Disjunctive Anomaly Detection: Identifying Complex Anomalous Patterns

Author

Robin Sabhnani

Committee Members

Artur Dubrawski
Jeff Schneider
Aarti Singh
Gregory Cooper (University of Pittsburgh)

Machine Learning Department
Carnegie Mellon University

August 23, 2010

Abstract

The problem of anomaly detection in multivariate time series data is common to many applications of practical interest. A few examples include network intrusion detection systems, manufacturing processes, climate studies, syndromic surveillance, video stream processing, etc. Our motivating application is syndromic surveillance that aims to detect potential disease outbreaks in pre-diagnosis data to facilitate timely public health response. To achieve this goal, efficient data structures and smart algorithms are needed to analyze highly multivariate temporal data.

We focus on multivariate datasets that additionally include a set of categorical values for every covariate present in the data. In fact, any time-stamped event or transactional data with categorical features is a relevant dataset. Example categorical features (or dimensions) in syndromic surveillance datasets include age group, gender, location, disease, etc. We define anomalies in terms of these categorical dimensions. A typical disease outbreak affects observed frequencies a subset of values along each dimension giving rise to an anomalous cluster. To search for these anomalous clusters, traditional algorithms either limit the number of values affected along each dimension or simply focus only on spatial dimensions ignoring potentially useful relationships among the non-spatial dimensions. Such constraints are imposed because the search space quickly grows both in the number of data dimensions and in the number of values along each dimension. Also, multiple disease outbreaks do occur simultaneously in the real-world. The state-of-the-art multivariate algorithms focus on finding the most anomalous cluster (as a potential disease outbreak) and then rely on re-executing the algorithm to find the remaining clusters. Such an algorithm can perform poorly if there is a significant overlap in the affected dimensions making it nearly impossible to iteratively isolate different outbreaks at each induction step.

In this thesis work, we introduce Disjunctive Anomaly Detection (DAD), an algorithm for detecting complex anomalous clusters in multivariate datasets with categorical dimensions. Our proposed algorithm assumes that an anomalous cluster can affect any subset of values (disjunction) along each data dimension. We believe that such a cluster definition is more informative of the real outbreaks as compared to the current approaches. In addition, the DAD algorithm models multiple anomalous clusters simultaneously, hence promising better detection power in the presence of multiple overlapping anomalous events. We focus on both detection and characterization of anomalies in such multidimensional and multivariate data sets. We compare the DAD algorithm against the relevant powerful alternatives on two important tasks: finding sample-variable associations in cancer microarray data, and searching for the emerging disease outbreaks in public health data. Experimental results indicate that DAD is able to detect and explain complex anomalous clusters better than the alternative approaches such as the Large Average Submatrix (LAS) algorithm and the What's Strange About Recent Events (WSARE) algorithm.

To assist in the development of future complex multidimensional and multivariate algorithms (including extensions to DAD), we also propose using the T-Cube data structure that efficiently represents any time series data with multiple categorical dimensions (typical in many fields of application including surveillance). The T-Cube data structure acts as a cache and quickly responds to any ad-hoc queries during an investigation. It enables processing of millions of time series during massive data mining operations. We have successfully applied T-Cube to mine interesting patterns in diverse projects involving temporal event data.

List of Figures

List of Tables

Contents

1 Introduction

In many practical situations, such as syndromic surveillance or gene microarray studies, data analysts are faced with a daunting task of identification and characterization of various anomalous patterns in data. The difficulty is often exacerbated by the complexity of the multi-dimensional multi-valued data, and by the multiplicity of potentially convoluted and overlapping processes which produce these observations.

For instance, syndromic surveillance systems monitor records of patient admissions to hospital emergency rooms, daily volumes of over-the-counter medication sales, records of school absenteeism, etc., for any unusual activity which might indicate an emergence of a disease outbreak. Accurate and timely detection of such advere events is the key to the mitigation and containment of their consequences. Such an event can affect any subset of the data dimensions and also multiple values along each dimension, making the identification task challenging. Very few detection algorithms can search the large hypothesis space comprehensively, as it involves monitoring millions of distinct hypotheses even in a reasonably-sized dataset.

To motivate the problem let us consider the Sri Lanka Minstry of Health data [46]. It records patient visits across 26 regions in the country and 9 reportable diseases. Figure 1 shows one possible subset of the data in the form of 6 distinct time series between June 2008 and August 2008. Each time series represents the daily counts of patients being diagnosed with a specific disease in a particular region. Food poisoning cases in the regions of Nuwara Eliya, Gampaha and Badulla are labeled as *fp-ne* (solid red), *fp-gam* (solid blue), and *fp-bad* (solid green) respectively. Similarly, leptospirosis cases in the regions of Kurunegala, Gampaha, and Colombo are labeled as *lep-kur* (dashed red), *lep-gam* (dashed blue), and *lep-col* (dashed green).

It is visually obvious from Fig. 1 that a food poisoning outbreak occurred in early August followed by a leptospirosis outbreak in late August. In fact, if the aggregated time series across the three food poisoning regions was monitored daily, we could have detected the outbreak within a few days of its onset and hopefully prevented its spread into Badulla region. Unfortunately, a disease outbreak can affect any subset of the regions and tracking all potential subsets is computationally infeasible. We also cannot rely on monitoring all of the 26×9 time series separately using some univariate algorithm as the signal might be too weak in any individual series to warrant an alert. Even if we assume that an outbreak can affect up to 4 regions and up to 2 disease types, a naive algorithm will need to monitor $(\sum_{i=1}^{4} \binom{26}{i})(\sum_{i=1}^{2} \binom{9}{i}) = 805,545$ distinct hypotheses in the Sri Lanka data. These computations become prohibitively expensive and intractable as the number of dimensions and number of values along each dimension increase. Hence, there is a need for an efficient algorithm that can search across exponentially many outbreak hypotheses in the data.

Figure 1 also shows that the two disease outbreaks overlap both in time (mid August) and space (region Gampaha). Specifically, *fp-ne*, *fp-gam*, *lep-gam*, and *lep-kur* overlapped during early part of August. This was followed by overlapped signals of *fp-bad*, *lep-kur*, and *lep-gam* during mid August. Finally, the leptospirosis outbreak peaked in all three regions by the end of August. Such overlapping outbreaks can make early detection and accurate identification very difficult for the currently used algorithms.

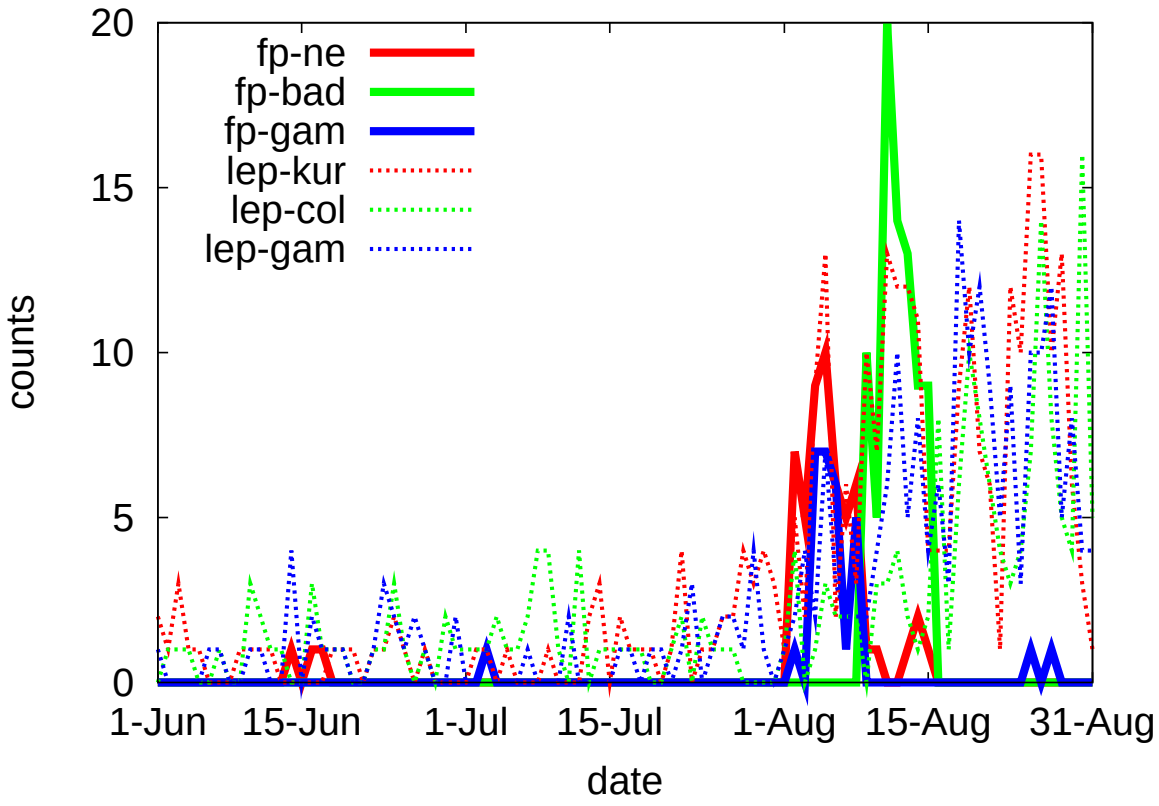


Figure 1: Subset of the Sri Lanka data

The disease outbreak detection in Sri Lanka data can also be thought of as a multivariate anomaly detection problem. There are $26 \times 9 = 234$ covariates each with a distinct time series of daily patient counts. Most multivariate anomaly detection algorithms find periods of time with significantly elevated counts. The key difference between multivariate analysis and our research is to go one step further and characterize the anomalies in terms of the categorical dimensions associated with each covariate. In Sri Lanka data, the two categorical dimensions are region and disease type. The goal of our work is to not only find the time periods of anomalies but also to define them using the categorical dimensions.

