whether classification results are significant \cite{16, 18}: If a class label cannot be predicted based on a set of attributes, we can conclude that those attributes and the class label do not show a pattern. Testing this for a large number of potential class labels, corresponds to the pattern mining problem discussed in this paper, and we use a classification-based technique for comparison purposes. We use a significance test that is based on 2-fold cross validation and treats the confusion matrix \cite{22} as a contingency table.

The vector-item pattern problem is also loosely related to multi-view learning, where the objective is to make predictions based on multiple representations \cite{26} or to use multiple representations for clustering \cite{3} rather than finding which sets of features are most strongly related to which item sets. The problem of relating item sets to multiple continuous attributes is also discussed in the multivariate discretization literature \cite{2}. Note that the presented vector-item pattern mining algorithm does not require discretization and is, therefore, more general.

In bioinformatics applications, the problem of relating continuous attributes, in particular gene expression data to Boolean attributes, such as biological functions, is typically addressed through gene set enrichment analysis, GSEA\cite{35, 34}. In its original form, GSEA takes a single continuous attribute as input and tests whether any one functional category shows comparatively high gene expression overall for genes that have that function or item. A main motivation of this work is to improve statistical significance by considering all genes that have the particular function rather than evaluating the expression of individual genes.

Generalizations have been developed that compare groups of continuous attributes using a hypothesized profile, as discussed in the GSEA documentation, or results of clustering or biclustering \cite{30}. While the latter approach does effec-
3. POINT–DISTRIBUTION ALGORITHM

3.1 Vector and Item Data

The algorithm assumes that multiple \((D)\) continuous attributes are considered to be related based on prior knowledge. Patterns involve all of these attributes \(x_i \in \mathbb{R}, 0 \leq i < D\) together as one “vector” attribute \(\mathbf{x}\) with domain \(\text{dom}(\mathbf{x}) = \mathbb{R}^D\). The set of occurring data points (extant domain) is \(V \subseteq \mathbb{R}^D\). Traditionally, vector space representations are used to describe all independent variables in a particular data mining problem. Our approach differs in that vector attributes are considered as one building block of many.

Item data are conceptually viewed as binary attributes \(B^{(i)}, 0 \leq i < M\), that represent the presence of item \(i\), with \(M\) distinct items occurring in the database. This view is in accordance with the original formalism used by Agrawal et al. [1], and the related downward closure properties of the support of item sets apply to our problem. However, the Kullback–Leibler divergence measure does not provide any such pruning opportunities.

3.2 Overview

To find patterns among vector and item data we perform the following steps

**Normalization:** In contrast to histogram-based techniques that depend on the data having a uniform distribution [38], the Kullback–Leibler divergence does not place specific requirements on the normalization. In the evaluation section, we normalize the vector data by subtracting the mean and dividing by the norm of vector, which is typically recommended for both the gene expression and the time series [14] we use. The algorithm does not, however, critically depend on such a normalization.

**Determine frequent item sets:** Standard downward closure considerations apply to item data.

**Determine overall distribution:** The distribution based on all data points is calculated, using kernel-smoothing. The kernel-width is chosen such that each the average number of subset points within the kernel hypervolume is one. If this item-support-dependent kernel-width is chosen, the overall distribution has to be recalculated for some representative support values.

**Determine subset distribution:** Each item set determines a subset of data points that define the subset distribution. The distribution and the overall distribution are sampled based on the data points with the item.

**Determine Kullback–Leibler divergence:** The Kullback-Leibler divergence is calculated for the subset distribution with respect to the overall distribution.

**Repeat process for multiple vector attributes:** The process can be repeated for different vector attributes.

3.3 Kullback–Leibler Divergence

The Kullback–Leibler, or K–L, divergence is a measure of the similarity of probability distributions. It can be written as

\[
D_{KL}^{(d)}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx_1, \ldots, dx_d
\]

where \(q\) is the reference distribution, which in our case is the full distribution of all objects, and \(p\) is the distribution of objects that have the item of interest. Both distributions are assumed to be defined over a \(d\)-dimensional vector space. Using the law of large numbers this expression can be estimated as follows

\[
D_{KL}^{(d)}(P||Q) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{p(x^{(i)})}{q(x^{(i)})}
\]

where the \(x^{(i)}\) are samples drawn according to the distribution \(p(x)\). The samples that have the item are assumed to be independent and identically distributed, i.i.d..
Since we do not know the probability density from which the observed points are sampled, we have to estimate it. The estimation of a probability density function from a distribution of data points goes back to Parzen [27] and is commonly done by kernel-smoothing. We use a uniform kernel since Gaussian kernels are known to be problematic in high dimensions [36]. The similarity is determined based on whether the product exceeds a threshold. For data that are normalized, such that they lie on a hypersphere of radius one, this results in a uniform kernel that is equivalent one defined based on Euclidean distance. A threshold on a Euclidean distance $t_k$ can be converted to a product threshold $t$ given the normalization

$$ t_k = \frac{\sqrt{(\frac{x}{|x|} - \frac{y}{|y|})^2}}{2} = \sqrt{2 \left(1 - \frac{x}{|x|} \cdot \frac{y}{|y|}\right)} $$

$$ = \sqrt{2(1 - t)} \quad (3) $$

We use the product threshold because the complexity of calculating products is lower than of calculating Euclidean distance. A threshold on a Euclidean distance $t_k$ can be calculated such that the expected number of data points per kernel volume is one. We account for random contributions to the K–L divergence by comparing with randomly created subsets of the data. Nevertheless, one may argue that a choice of kernel-width that removes some of the random fluctuations in $p(x)$ might be beneficial. Our study in Sect. 4.3 shows that there is no empirical basis for this argument and that, in fact, a kernel-width corresponding to one expected subset point per kernel volume does consistently lead to the strongest results. A likely explanation is that fluctuations in random distributions are expected to be relevant at all length scales.

The number of available samples for $p(x)$ and $q(x)$ typically differ widely, since $p(x)$ only has as many available samples as there are objects with the item. This number is normally much smaller than the total number of objects. We use the same kernel function for $q(x)$ as for $p(x)$ to ensure that those data points that are considered for $q(x)$ include the samples considered for $p(x)$. Hence, there are typically many more than one data point within the kernel volume for $q(x)$.

Choosing a larger kernel width does not fundamentally resolve any of the randomness. One may still be concerned that even a random distribution of data points can show large variations in $p(x)$ if the expected number of points per kernel volume is one. We account for random contributions to the K–L divergence by comparing with randomly created subsets of the data. Nevertheless, one may argue that a choice of kernel-width that removes some of the random fluctuations in $p(x)$ might be beneficial. Our study in Sect. 4.3 shows that there is no empirical basis for this argument and that, in fact, a kernel-width corresponding to one expected subset point per kernel volume does consistently lead to the strongest results. A likely explanation is that fluctuations in random distributions are expected to be relevant at all length scales. Choosing a larger kernel width does not fundamentally resolve any of the randomness.

### 3.4 Choice of Kernel Width

We choose a kernel width such that one data point is expected within the hyper-volume that is covered by the kernel, see Fig. 3. The rationale for this choice is that once more than one data point is expected within the kernel volume, local properties are lost. A smaller kernel volume, on the other hand, would result in fluctuations of the probability density even for data points that are approximately equidistant. One may still be concerned that even a random distribution of data points can show large variations in $p(x)$ if the expected number of points per kernel volume is one. We account for random contributions to the K–L divergence by comparing with randomly created subsets of the data. Nevertheless, one may argue that a choice of kernel-width that removes some of the random fluctuations in $p(x)$ might be beneficial. Our study in Sect. 4.3 shows that there is no empirical basis for this argument and that, in fact, a kernel-width corresponding to one expected subset point per kernel volume does consistently lead to the strongest results. A likely explanation is that fluctuations in random distributions are expected to be relevant at all length scales. Choosing a larger kernel width does not fundamentally resolve any of the randomness.