The sample-variable association problem in microarray analysis is another relevant application. It involves searching for interesting clusters over large subsets of discrete variables. The data is typically arranged in the form of a matrix with columns denoting samples (usually in hundreds) each collected from a distinct patient and rows representing thousands of measured genes. The goal is to simultaneously cluster both dimensions (samples and genes) to find a subset of genes that are all active in a group of samples. Such clusters may reveal useful insights for the design of novel drugs. The term sample-variable association refers to the association between a set of genes and a group of samples (or patients) and is defined by a bicluster in the data. These biclusters often overlap, sharing a set of genes and/or samples between clusters, in real-world. Very few algorithms exist in the literature that model and search for overlapping biclusters.

The two motivating applications, namely syndromic surveillance and microarray analysis share

a similar objective: to find multiple overlapping anomalous clusters defined over a set of values along different categorical dimensions. We believe that better detection and characterization of complex anomalies is possible as compared to state-of-the-art algorithms using richer hypothesis space (anomalies defined as disjunctions along each categorical dimension and conjunctions along different dimensions). We further claim that it is possible to build approximate algorithms that can efficiently search for such anomalies in many real datasets.

In Sect. 2, we provide a brief literature survey of the related work. Our proposed algorithm along with the experimental results obtained so far are presented in Sect. 3. We then describe an efficient data structure, called the Time Series Cube (T-Cube) in Sect. 4 to aid in the development of complex algorithms that need to frequently analyze millions of distinct time series. We discuss our proposed future ideas and the thesis timeline in Sect. 5. Finally, Sect. 6 summarizes our conclusions so far.

2 Related Work

Traditional syndromic surveillance algorithms detect disease outbreaks by looking for peaks in the distinct univariate time series data. In the case of the Sri Lanka data (a subset of which is shown in Fig. 1), univariate analysis would consider each of the 26×9 time series individually without any aggregation. Such methods tend to perform poorly when the signal magnitude in separate time series is too weak to warrant alerts. Moreover, univariate analyses offer limited explanatory power and they often provide limited information on the impact factors during public health investigations.

There has been a range of work towards multivariate time series analysis. Use of linear dynamical systems to model multivariate time series data and possibly for anomaly detection has been described in [41]. Multivariate anomaly detection algorithms for temporal data, both supervised [9] and unsupervised [7], have been proposed for numerous applications. Our goal is to combine detection of anomalies with their characterization in terms of categorical dimensions.

In the case of syndromic surveillance, there are very few algorithms that truly perform multidimensional and multivariate analysis when searching for the disease outbreaks in the data. What's Strange About Recent Events (WSARE) [47] and Multivariate Bayesian Scan Statistic (MBSS) [30] are two notable exceptions explained briefly below. Both algorithms enable early detection of emerging patterns. They have been applied to datasets containing temporal, spatial, demographic and symptomatic dimensions.

WSARE performs a rule-based pattern search for early disease outbreak detection. The dataset contains multiple categorical dimensions and exactly one temporal dimension. The rules are made of multiple components, joined by a conjunction, each of the form $X_i = V_j$ where X_i is the i^{th} dimension and V_j is the j^{th} value in that dimension. Using the time dimension, WSARE searches for such conjunctive rules (bounded in complexity by up to K components) that have shown significant change in the data distribution as compared to the baseline data. The baseline data is usually the last few weeks prior to the period of concern or analysis. WSARE uses Fisher's exact test to score each rule and then uses randomization to report the significance of the best rule. WSARE also proposes greedy heuristics to ease the computation problem of searching over

exponentially many rules. Our algorithm improves upon WSARE in at least two ways. First, our clusters, unlike WSARE rules, are defined over a set of dimensions where each dimension can take multiple values (joined by disjunctions). Second, we model and search for multiple overlapping clusters simultaneously whereas WSARE reports the one best anomalous rule in the data and then re-executes the algorithm to find additional clusters. We compare the performance of WSARE to DAD in Sect. 3.2 on the Sri Lanka data.

Scan statistic algorithms work with data arranged on a set of spatial locations. The goal is to find a contiguous anomalous region containing a set of locations such that the data distribution inside the region is significantly different than outside of it. Temporal distribution of counts at each location is typically modeled using Poisson distribution whose parameters are estimated from historical data. The Multivariate Bayesian Scan Statistic (MBSS) algorithm extends the standard scan statistic algorithm to use Bayesian graphical model for event detection and characterization in multivariate spatial time series data. The MBSS models temporal counts using Poisson model with hyper parameters defined using Gamma distribution. Again, the hyper parameters are estimated using historical data. MBSS evaluates the posterior probability of observing a disease outbreak as new data arrives in time. The outbreak definitions are restricted to pre-defined types of events along the non-spatial dimensions and to contiguos sets of regions along the spatial dimension. A pre-defined event is a process that affects the different non-spatial data dimensions in a known probabilistic manner and it needs to be defined by the user in advance. Our proposed method, DAD, relaxes the assumption that each outbreak occurs in a contiguous region. The motivation being that far away locations might be linked by some underlying latent variable such as tourist spots, presence or absence of rivers, part of distribution networks, urban or rural regions, etc. that might explain the cause of the anomalies. Such disease outbreaks are not in the search space of the MBSS algorithm. Also, unlike MBSS, DAD does not require any pre-defined event distributions over the non-spatial dimensions. DAD automatically searches for anomalous clusters that can be written as disjunctions over the dimension values and conjunctions over the data dimensions.

We have recently found some related work in the domain of online analytical processing (OLAP). Subspace Iterative Time-Series Anomaly Search (SUITS) algorithm [24] searches for top-K subspace anomalies in multidimensional time series data. The anomalies are defined as changes in temporal trends, magnitude, or phase. SUITS has been applied to grocery sales data with 20-100 categorical dimensions to find interesting sales patterns. Similar to WSARE, SUITS restricts the definitions of anomalies to contain either all or at most one value in each categorical dimension. SUITS heuristics might potentially inspire scaling-up our algorithm to very high dimensional datasets. This will be part of the proposed future work.

Biclustering methods search for submatrices U in the data matrix X whose entries meet some pre-defined criteria. Note that the rows and columns in U may not be contiguous in X. Some example criteria include: rows of U are approximately equal to each other [2]; columns are approximately equal [17]; the elements of U fit a 2-way ANOVA model [10]; the rows of U have equal [5] or approximately equal [25] rank statistics; all elements of U are above a given threshold [31]; and the average of the entries of U is a large positive or a large negative number [40]. Shabalin et al. [40] proposed Large-Average-Submatrix (LAS) algorithm to find possibly overlapping submatrices (or clusters) with a large positive average or a large negative average in microarray data.

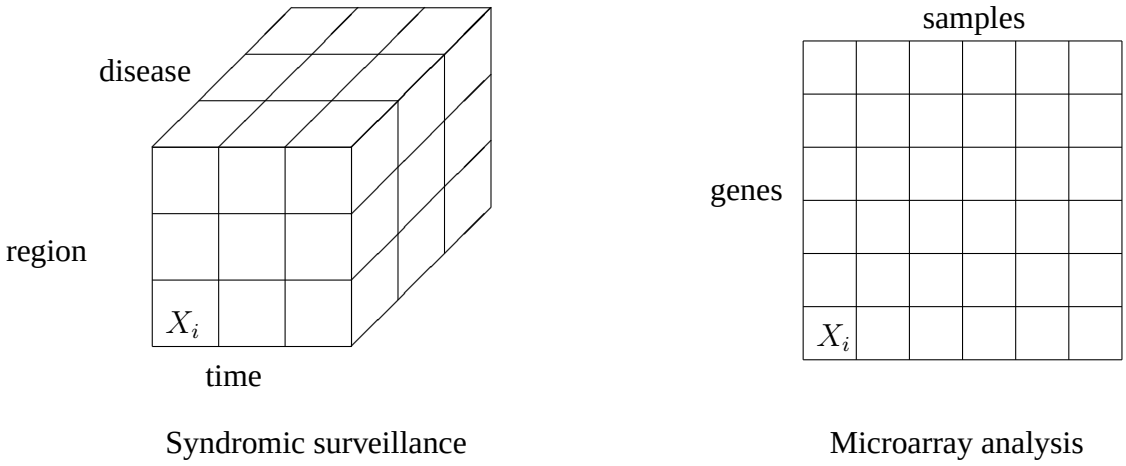


Figure 2: Tensor representation of categorical data

These submatrices represent positive or negative correlations between the measured genes and the various samples in the data. Shabalin applied many biclustering methods including LAS to real-world datasets and showed that LAS outperformed other algorithms at finding overlapping clusters. Section 3.1 shows the comparison of DAD to LAS on both synthetic and real-world datasets.

3 Disjunctive Anomaly Detection (DAD)

Consider a multivariate dataset $\mathcal{X}$ where each covariate takes one value along d categorical dimensions. The categorical dimension i takes values from a fixed set $\mathcal{S}^i$. Imagine arranging the data dimensions on an order-d tensor represented as $\mathcal{T}$. Figure 3 shows example data representations for syndromic surveillance and microarray analysis applications.

The entries of $\mathcal{T}$ represent a single corvariate and measure some domain specific quantity. In the case of microarray analysis, the first dimension represents the measured genes and the second dimension represents the samples each from different patients in the experiment. Each entry of the tensor (a matrix in this case) corresponds to the activation level of a specific gene in a particular sample. In the syndromic surveillance domain, each dimension of $\mathcal{T}$ represents a separate categorical dimension in the data. Based on the observed data, the goal of the Disjunctive Anomaly Detection algorithm is to find a set of anomalous clusters that are sub-tensors of $\mathcal{T}$. These clusters represent the strong associations between the samples and genes in microarray data or subsets of data likely affected by a disease outbreak in syndromic surveillance data.

We now present the DAD algorithm to solve the sample-variable association problem. Assume that there are K clusters, C_1 to C_K, in data. Cluster C_p is defined as a sub-tensor (a submatrix) in $\mathcal{T}$ (data matrix) with a corresponding intensity α_p. We assume each entry X_i in the data matrix is a random variable with the following distribution:

$$X_i \sim \mathrm{N}(\mu_i, 1) \tag{1}$$

such that

$$\mu_i = \sum_{p=1}^{K} I(X_i \in C_p)\alpha_p \tag{2}$$

As modeled here, every element X_i in the matrix follows Gaussian distribution with mean μ_i and unit variance [1]. The input data is pre-processed to closely satisfy the unit variance assumption. The mean, μ_i, is the sum of all cluster intensities containing the element X_i, yielding an additive model. Since we are only interested in positive associations, we assume $\alpha_p \geq 0$. Negative associations can be found by inverting the sign of all values in the observed data. Now, given only the observed data and the number of clusters K, the goal of DAD algorithm is to find the maximum likelihood estimate of the anomalous sub-tensors and corresponding intensities (α) for K clusters using the following objective function:

$$\begin{aligned} f &= \max_{\{C_1, C_2, \ldots, C_K\}} \log\left(\mathcal{L}(\mathcal{X})\right) \\ &= \max_{\{C_1, C_2, \ldots, C_K\}} \sum_{i=1}^{n} -\frac{1}{2}(x_i - \sum_{p=1}^{K} I(X_i \in C_p)\alpha_p)^2 + M \quad \text{s.t. } \alpha_p \geq 0 \quad \text{(Eqns. 1, 2)} \end{aligned} \tag{3}$$

where x_i is the observed value of variable X_i, n is the number of variables and M is a constant. Note that since we care only about clusters $C_1, \ldots, C_K$, we can ignore computing the exact value of C.

It is important to note that even if K is equal to 1, maximizing Eqn. 3 over all possible sub-tensors for C_1 in the data is a NP-hard problem ([10]). Hence, we propose an approximate algorithm to find a good set of candidates for the clusters $C_1, C_2, \ldots, C_K$. We first start by finding a good candidate for cluster C_1 in the data. We then remove C_1 from the data and find another good candidate for cluster C_2. Once cluster C_2 is computed, we then try to simultaneously improve both C_1 and C_2 in order to maximize the likelihood of the observed data. This alternate process of adding new clusters and optimizing currently known clusters is repeated until a good set of K clusters is generated. Finding a good candidate cluster at each step is NP-hard. Hence, we propose hill-climbing combined with random restarts to overcome the local maxima problem (details discussed below).

Formally, given an order-d tensor $\mathcal{T}$, DAD algorithm finds K clusters $C_1, C_2, \ldots, C_K$ that best explain the observed data using some parametric model for the entries in $\mathcal{T}$. The cluster definition C_k includes parameters $\{s_k^1, s_k^2, \ldots, s_k^d; \alpha_k\}$ where $s_k^i \subseteq \mathcal{S}^i$ represents the set of affected values along dimension i and α_k indicates the cluster intensity. Algorithm 3.1 forms the generic framework for the DAD algorithm. We will describe this in detail next. In Sect. 3.1, we present

[1] Unit variance is borrowed from [40]; later in Sect. 3.2 we relax this assumption.

the results of applying DAD to microarray analysis. Sect. 3.2 describes modifications used for syndromic surveillance application.

Algorithm 3.1: COMPUTEDADCLUSTERS($\mathcal{X}, d, K, R$)

for $k \leftarrow 1$ **to** K
do
- subtract contribution of $C_1, \ldots, C_{k-1}$ from $\mathcal{X}$
- $C_k \leftarrow$ FINDCLUSTER($\mathcal{X}, R, d$)
- add contribution of $C_1, \ldots, C_{k-1}$ to $\mathcal{X}$
- **repeat**
  - compute $\alpha_1, \ldots, \alpha_k$ using NNLS
  - $e \leftarrow$ SUBSETUPDATE($\mathcal{X}, k, C_1, \ldots, C_k$)
- **until** $e = 0$

return $(C_1, \ldots, C_K)$

Algorithm 3.1 outlines the pseudocode for computing DAD clusters. The input parameters are the observed data $\mathcal{X}$, the number of categorical dimensions d, the number of required clusters K, and the number of restart steps R. Starting with $k = 1$, we first find a reasonable cluster C_k (Alg. 3.2) by removing the contribution of clusters $C_1, \ldots, C_{k-1}$. We then try to improve the cluster locations for $C_1, C_2, \ldots, C_k$ using the subset update algorithm (Alg. 3.3), and the set of cluster intensities $\alpha_1, \ldots, \alpha_k$ using non-negative least squares (NNLS) optimization (described below). This alternating process continues until the data likelihood of $\mathcal{X}$ in the presence of $C_1, C_2, \ldots, C_k$ cannot be improved any further. The algorithm stops when K clusters have been generated.

Lets derive the NNLS procedure for the sample-variable association problem. By fixing the cluster locations s_j^i where $1 \leq i \leq d$ and $1 \leq j \leq k$, we need to find the values for $\alpha_1, \ldots, \alpha_k$ that maximize the data likelihood. Assuming that there are a total of k clusters in Eqn. 3 and the cluster locations are known in advance, we get

$$\begin{aligned}
f &= \max_{\{C_1, C_2, \ldots, C_k\}} \sum_{i=1}^{n} -\frac{1}{2}(x_i - \sum_{p=1}^{k} I(X_i \in C_p)\alpha_p)^2 + M \quad \text{s.t. } \alpha_p \geq 0 \\
&= \max_{\{\alpha_1, \alpha_2, \ldots, \alpha_k\}} \sum_{i=1}^{n} -\frac{1}{2}(x_i - \sum_{p=1}^{k} I(X_i \in C_p)\alpha_p)^2 + M \quad \text{s.t. } \alpha_p \geq 0 \quad (\text{all } s_j^i \text{ fixed}) \\
&= \min_{\{\alpha_1, \alpha_2, \ldots, \alpha_k\}} \sum_{i=1}^{n} \frac{1}{2}(x_i - \sum_{p=1}^{k} I(X_i \in C_p)\alpha_p)^2 - M \quad \text{s.t. } \alpha_p \geq 0 \\
&= \min_{\{\alpha_1, \alpha_2, \ldots, \alpha_k\}} \frac{1}{2}||B - A\varphi||_2^2 - M \quad \text{s.t. } \alpha_p \geq 0 \qquad (4)
\end{aligned}$$

where column vector $B_{n\times 1}$ contains observed values x_i, matrix $A_{n\times k}$ has element a_{ij} set to $I(X_i \in C_j)$ and column vector $\varphi_{k\times 1}$ has the unknown intensities $\alpha_1, \ldots, \alpha_k$. It is easy to see that the unknown intensities $\alpha_1, \ldots, \alpha_k$ in Eqn. 4 can be found by minimizing the equation

$$\min_{\{\alpha_1, \alpha_2, \ldots, \alpha_k\}} ||B - A\varphi||_2^2 \quad \text{s.t. } \alpha_p \geq 0 \qquad (5)$$

We solve Eqn. 5 using the NNLS solver described in [23]. Note that Eqn. 5 can be solved using any standard quadratic programming optimizer.

Algorithm 3.2: FINDCLUSTER($\mathcal{X}, R, d$)

$\mathcal{L}^* \leftarrow$ likelihood of $\mathcal{X}$ assuming zero clusters[a]
$C^* \leftarrow \varnothing$
for $r \leftarrow 1$ **to** R
do
- generate a random cluster C^r
- **repeat**
  - $\{e \leftarrow$ SUBSETUPDATE$(\mathcal{X}, 1, d, C^r)$
- **until** $e = 0$
- subtract contribution of C^r from $\mathcal{X}$
- $\mathcal{L} \leftarrow$ likelihood of $\mathcal{X}$ assuming zero clusters
- **if** $\mathcal{L} > \mathcal{L}^*$
  - **then** $\mathcal{L}^* \leftarrow \mathcal{L}$; $C^* \leftarrow C^r$
- add contribution of C^r to $\mathcal{X}$

return (C^*)

[a] By "zero clusters" we mean $K = 0$ in Eqn. 2; hence $\mu_i = 0$

Algorithm 3.2 computes the cluster C^* that maximizes the likelihood of $\mathcal{X}$ assuming there is only one cluster present in $\mathcal{X}$. As described earlier, finding the single optimal cluster C^* when $d > 1$ is a NP-hard problem (see [10]) similar to the standard biclustering problem. Hence, we resort to approximation by combining hill climbing with random restart to find a good candidate for C^*. We first randomly place a cluster (C^r) in the data. The cluster definition is then iteratively improved using hill climbing (Alg. 3.3) until convergence. To prevent the local maxima problem, we repeat the process R times and return the best cluster found using random restarts. Note that we compute the data likelihood of $\mathcal{X}$ assuming that there are no clusters in the data. This is because the effect of the previously known clusters has already been removed before calling Alg. 3.2. This is the same as computing the original data likelihood of $\mathcal{X}$ (input to Alg. 3.1) in the presence of clusters $C_1, \ldots, C_{k-1}, C^r$ as defined in the Eqn. 3.