The number of available samples for $p(x)$ and $q(x)$ typically differ widely, since $p(x)$ only has as many available samples as there are objects with the item. This number is normally much smaller than the total number of objects. We use the same kernel function for $q(x)$ as for $p(x)$ to ensure that those data points that are considered for $q(x)$ include the samples considered for $p(x)$. Hence, there are typically many more than one data point within the kernel volume for $q(x)$.

Following these assumptions, the threshold parameter $t$ is to be calculated such that the expected number of data points is $f = 1/N_{i}$, where $N_{i}$ is the support of the item set in question. Fig. 4 shows the setup: We calculate the surface of the hypersphere cap in $d$ dimensions by integrating over contributions for those values of $d$ that are smaller than the threshold. For each value of $d$, the surface of a hypersphere in $d-1$ dimension with radius $r$ contributes to the integral. For example, if the total density were three the points of...
constant $\theta$ lie on a circle of radius $r$. Since the circle itself is a 1-dimensional structure it is denoted as $S_1$. The surface of the cap can be calculated by integrating over the circles for all values of $\theta$ that satisfy the t-threshold. In $d$ dimensions we get

$$S^{(\text{cap})}_{d-1} = \int_0^{\theta_1} S_{d-2} \, r^{d-2} \, d\theta = \int (d-1) \, C_{d-1} \, r^{d-2} \, d\theta$$

(10)

where $S_{d-2}$ is the surface of a hypersphere of radius 1 in $d-1$ dimensions. $C_{d-1}$ is the volume of a unit hypersphere in $d-1$ dimensions. Using

$$r = \sin \theta$$

$$\frac{dr}{d\theta} = \cos \theta$$

$$dr = \sqrt{1 - r^2} \, d\theta$$

(11)

we can rewrite $S^{(\text{cap})}_d$ as follows assuming threshold $t > 0$:

$$S^{(\text{cap})}_{d-1} = d \, C_{d-1} \int_0^{\sqrt{1-t^2}} \frac{r^{d-2}}{\sqrt{1-r^2}} \, dr$$

(12)

This integral can be performed analytically in Matlab. Calculating the hypersphere volume $C_{d-1}$ can be avoided since only the ratio $S^{(\text{cap})}_{d-1}/S_{d-1}$ is needed, which can be evaluated by integrating equation (12) up to $t = 0$. For "support" being the absolute item set support we get

$$\frac{1}{\text{support}} = \int_0^{\sqrt{1-thresh^2}} \frac{r^{d-2}}{\sqrt{1-r^2}} \, dr$$

(13)

In practice, are interested in $\text{thresh}$ as a function of "support". However, eq. (13) cannot be inverted analytically. We use an interval-halving algorithm for inverting it numerically.

### 3.5 Algorithm Outline

The complete algorithm is summarized in Algorithm 1. Array-based notation similar to what is used in Matlab is assumed. The algorithm takes the data points as input. It is assumed that frequent item sets have been determined previously. For simplicity, we also assume that the presence of items and item sets is provided in a binary array. It would be straightforward to modify the algorithm to use lists of items, but such an implementation is less concise in Matlab. It is also assumed that a cutoff of the K-L divergence has been determined from random subsets of the data. We use the mean plus twice the standard deviation over a set of random filters as cutoff.

Next, an array of similarity values is determined. In the evaluation, data set sizes were small enough to keep this array in memory; but the algorithm could easily be rewritten, to avoid storing the full array. Lines 5-8 in Algorithm 1 depend somewhat on the item set support and are, therefore, performed for representative support values. It will be shown, that the representative support can differ from the actual support by a factor of two without substantially affecting results.

The similarity threshold is then determined, given the item set support, using eq. (13) and the interval halving process motivated there. The binary matrix of those point combinations that satisfy the similarity threshold ($\text{sim}$) is determined. Summing over the rows of this matrix (line 8) gives the overall density. The subset density for the given item set is determined by selecting only those rows that have the particular item set (line 12). The K-L divergence for the respective item set is then calculated.