Given the set of n input clusters, the goal of Alg. 3.3 is to change the selection of values along exactly one dimension of a single cluster such that the overall data likelihood of $\mathcal{X}$ is improved. The objective function f_S finds the best set of values in the cluster j and the dimension i that maximizes the data likelihood assuming all clusters except j and all dimensions in C_j except for i are fixed. We derive the appropriate function f_S in Appendix A for the sample-variable association problem. We find the best subset using f_S for all possible values of i and j keeping track of the best pair (i^*, j^*) that gives the largest gain in the data likelihood. Finally, the algorithm updates

the values along the i^* dimension in the j^* cluster.

Algorithm 3.3: SUBSETUPDATE($\mathcal{X}, n, d, C_1, \ldots, C_n$)

subtract $C_1, \ldots, C_n$ from $\mathcal{X}$
$f^* \leftarrow$ log-likelihood of $\mathcal{X}$ assuming zero clusters
$i^* \leftarrow -1$
$j^* \leftarrow -1$
$S^* \leftarrow \phi$
$\alpha^* \leftarrow 0$
for $i \leftarrow 1$ **to** d
do {
  add C_i to $\mathcal{X}$
  for $j \leftarrow 1$ **to** n
  do {
    Compute S_j^i, α using f_S assuming other dimensions constant
    if $f_S > f^*$
    then {
      $f^* \leftarrow f_S$
      $i^* \leftarrow i$
      $j^* \leftarrow j$
      $S^* \leftarrow S_j^i$
      $\alpha^* \leftarrow \alpha$
    }
  }
  subtract C_i from $\mathcal{X}$
}
add $C_1, \ldots, C_n$ to $\mathcal{X}$
if $i^* \neq -1$
then {
  comment: update cluster j^*'s dimension i^* values and intensity
  $S_{j^*}^{i^*} \leftarrow S^*$
  $\alpha_{i^*} \leftarrow \alpha^*$
  return (1)
}
else {**return** (0)

3.1 Application: Microarray Analysis

Large-Average-Submatrix (LAS) algorithm

LAS is the state-of-the-art algorithm designed to detect multiple clusters in microarray data. We use it to compare against our evaluations of DAD. LAS assumes that each data element follows the distribution mentioned in Eqn. 1. It uses the following scoring function (Eqn. 6) for every possible submatrix U in the data

$$S(U) = -\log\left[\binom{m}{k}\binom{n}{l}\Phi(-\tau\sqrt{kl})\right] \tag{6}$$

where (m, n) are the number of rows and columns in the data, (k, l) are the number of rows and columns in U, τ is the average of all the elements in U, and Φ is the CDF of the standard normal

distribution. Note that this function scores small clusters with large averages the same as large clusters with small averages. LAS first reports a submatrix U as cluster C_1 that maximizes the scoring function (Eqn. 6) in the data. Similar to DAD, LAS also uses random restarts and hillclimbing to find a good candidate U in the data. LAS then removes the cluster C_1 from the data and repeats the cluster search process to find C_2. LAS stops after either K clusters have been reported in the data or the score $S(U)$ a new cluster found becomes very small.

The key difference between LAS and DAD is that LAS does not optimize multiple clusters simultaneously whereas DAD does. Hence there is no NNLS optimization required by LAS. Even though LAS computations are simpler, DAD has a potential to find better clusters. DAD directly optimizes the data likelihood using the objective function f_S (see Appendix A) whereas LAS iteratively finds clusters using the function $S(U)$.

Experiments

To enable initial comparison, we generated synthetic data in the form of a matrix A of size ($n \times n$) containing entries randomly sampled from a standard normal distribution. We then added 5 clusters (of type submatrix) at random locations in A with α for each cluster uniformly drawn between α_{min} and α_{max}. The number of values along each dimension in the cluster was randomly selected between s_{min} and s_{max}. The matrix A was then used as an input to both the DAD and LAS algorithms in order to recover the locations and intensities of the injected clusters. Figure 3(a) shows the average performance (over 25 executions) of DAD and LAS as a function of the number of identified clusters K. The data generation parameters were set to $n = 25, \alpha_{min} = 1, \alpha_{max} = 2, s_{min} = 2,$ and $s_{max} = 10$. Similarly, the parameters for Fig. 3(b) and Fig. 3(c) were set to $n = 50, \alpha_{min} = 1, \alpha_{max} = 2, s_{min} = 5, s_{max} = 20$ and $n = 100, \alpha_{min} = 1, \alpha_{max} = 2, s_{min} = 10, s_{max} = 25$ respectively. The results in Fig. 3(a), 3(b), and 3(c) clearly show that DAD significantly outperforms LAS in explaining the data when $K \geq 4$. We also present the slice of results at $K = 5$ (true number of injected clusters) for different sizes of A in Fig. 3(d) indicating the gain in the data likelihood by using DAD instead of LAS.

Apart from using the likelihood, we also evaluated how well do the predicted clusters match with the true clusters using overlap score as a metric. An overlap score θ measures the goodness-of-fit of a set of predicted clusters $\mathcal{C}_p$ to the known set of true clusters $\mathcal{C}_g$. The overlap score is equal to the ratio of the sum overlap scores between best matching cluster pairs in $\mathcal{C}_p$ and $\mathcal{C}_g$ (chosen without replacement) and $\max\{|\mathcal{C}_p|, |\mathcal{C}_g|\}$. The overlap score between two clusters G and P is computed as:

$$\theta' = \frac{|G \cap P|}{|G \cup P|} \quad . \tag{7}$$

The intersection and union is defined over the set of elements in the tensor that belong to clusters G and P. Hence, $\theta' = 1$ indicates G and P match exactly and $\theta' = 0$ indicates no match. Figure 3(e) shows the average overlap scores between LAS clusters and DAD clusters when $K = 5$ over 25 independent runs. The results indicate that DAD outperforms LAS in finding the true cluster locations.

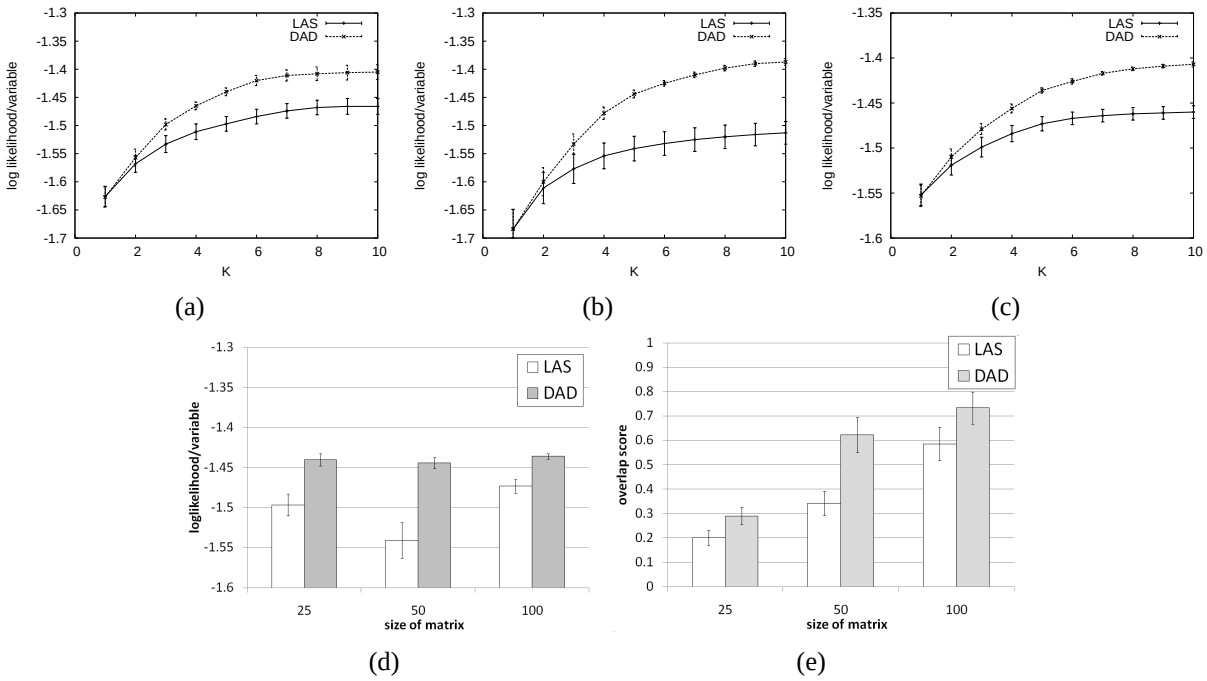
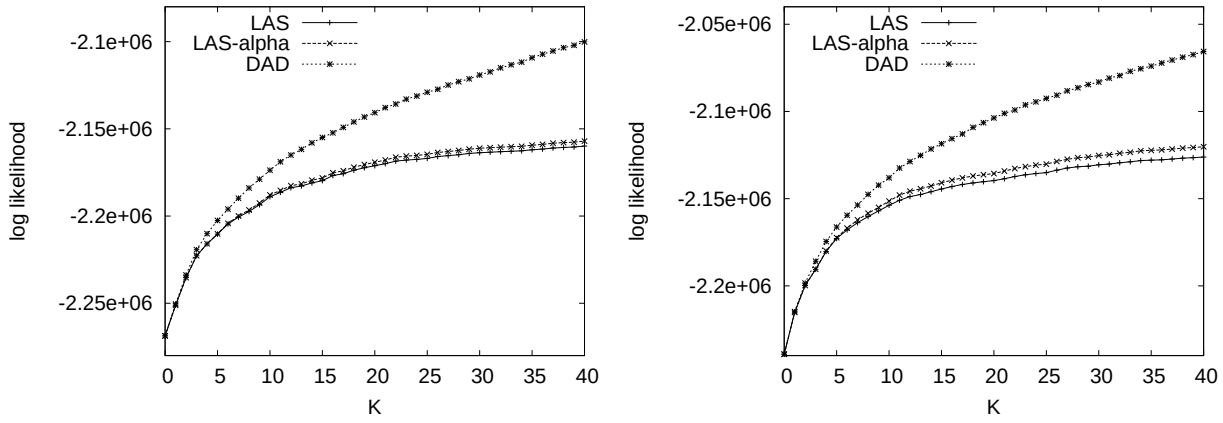