### 3.6 Comparison with Classification-based Algorithm

While the objective of this paper is not directly classification, it is nevertheless possible to use classification algorithms to derive corresponding results. To do so, items are considered to be class labels. If the item can be successfully predicted based on the vector data using classification then it is fair to assume that there is a relationship between the vector data and the particular item. This process has to be repeated for each item attribute. We consider classification to be successful if the confusion matrix, treated as a contingency table, shows a significant relationship between actual and predicted class labels on the test set. We use 2-fold cross-validation to derive the confusion matrix and determine significance based on $\chi^2$ testing. Since elements within the confusion matrix may be small, the Yates correction [39] is applied.

Notice that classification-based measures, such as accuracy, are also used in Section 4.2. In that section a data set is constructed such that some items are expected to show a significant relationship to the vector data, while others don’t. Such an analysis is performed both for the classification-based comparison and for the point-distribution algorithm and should not be confused with the classification-based algorithm discussed in this section. For the classification-based comparison algorithm, accuracy is not directly of interest, since the entire confusion matrix is necessary to determine whether the classification is successful.

### 3.7 Comparison with Previous Approaches

The histogram-based approaches of [38] and [8] that are used as comparison algorithms have the same objective as...
this paper: to identify item sets that show a significant pattern with respect to a vector attribute. In those papers the subset distributions are summarized using histograms. Those histograms are compared with expected histograms that are either derived through resampling or through a theoretical model. The comparison runs in this paper all use resampling, which results in higher accuracy but slower speed. In [38] similarity is evaluated using a subspace-based similarity measure, and [8] also presents results using a product similarity measure. Results for both similarity measures are presented in the comparison.

4. IMPLEMENTATION AND EVALUATION

The algorithm was implemented in MATLAB. A bitvector representation was used for items because of the conciseness of array-based implementations MATLAB. We did not experience memory constraints for any of the data sets in the evaluation. Performance results are based on running the code on a Mac power book with a 2.6 GHz Intel Core 2 Duo Processor with 4GB 667 MHz SDRAM memory in a Vista shell using VMware Fusion. For the classification-based comparison the Classification Tree classifier within the MATLAB Statistics Toolbox is used and significance is evaluated based on 2-fold cross-validation. A \( \chi^2 \) significance test with Yates correction is used on the confusion matrix as contingency table. A \( p \)-value smaller than 0.05 is considered to indicate a significant result.

4.1 Evaluation on Gene Expression Data

The algorithm is first evaluated on gene expression data sets from cell cycle experiments on yeast [32], which are available for download at [31]. Results from four different types of experiments are available, each of which includes measurements at between 14 and 24 time points. The results come from different types of experiments, but all of them show cell-cycle-related time-dependent gene expression. We only consider genes, for which all four experiments are reported which leaves 5878 of the total of 6178 genes. Item data were extracted from Interpro database that contains information on many types of sequence signatures including protein domains and motifs [25]. For simplicity, we refer to all sequence signatures as domains. Yeast domain information was downloaded from [6]. We only consider domains with at least 10 instances, leaving 432 domains.

The goal of the analysis in this section is to identify protein domains, that are significantly related to cell cycle activity. Conventionally, biologists would either look at the lists of differentially expressed genes, and try to find domains or functions that occur particularly often within those sets. Alternatively, they might use clustering or biclustering techniques and then test those clusters for enrichment. Both types of reasoning correspond to two-step approaches, in which the expression data are first analyzed with traditional techniques and relationships with domain or functional data are evaluated based on the already completed evaluation of the first step. Our approach, in contrast, directly searches for patterns between the expression data and domain or functional information respectively.