Figure 3: Performance comparison of DAD vs. LAS on synthetic data (a) $n = 25$, (b) $n = 50$, (c) $n = 100$, (d) Likelihood ($K = 5$) and (e) Overlap score ($K = 5$)

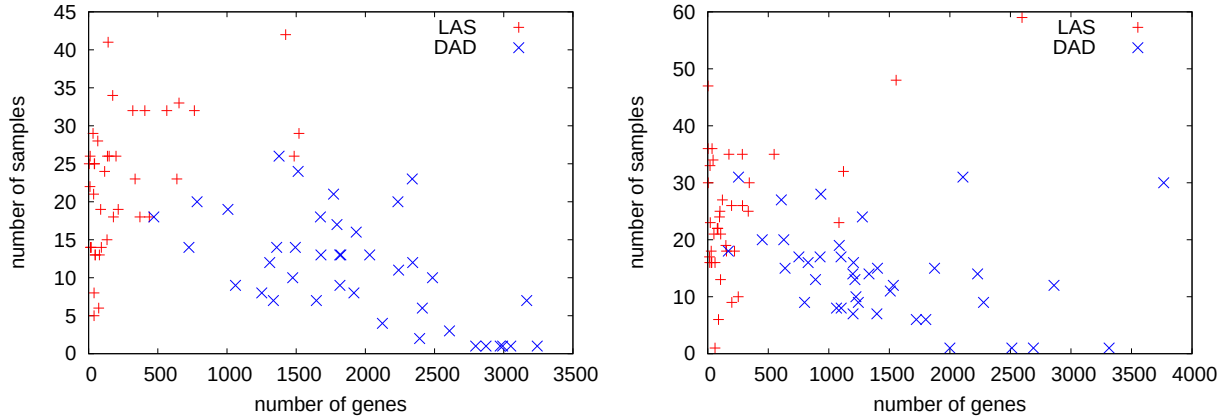
We also compared performance of DAD and LAS on two real-world microarray datasets: the breast cancer study from [21] and the lung cancer study from [6]. The rows of the microarray data correspond to the measured genes and columns to the tumor samples. The breast cancer data has $13,666$ rows and 117 columns and the size of the lung cancer data is $12,625 \times 125$. We pre-processed the data as described in the original paper on LAS algorithm [40], including the log transform to compensate for heavy tails in the observed data distributions and column normalizations. For breast cancer data, LAS algorithm with 1000 random restarts took a few hours to find 40 clusters whereas DAD took almost 6 days with 100 random restarts. We expect DAD to be slower than LAS because LAS does not require NNLS optimization and it does not redefine old clusters when finding new clusters. We plan to reduce DAD's execution time as a part of the proposed work.

Figure 4 shows the results of executing DAD and LAS on the breast and lung cancer data. LAS-alpha curve was obtained by keeping the cluster locations same as LAS at each K but optimizing intensity values using NNLS to maximize the likelihood score. We can clearly see that for $K \geq 3$, DAD clusters better explain the observations in the data as compared to results from LAS. Specifically, Fig. 4(a) shows that the data likelihood obtained using 10 DAD clusters matches to that obtained using 40 LAS clusters. Hence, DAD clusters are more informative than LAS clusters. Figure 4 also indicates that for $K \geq 25$ the likelihood improvement using LAS saturates (slope ~ 0) whereas DAD continues to improve the data likelihood. The likelihood curve using LAS-alpha is a marginal improvement over the curve using LAS. This suggests that optimizing both the cluster locations and intensities as done by DAD is useful.



(a) (b)

Figure 4: Likelihood curves using real data (a) Breast and (b) Lung



(a) (b)

Figure 5: Size of clusters using real data (a) Breast and (b) Lung

Figure 5 shows the scatter plot of various cluster sizes found by LAS and DAD. The plots indicate LAS clusters are more compact than obtained by DAD. Even though the difference seems significant, we do not know its biological implications yet. LAS objective function heavily penalizes clustes with large sizes where as DAD just optimizes for better likelihood.

Table 1 shows a few other metrics computed using LAS and DAD clusters when $K = 40$ that were found in the real data. Effective clusters is defined using the following equation

$$F(C_1, \ldots, C_K) = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{x \in C_k} \frac{1}{N(x)} \tag{8}$$

where $N(x) = \sum_{k=1}^{K} I(x \in C_k)$ is the number of clusters containing element x. The number of effective clusters, $F(\cdot)$, is upper bound by K. It can be thought as the effective number of

Table 1: Other comparison metrics for LAS and DAD

Metric	Breast		Lung	
	LAS	DAD	LAS	DAD
Effective clusters	33.54	**34.182**	29.52	**33.39**
Gene correlation	**0.186**	0.176	**0.212**	0.131
Sample correlation	0.225	**0.347**	0.258	**0.304**
Gene standard deviation	0.967	**0.774**	1.111	**0.930**
Sample standard deviation	1.001	**0.932**	1.203	**1.103**

non-overlapping clusters that can be formed using $C_1, \ldots, C_K$. Gene correlation of a cluster is the average pairwise correlation of the constituent genes (the higher the better). Gene standard deviation of a cluster is the average standard deviation of values of the constituent genes (the lower the better). Sample correlation and standard deviation are defined using respective samples in the cluster. Table 1 shows the average, across all 40 clusters, correlation and standard deviation scores.

The number of effective clusters using DAD is greater than using LAS. This combined with improved likelihood (see Fig. 4) suggests that DAD clusters are more informative. DAD outperforms LAS on all metrics (shown in bold) except Gene correlation. This can partially be explained using Fig. 5. DAD clusters contain 5 to 10 fold more genes than LAS. This results in low gene correlation scores using DAD. To conclude, even though we do not have the true clusters for breast and lung datasets, the results in Fig. 4, Fig. 5 and Tbl. 1 suggest that DAD outperforms LAS in finding the sample-variable associations for microarray analysis.

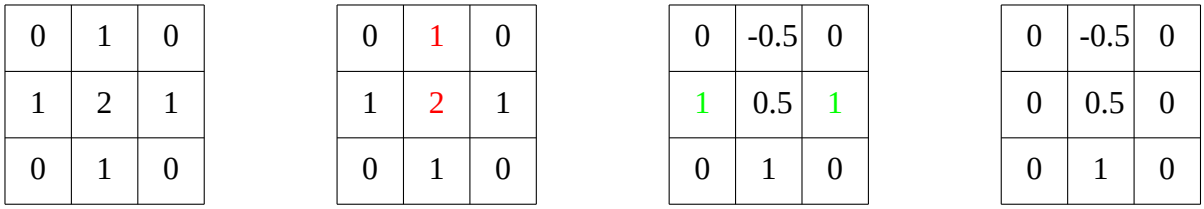
To illustrate the shortcomings of LAS, we present results of executing LAS and DAD on a small noise-free toy dataset, called *2-Cluster* data, in Fig. 6. The data contains two true clusters (all elements in the second row and all elements in the second column) each with $\alpha = 1$. LAS results are shown in Fig. 6(a). The first matrix shows the input data. The red elements in the second matrix represent the first cluster found by LAS ($\alpha_1 = 1.5$). The elements in the third matrix show the residue after the first LAS cluster is removed from the data. We also see the location of the second LAS cluster with green elements ($\alpha_2 = 1$). Finally, the fourth matrix shows the data residue after removing the first two clusters as reported by the LAS algorithm.

Figure 6(b) presents the clusters found using DAD algorithm. The second matrix shows the one best cluster (in red) with $\alpha_1 = 1.33$. The third matrix shows the 2-best clusters found using DAD (in red and green with the common element shown in yellow) with $\alpha_1 = 1, \alpha_2 = 1$. These two cluster definitions result in zero-residue matrix as shown in the fourth matrix. Hence, DAD outperforms LAS because it has the power to re-define old clusters while it generates the new ones.

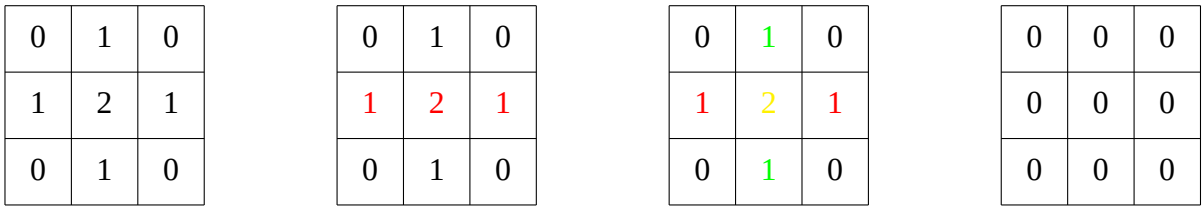
3.2 Application: Syndromic Surveillance

Theory

Consider d categorical dimensions in a syndromic surveillance dataset including one temporal dimension. We will now describe how to run DAD in a prospective surveillance mode in which the algorithm is executed each day to find emerging anomalous clusters. Let T be the day of analysis.



(a)



(b)

Figure 6: DAD vs. LAS on 2-Cluster data (a) LAS results (b) DAD results

At time T build an order-d tensor $\mathcal{T}$ from the data where all the non-temporal dimensions take all possible values in the data and the temporal dimension takes W values: $T-W+1, \ldots, T-1, T$. W is the assumed maximum duration of an outbreak. It is a user-provided parameter. For all experiments discussed here we have set $W = 7$ days. Now each element X_i in the tensor $\mathcal{T}$ is assumed to have the following distribution:

$$X_i \sim \mathrm{N}(\mu_i, \sigma_i^2) \tag{9}$$

$$\mu_i = b_i(1 + \sum_{p=1}^{K} I(X_i \in C_p)\alpha_p) \quad \text{s.t. } \alpha_p \geq 0 \tag{10}$$

where b_i, σ_i are the mean and variance of X_i respectively. The parameters (b_i, σ_i) are estimated using the baseline data prior to the time period $T-W+1$. The cluster intensity (α_p) represents the increased disease rate and acts as a multiplicative factor over the baseline mean. The multiplicative approach taken here differs from the additive model used for the sample-variable association problem described in Sect. 3.1.

Given the data observed up to time T, the goal of DAD is to recover the clusters $C_1, \ldots, C_K$ with disease rates $\alpha_1, \ldots, \alpha_K$ such that most anomalies in the observed data are accounted for successfully. The objective function f_S to be used in Alg. 3.3 is derived in Appendix B. We now describe the NNLS optimization for the multiplicative model. Given the cluster locations, the goal of NNLS is to find the set $\alpha_1, \ldots, \alpha_K$ that maximizes the data likelihood. Similar to Eqn. 4, we can use the log-likelihood of the multiplicative model to find intensities $\alpha_1, \ldots, \alpha_K$ as follows:

$$
\begin{aligned}
f &= \max_{\{\alpha_1,\alpha_2,\ldots,\alpha_k\}} \sum_i -\frac{1}{2\sigma_i^2}(x_i - b_i - \sum_{p=1}^{k} I(x_i \in C_p) b_i \alpha_p)^2 + M \quad \text{s.t. } \alpha_p \geq 0 \\
&= \min_{\{\alpha_1,\alpha_2,\ldots,\alpha_k\}} \sum_i \frac{1}{2}(\frac{x_i - b_i}{\sigma_i} - \sum_{p=1}^{k} I(x_i \in C_p) \frac{b_i \alpha_p}{\sigma_i})^2 - M \quad \text{s.t. } \alpha_p \geq 0 \\
&= \min_{\{\alpha_1,\alpha_2,\ldots,\alpha_k\}} \frac{1}{2}||B - A\varphi||_2^2 - M \quad \text{s.t. } \alpha_p \geq 0 \qquad (11)
\end{aligned}
$$

where x_i is the observed value for variable X_i, column vector $B_{n\times 1}$ contains values $\frac{x_i - b_i}{\sigma_i}$, matrix $A_{n\times k}$ has element a_{ij} set to $\frac{b_i}{\sigma_i} I(X_i \in C_j)$ and column vector $\varphi_{k\times 1}$ has the unknown intensities $\alpha_1, \ldots, \alpha_k$. Similar to solving Eqn. 5, we find optimal values for $\alpha_1, \ldots, \alpha_k$ using method described in [23].

We can now apply DAD to find a set of K clusters in the data on the day of analysis T. Note that usually K is not known in advance. Alternatively, we propose an AIC-based penalization technique to hedge against overfitting with the number of reported clusters K. The motivation for Eqn. 12 is similar to selecting the number of clusters using AIC regularization in Gaussian mixture models (GMM). As we increase the number of clusters in the model, we score the clusters $C_1, \ldots, C_k$ using the AIC criterion as follows:

$$AIC_k = 2\rho k M - 2l_x \qquad (12)$$

where M is the number of free parameters in each cluster, k is the number of reported clusters so far, l_x is the data log-likelihood, ρ is the regularization tuning parameter. M is equal to the sum of the number of distinct values along each dimension in the data plus one (for α). Adding clusters in the data increases both the data likelihood and also the AIC regularization penalty. Hence, for a given ρ, minimizing Eqn. 12 with respect to k provides a way to choose the right set of clusters (K); hence hopefully avoiding overfitting during the cluster estimation process. Note that ρ is not a part of standard regularization procedure used for example in GMM. We introduce it as a tuning parameter to compute ROC and AMOC curves for evaluating DAD versus WSARE.

Experiments

To demonstrate the utility of DAD in syndromic surveillance applications, we used the disease data from Sri Lanka [46]. It consists of weekly counts of cases of 9 diseases aggregated in 26 regions of the country from December 26, 2007 to July 10, 2009. The raw data was artificially converted to a daily resolution by sampling from a fixed day-of-week distribution. This makes the data similar to typical health surveillance application scenarios where data feeds are available daily. The goal of such surveillance is to monitor incoming data for statistically significant unexpected increases in disease activity in one or more regions of the country, which could not be explained by just random fluctuations in the recently observed data.

In order to compare DAD and WSARE, we injected synthetic disease outbreaks into the data. At each simulation trial, we generated a set of K synthetic disease clusters all ending at a fixed

date T. Each cluster affected at most V_{max} values across each dimension of the data: disease and region. Each synthetic cluster lasted a random duration w (between 1 and $w_{max} = 7$) such that all days in the range $T - w + 1, \ldots, T - 1, T$ were affected by the outbreak. The counts of the data falling into a synthetic cluster were increased by a fixed percentage that was randomly choosen between 100% (corresponding to $\alpha = 1$) and 200% (corresponding to $\alpha = 2$). Hence, this process injected a set of K potentially overlapping anomalous clusters each having a constant disease rate.

WSARE is designed to identify the most significant cluster in the data on day T. We made WSARE produce multiple clusters by iteratively removing the effect of each detected cluster from the data, and running WSARE again. Also, since WSARE does not natively characterize the duration of outbreaks, we allowed it to search for clusters using all fixed time windows from 1 to w_{max} days and then report the detection with the highest score.

For each set of injected clusters ending at T, we run DAD for each day starting on $T - w_{max} + 1$ and ending on T. During each day of analysis, we build an order-3 tensor with region, disease and time dimensions (using data from last 7 days). The baseline mean and variance parameters were estimated using the counts observed during the prior 12 weeks in the historic data. On each day, we compare clusters found by DAD (predicted clusters) against the known ground truth, measuring the detection rate and timeliness of detection. It is worth noting that the raw data contains real unlabeled disease outbreaks which - if detected - would be considered false positives in our evaluation. To characterize the accuracy and timeliness of outbreak detection, we produced ROC and AMOC plots for each of the evaluated algorithms. These plots were generated by changing the regularization parameter ρ for DAD and p-value threshold α (Fisher's score) for WSARE. Changing these parameters controlled the number of clusters reported by each algorithm on a specific day of analysis. The maximum number of reported clusters per day was upper-bounded by 10. Each set of predicted clusters was compared with the set of true clusters. Each pair of best matched clusters between predicted set and true set was assumed detected if $\theta' \geq \theta^t$ where θ^t was a fixed overlap threshold. θ' is previously defined using Eqn. 7. Note that a higher value of θ^t indicates higher accuracy requirement for successfully matching between a predicted cluster and true cluster.

Figure 7 shows the performance of DAD and WSARE on the Sri Lanka data. All plots show average performance over 100 independently injected cluster sets. Figures 7(a) and 7(b) show the detection accuracy and the timeliness performance respectively for different values of the overlap score. The injection parameters were set to $K = 1$ and $V_{max} = 1$. The curve labeled "w0.05" corresponds to the WSARE results with overlap score $\theta^t = 0.05$. Similarly, the curve labeled "d0.15" refers to the DAD's performance when $\theta^t = 0.15$. We observe that there is a significant drop in detection accuracy for WSARE as the overlap score increases but DAD's performance does not suffer. This indicates that DAD cluster definitions better match the ground truth than WSARE cluster definitions. DAD outperforms WSARE on both accuracy and timeliness at all settings of overlap score threshold. To be specific, at the allowance of 2 alerts per day DAD reveals 40% better accuracy and detects emerging outbreaks nearly 1.5 days earlier, when compared to WSARE.