Since we do not know, which domains are expected to be significantly related to the cell cycle, we limit the discussion in this section to a comparison among the four different data sets, as well as the combination of all four. For a quantitative evaluation against an independent standard we refer to the next section on time series data. Table 1 shows, in its diagonal, the total number of domains that are considered significant according to the experiments. The "All" entry refers to the combination of all four experiments (73 columns) and "Alpha" (18 columns), "Cdc15" (24 columns), "Cdc28" (17 columns), and "Elu" (14 columns) are used as in the description of the original experiments. The overlap in results is shown in the upper right corner of the table. The lower left corner shows whether that overlap is significant according a \( \chi^2 \) significance test using Yates correction. The largest \( p \)-value is 0.02, which indicates that all values for overlap are significant. Several of the entries have such low \( p \)-values that Matlab reported 0. The \( p \)-values are more consistently low than those reported using histogram-based techniques [38].

The consistency between results from different experiments suggests that the results do indeed represent patterns that are biologically relevant. The comparison with the full set of all experiments is primarily included to test whether the algorithm is capable of extracting meaningful results for high-dimensional data. It should be expected that results are consistent between a set of experiments and one of its subset. This comparison leads to \( p \)-values below the Matlab resolution in all cases, confirming that algorithm works well for the 73 dimensions of the full data set.

### Table 1: Results for Gene Expression Data

<table>
<thead>
<tr>
<th>All</th>
<th>Alpha</th>
<th>Cdc15</th>
<th>Cdc28</th>
<th>Elu</th>
</tr>
</thead>
<tbody>
<tr>
<td>259</td>
<td>114</td>
<td>117</td>
<td>160</td>
<td>166</td>
</tr>
<tr>
<td>0</td>
<td>119</td>
<td>67</td>
<td>85</td>
<td>79</td>
</tr>
<tr>
<td>7E-15</td>
<td>3E-12</td>
<td>134</td>
<td>86</td>
<td>72</td>
</tr>
<tr>
<td>0</td>
<td>2E-16</td>
<td>7E-12</td>
<td>173</td>
<td>107</td>
</tr>
<tr>
<td>0</td>
<td>1E-7</td>
<td>0.027</td>
<td>198</td>
<td>5E-8</td>
</tr>
</tbody>
</table>

### Figure 5: Schematic showing how the time series data set is designed. Each window (after preprocessing) corresponds to one vector. Items that are constructed by assigning a label to those vectors that come from one particular time series are expected to be significant (time series items). Items that constructed by randomly assigning labels (randomized items) are not expected to be significant.

4.2 Quantitative Evaluation on Time Series Data

In this section to a comparison among the four different data sets, as well as the combination of all four. For a quantitative evaluation against an independent standard we refer to the next section on time series data. Table 1 shows, in its diagonal, the total number of domains that are considered significant according to the experiments. The "All" entry refers to the combination of all four experiments (73 columns) and "Alpha" (18 columns), "Cdc15" (24 columns), "Cdc28" (17 columns), and "Elu" (14 columns) are used as in the description of the original experiments. The overlap in results is shown in the upper right corner of the table. The lower left corner shows whether that overlap is significant according a \( \chi^2 \) significance test using Yates correction. The largest \( p \)-value is 0.02, which indicates that all values for overlap are significant. Several of the entries have such low \( p \)-values that Matlab reported 0. The \( p \)-values are more consistently low than those reported using histogram-based techniques [38].

The consistency between results from different experiments suggests that the results do indeed represent patterns that are biologically relevant. The comparison with the full set of all experiments is primarily included to test whether the algorithm is capable of extracting meaningful results for high-dimensional data. It should be expected that results are consistent between a set of experiments and one of its subset. This comparison leads to \( p \)-values below the Matlab resolution in all cases, confirming that algorithm works well for the 73 dimensions of the full data set.
Secondly, the algorithm is tested on time series subsequence data from the same nine time series as used in [7]. Subsequences are extracted from the buoy sensor, balloon, glass furnace, steamgen, speech, earth quake, ocean, and darwin of the UCR time series repository [20]. Descriptions of these data sets are distributed with the data. One time series (ecg) was collected independently: The Ecg series (MIT-BIH Arrhythmia Database: mitdb100) originates from PhysioBank [15]. The same preprocessing steps were done as in [7]: The buoy sensor series was compressed by averaging over 4 consecutive values and the ecg series by averaging over 20 consecutive values.