Figures 7(c) and 7(d) show the performance comparison by varying the maximum number of affected values V_{max} along each data dimension. The other injection parameters were set to $K = 1$ and $\theta^t = 0.10$. We see that for all values of V_{max} (measure of disjunctiveness) DAD performs better than WSARE. Figures 7(e) and 7(f) compare the performance of DAD and WSARE with respect

to the number of simultaneous injected clusters (varying K). The injection parameters for these experiments were $V_{max} = 1$ and $\theta^t = 0.10$. As we increase the number of clusters, both algorithms improve accuracy but DAD performs far better than WSARE. DAD is indeed well suited to detect and disambiguate sets of complex overlapping disease clusters. Even though $V_{max} = 1$ means that the cluster definitions fall in the WSARE search space, DAD gets less confused by the noise as it can quickly remove noise from the data and find the relevant disease outbreak signal, characterizing it with a compact and comprehensible conjunctive-disjunctive statement. Hence, Figure 7 clearly shows that DAD beats WSARE across all settings: explanation power (θ^t), size of impact (V_{max}) and the detection of multiple simultaneous events (K).

Figure 8 illustrates a hypothetical scenario when WSARE fails but DAD does detect the right set of clusters. We refer to this data as *Zero-Margin* data. The data contains 2 categorical dimensions (rows and columns of the matrix) each with 4 categorical values. Assume that the baseline counts in the data are all set to 10 (not shown in the figure). Figure 8 shows the recent observed counts with the location of 3 anomalous clusters in red, blue and green. It is clear from the figure that all the single component WSARE rules (any row or column marginal) have a Fisher's Exact score of 1.0 (least significant) as the 2×2 table for Fisher's Exact test has first row entries equal to 40 and the second row entries equal to 120. Hence, WSARE does not report any clusters. It is important to note that even though the clusters labeled in red and blue are in the search space of WSARE, it fails to detect them. DAD, on the other hand, can easily find all the anomalous clusters correctly. In summary, over-aggregation during marginalization can mislead WSARE.

4 Time Series Cube (T-Cube)

Time series data is abundant in many domains including finance, weather forecasting, epidemiology, and many others. Many such datasets can be interpreted as the time series of interval (e.g. daily) counts of events (such as the number of certain types of drugs, e.g. anti-diarrheals sold; number of patients reporting to the emergency department with specific symptoms, etc). These time series can be sliced-and-diced across multiple categorical dimensions such as location, gender and age group of patients, and so on. The computational efficiency of the mining operations which one may want to apply to such a data, as well as the efficiency of accessing the interesting information in a manual drill-downs, heavily depends on the efficiency of extracting the series of counts aggregated for specific values across these categorical dimensions. This section introduces a data structure designed to dramatically decrease the time of retrieval of such aggregates for any complex ad-hoc query across multiple categorical variables. The data structure achieves its efficiency by pre-computing and caching the responses to all possible queries against the underlying temporal database of counts annotated with sets of symbolic labels, while keeping the memory requirements in check.

We are aware of at least two good examples of practical settings in which ad-hoc time series querying capability is needed. Firstly, take a database used by the public health officials for the syndromic surveillance. These officials perform disease outbreak monitoring on a daily basis. It often involves investigation of the alerts of possible problems flagged by the automated outbreak detection systems. For such investigations, the health officials need to execute large numbers of

complex ad-hoc queries in order to extract the data needed for interpretation of the alerts and/or for isolating their likely causes, while differentiating the real outbreaks from the false alarms. Such investigations need to be executed in a timely fashion, and long waits for data extracts are not acceptable. Secondly, automated statistical analyses or data mining procedures executed against such databases require access to a large number of different projections of data, if they are to comprehensively screen for possible indications of the public health problems. Long waits for the corresponding extracts may (and often do) render such systems impractical as both of the example usage cases heavily rely on quick responses to the complex queries. The users would dramatically benefit if the cached responses to all the possible queries was available in advance.

Standard approach to handling ad-hoc queries in the commercial databases is to use an On-Line Analytical Processing (OLAP) system. The idea relies on the data cubes: cached data structures extracted from (usually only parts of) the original data and constructed in a form allowing for fast ad-hoc querying of pre-selected subsets of the aggregated data [19]. For the sake of brevity we do not review the details of the OLAP technology here, but these methods are known to suffer from long build times and large memory requirements (causing the need to rely on high-end database servers). Additionally, as we have observed empirically, data cubes still typically require in the order of one second or longer to respond to a complex query on the datasets which we tested. Such latency is a substantial inconvenience to the users who want to execute multiple queries in an online fashion. It also hampers statistical analyses which may require responses for millions of complex queries. That could take days of processing time using industry-standard OLAP data cubes. There are different types of implementations of the data cubes based on how counts are stored internally (refer [19] for more details). Irrespective of the implementation, the goal of the data cubes is to respond to simple and complex queries against large databases as fast as possible. With growing demand for data mining and statistical analyses against large databases, innovation work has been focusing on improving data cube performance ([18], [20], [33]).

Data cubes are closely related to another technology which originated from the computer science research: Cached Sufficient Statistics. Similar to the data cubes, cached statistics structures pre-compute answers to queries, but they are designed cover all possible future queries (not just pre-selected subsets), and they aim at the efficiency of not only the data retrieval, but also their (most often in-memory) representations. AD-Trees are very good examples of such data structures. AD-Tree [27] (All-Dimensional Tree) is designed to efficiently represent counts of all possible co-occurrences among multiple attributes of the categorical data. This is very important in many scenarios involving statistical modeling of such data, where most operations require computing aggregate counts, ratios of counts or their products. Quick access to the counts of arbitrary subsets of demographic properties is essential for the overall performance of analytic tools which rely on them. AD-Trees have been shown to dramatically speed-up notoriously expensive machine learning algorithms including Bayesian Network learning [27], Empirical Bayesian Screening, Decision Tree learning and Association Rule learning [3]. The attainable speedups range from one to four orders of magnitude with respect to the previously known efficient implementations. These efficiencies are attainable at moderate memory requirements, which are easy to control. Moore describes in [27] the details of the structure, its construction algorithm as well as fundamental characteristics of the AD-Trees. There are also dynamic implementations of the AD-Trees [22]

which help grow the structure on demand and which can be more memory efficient than fundamental implementations. AD-Trees are the best of the existing solutions to the categorical data representation when it comes to very quickly responding to the ad-hoc queries against the large datasets.

4.1 Data Structure

T-Cube (Time series Cube) is an in-memory data structure designed for very fast retrieval and analysis of additive data streams such as the time series of counts. It is a derivative of the AD-trees idea. We show how the AD-Trees can be easily and efficiently extended to the domain of time series analysis. T-Cube consists of two main components: **D-Cache** and **AD-Tree**. Note that because AD-Tree is designed to work with categorical data, the T-Cube is only applicable to the temporal datasets with categorical dimensions.

Consider a dataset with d dimensions ($d-1$ dimensions are categorical and 1 temporal dimension). Now lets organize the categorical dimensions on a $d-1$ dimensional tensor and label each element in the tensor as a covariate. For every covariate with support in the data, the D-Cache stores its corresponding time series using the temporal dimension. Table 2 shows the corresponding structure of a D-Cache built for a clothing retail dataset whose dimensions are the color of T-shirt, size of T-shirt, and date of sale (data spans over T days). Note that it is common to have lots of covariates with no data and hence these are not stored in the D-Cache. The result of any time series query that is restricted to a sub-tensor of the $d-1$ dimensional tensor can be reported as the aggregated time series over the list of covariates that match the query in the D-Cache. Hence, using the D-Cache, the query response time is linear in the number of covariates in the data. This can be slow if the number of covariates in the D-cache is large (say in the order of thousands or millions).

Table 2: D-Cache for clothing retail dataset

	d_1	d_2	...	d_T
(red, small)	10	15	...	20
(red, large)	5	1	...	7
...	...	...	...	...
(blue, medium)	7	5	...	9
(green, small)	5	4	...	10

The goal of using the AD-Tree structure is to reduce the query response time from linear to logarithmic in the number of covariates in the D-Cache. Figure 9 depicts the fully developed AD-Tree representation of the clothing retail dataset. The data nodes with the time series are shown as circles with shaded bars and the vary nodes over the categorical dimensions are depicted with rectangles. Traditionally, each data node of the AD-Tree [27] contains a single count of occurrences corresponding to the particular covariate. In the T-Cube representation, the data node consists of a time series. This time series is the result of summation of corresponding counts across all the covariates in the D-Cache that match the current AD-Tree data node. These matching covariates

are called the leaf-list for the corresponding data node in the tree. As we go from the top to the bottom of the tree, the sizes of the leaf-lists decrease. In a fully developed tree, the leaf nodes have leaf-lists of size one pointing to a single covarite in the D-Cache. Therefore the AD-Tree combined with the D-Cache stores response to all possible time series queries that are either single covariate or any marginal query of the $d-1$ dimensional tensor. Note that answering more general as well as more complex time series queries (that correspond to type sub-tensors) can be accomplished very quickly by navigating the AD-Tree and performing simple arithmetic operations on the vectors of counts pointed to by the traversed data nodes. The attainable efficiencies are analogous to those reported in the fundamental Ad-Tree paper [27].

It is easy to see that the AD-Tree memory requirements are exponential in the number of categorical dimensions and the number of values per dimension. There are two solutions to manage high memory requirements: the tree depth and the most common value trick (refer [27], [38] for more details). Both solutions trade slower query response time for lower memory requirements of the AD-Tree. We briefly sketch these solutions here. For further details including extensive experiment results, refer to [38].