Figure 5 shows the design of the time series data set. Nine items are constructed, such that they can be expected to show a vector-item pattern with the subsequence data. Each one of these items is associated with one of the original time series. The binary item variable is constructed such that it is 1 for the subsequences of its associated time series and 0 for all other subsequences, including the random walk subsequences. We will refer to these items as time series items. Nine other items, which we call randomized items, are constructed through random selection of subsequences. Using this setup we do a quantitative evaluation of the algorithms using classification-style measures, in which time series items are the positive examples and randomized items are the negative examples. It is important to understand that although this evaluation uses classification concepts there is no training involved. Time series subsequences were chosen since time series subsequence data are notorious for being hard to cluster meaningfully [21]. The averaging over 9 series is furthermore expected to make the results generalizable.

Subsequences are extracted using a sliding window with $w = 17$, and differences between successive data points are taken, resulting in a 16-dimensional vector space, i.e. $2^4$ dimensions. $Npts$ subsequences that are randomly chosen from the first 1000 windows of each time series are used from each time series, i.e. the absolute support of all items is $Npts$. In total $Nsets*Npts$ subsequences are non-random, where $Nsets = 9$ is the number of time series. These are combined with $Nrand*Nsets*Npts$ random subsequences, where $Nrand$ is a parameter that is varied from 0 to 8.

$Nsets = 9$ positive items are constructed from the original time series and $Nsets$ are constructed assigned by randomly selecting sequences. Averages over 50 such runs are reported. Unless otherwise stated, the product algorithm chooses a threshold of one expected data point with item per kernel hypervolume. In general, accuracy is reported, although Fig. 8 also shows sensitivity and specificity. Data sets are constructed such that the number of positive and negative items is equal, thereby avoiding problems that are encountered when the class label is skewed.

4.3 Effectiveness

First we vary $Npts$ and keep $Nrand = 2$, i.e. two batches of $Nsets * Npts$ points are added in each case. That means that the absolute support is varied, while the relative support is kept constant (1/27 $\sim$ 0.037). Fig. 6 shows the accuracy in comparison with algorithms described in [38] and [8] and the classification-based comparison approach discussed in Section 3.6. It can be seen that the K-L-based algorithm
outperforms all comparison algorithms. Next we keep the item support fixed at $Npts = 40$ and vary $nRand$. Fig. 7 shows that the K-L-based approach outperforms all comparison algorithms.

Fig. 8 shows the dependence of accuracy, sensitivity, and specificity on the choice of threshold for $Npts = 40$ and $nRand = 2$. The parameter choice indicates how many points with item set are to be expected per kernel volume. The plot confirms that the accuracy is highest for one expected point and falls off quickly for smaller choices. For larger choices it decreases more slowly. Fig. 8 also reports sensitivity and specificity separately, showing that it is the sensitivity that most strongly depends on parameter choice, while the specificity remains stable over the tested parameter range.

### 4.4 Efficiency

The design of the algorithm was, so far guided by the goal of achieving accuracy more then efficiency. For the data set sizes used in the evaluation, the execution time is nevertheless comparable with histogram-based approaches as well as the classification-based comparison using the Matlab tree-classifier, as can be seen from Fig. 9. Notice that for the classification-based comparison, a fast tree-based classifier was used.

### 5. CONCLUSIONS

We have introduced an algorithm to mine vector-item patterns using Kullback-Leibler distributions of kernel densities. We have demonstrated that the resulting algorithm is far more effective than previous algorithms, which summarized point distributions first and then compared the histogram summaries. We have also shown that the approach is more effective at determining vector-item relationships than directly determining the significance of classification. We have discussed the choice of the only parameter in the algorithm, in particular the width of the kernel function, and validated the choice empirically. We have also shown that in a practical application to gene expression data, our resulting patterns are consistent among biologically distinct experiments.

### 6. REFERENCES


[16] P. Golland, F. Liang, S. Mukherjee, and