The root node of the AD-Tree represents the aggregate time series that matches all covariates in the D-Cache, i.e. the leaf list of the root node is as large as the number of covariates in the D-Cache. A complete tree keeps growing until the leaf nodes represent exactly one covariate, at which point such nodes directly point to the individual D-Cache covariate. The data node leaves in the fully grown AD-Tree will have a leaf list size of 1. To reduce the memory requirement, we can set a lower bound on the leaf list size of all the data nodes in the tree (α). A data node is further expanded only if its leaf list size is greater than α. Larger values of α will correspond to smaller trees and hence lower memory usage. Note that with $\alpha > 1$, the AD-Tree leaves will point to multiple covariates in the D-Cache. Hence, queries that invoke more specific time series will have to sequentially scan the D-Cache rows which can be done in linear time.

It is possible to further reduce the memory requirements by exploiting redundancies in the AD-Tree structure. Moore and Lee [27] have shown that it is possible to remove one vary node together with the corresponding sub-tree from under each dimension node, and still be able to recover all the counts which were represented in the complete tree. Typically, the greatest possible savings in terms of the number of nodes in the tree can be attained if the sub-trees removed correspond to the most common value (MCV) (i.e. the one with the largest sub-tree) of the dimension under consideration. Applying MCV to reduce the memory requirement is not always beneficial. For instance, if the MCV trick is applied to a vary node with $10K$ values each having an equally long leaf list, then we will have to add $\sim 10K$ time series at run time to respond to a query containing the MCV, which can be computationally expensive. In order to mitigate such effects, we introduce a parameter called MCV fraction (γ) that is the ratio of the leaf list size of the MCV node and the leaf size of its immediate parent vary node. The tree growing algorithm will prune a MCV node only if the current node MCV fraction is higher than the MCV fraction threshold. This parameter helps to balance the memory savings against the average query response time. For $\gamma \geq 1.0$, the MCV trick will never be applied and for $\gamma = 0.0$, the MCV trick will always be applied. The MCV trick really helps in the case of skewed dimensions where one value occurs more often than others (e.g. binary dimensions with high sparsity).

4.2 Evaluation

We tested T-Cube on three different datasets: two real-world and one synthetic. All the datasets used in the experiments consisted of records collected over one year period, with counts aggregated daily. The datasets vary in the number of records (volume), the number of categorical dimensions and the number of values along each dimension (arity). Even though the first two datasets are related to the domain of bio-surveillance, their size and characteristics are similar to data that can be found in other domains: a few dimensions of high arity (zip codes, business names, etc.) and many dimensions of low arity (gender, age group, company sizes, etc.).

The emergency room chief complaint (ED) dataset contains hospital emergency room patient visit records from 4 US states (PA, NJ, OH and UT). Each record consists of the following dimensions: visit date, syndrome, patient home zip code, age group, gender. The patient home zip code has $21K$ distinct values. The syndrome dimension represents the chief complaint reported by the patient and has 8 distinct values (Respiratory, Gastrointesitnal, Constitutional, etc.). Age group dimension takes one of the three values (Adult, Child and Unknown). Similarly, the dimension gender could also take three values (Male, Female, and Unknown). The dataset had approximately 3.4 million records over the one year period. There were a total of $120,604$ covariates stored in the D-Cache for the ED data.

The over-the-counter (OTC) data contains the volumes of daily medication sales collected at more than $10K$ pharmacies throughout the US. The data has been geographically aggregated to the level of a zip code in order to preserve the privacy of the individual store operations. Each record contains the following dimensions: purchase date, store zip code, medicine category, and sale promotion. This data covers $8,119$ distinct zip codes. The category dimension represents the class of medicine (e.g. cough/cold remedies, baby/child electrolytes, etc.) and took one of the 23 distinct values. Binary information about the occurrence of store promotions on medications is provided in the dimension promotion as 'Y' or 'N'. The T-Cube stored $356,000$ covariates in its D-Cache representation.

The ED and OTC datasets had only a few categorical dimensions. In order to evaluate the utility of the T-Cube in a more general setting, we have created a sparse binary synthetic data (SYN). It had 32 dimensions including date, count, zip code, and 29 binary dimensions. Each of the 29 binary dimensions were 95% sparse, i.e. the records took the value of zero 95% of the time. Each record has the zip code randomly assigned from the pool of $10K$ values. The date spanned a one year time period. The count dimension took values randomly selected from the range of 5 to 10. The SYN dataset contained 12 million records resulting in approximately 4.5 million covariates in the D-Cache.

Table 3 shows the T-Cube build time, memory utilization and the query response time for different α and γ input parameters across the three datasets. For brevity, we have shown only a few combinations of α and γ here (for further details refer [38]). The D-cache build times vary between 2 to 15 mins. Most of the time in building the D-Cache is spend in reading the data from the disk. The D-Cache build time is independent of the input parameters α and γ as these only change the AD-Tree size. The AD-Tree build times are negligible (a few seconds) as compared to the D-Cache build times. Even then the T-Cube build time is much smaller than the usually reported numbers for the data cubes (sometimes lasting hours).

The memory requirement of the D-Cache is much smaller than that of the AD-Tree. As we increase α from 1 to 100, the memory requirement for the ED and OTC datasets reduce dramatically (676 to 50 MB in case of ED and 1116 to 30 MB for OTC). This makes the T-Cube very practical as it does not require big servers with large memories. This shows that the T-Cube can be easily loaded on a standard laptop with a GB of RAM to analyze the data efficiently. In case of the SYN dataset that had 29 sparse binary dimensions, setting $\gamma = 1.0$ (i.e. no MCV trick) required more than 16GB of RAM. As soon as we set $\gamma = 0.8$, the memory needed reduced to 100 MB. All three datasets with reasonable settings of α and γ needed less than 100 MB of RAM.

The T-Cube response to simple queries that were pre-cached in the T-Cube (no aggregates needed) was within a few milliseconds (Table 3 shows average over 1000 queries). Even at $\alpha = 100$, the response was sub-millisecond for ED and OTC data. The slowest response time for SYN data was 50 milliseconds as queries with the MCV value in one of the dimensions resulted in rebuilding the AD-Tree nodes. We also randomly generated complex queries where each dimension could take upto β proportion of the values in that dimension. The T-Cube needed to aggregate various covariate time series to respond to such complex queries. Figure 10 shows the average response time over 1000 queries under different values of β. $\beta = 0.25$ means each dimension in the query could take upto 25% of the values. Even for the highly complex queries against the ED and OTC, the T-Cube required only a few milliseconds to respond.

Table 3: T-Cube evaluation

	α	γ	Build time (secs)		Memory (MB)		Response time (millisecs)
			D-Cache	AD-Tree	D-Cache	AD-Tree	
ED	1	1.0	124	16	36	676	0.1
	100	1.0		6		50	0.4
OTC	1	1.0	944	26	77	1116	0.01
	100	1.0		15		30	0.05
SYN	1	1.0	913	-	808	> 16000	-
	1	0.8		25		100	50

We also compared performance of the T-Cube against other data cube tools available commercially. Due to the privacy concerns, we do not list the names of these tools here, but most of these tools are commonly used in many practical OLAP applications involving time series data. The data used for these tests had three demographic dimensions with arities of 1000, 10, and 5 respectively. The data contained 12 million records of daily transactions over a period of one year. The experiments were performed on a system with 2.4GHz CPU and 2GB memory, running Windows XP operating system.

Table 4 shows the T-Cube performance as compared to the other tools. The response time is averaged over 1000 queries with $\beta = 0.10$. Each of the commercial tools required a different amount of memory to represent the test data, however for all tools, the response time improved with the increase in the amount of memory consumed. At most optimal settings, the commercial tools require seconds to respond to a complex query. The T-Cube on the other hand is able to respond in milliseconds, i.e. 1000 times faster than the commercial tools. The two versions of the T-Cubes (row 4 & 5) differ in the value of α. The first one uses $\alpha = 1000$ and the second uses

$\alpha = 10$ in order to illustrate the trade-off between the memory consumption and the response time attainable with the T-Cubes.

Table 4: Performance comparison: T-Cube vs. commercial tools

Query engine	Type	Memory (MB)	Response time (seconds)
Tool 1	RDMS	330	6.8
Tool 2	In memory	231	7.6
Tool 3	In memory	> 1000	3.5
T-Cube	In memory	236	0.022
T-Cube	In memory	845	0.005

4.3 Impact

- **T-Cube Web Interface (TCWI)**

  The Auton Lab team has developed a (publicly available) web-based interface that uses the T-Cube data structure at its core in order to mine temporal datasets [42]. TCWI allows users to upload and analyze a dataset with multiple categorical dimensions and one temporal dimension. Users can slice-and-dice the data across various dimensions and generate multiple time series with the intend to find interesting patterns in data. The interface responds to user queries in realtime. Also, there are various data mining tools that can perform automatic scans of millions of queries and report the findings within seconds. Figure 11 shows the snapshot of the interface.

- **Finding microbial outbreaks in Food Supply data**

  U.S. Department of Agriculture (USDA) routinely collects data from various food producing establishments across the country testing for microbial contaminations (such as Salmonella infections). We have analyzed these datasets using the TCWI to look for patterns of microbial clusters involving E.Coli, Listeria, etc. The users at USDA have found TCWI very helpful in quickly finding interesting spatio-temporal patterns that were not known before.

- **Linking Human Illness data to Food Sources**

  Center for Disease Control and Prevention (CDC) collects reports of Salmonella infections in humans. Using the TCWI, we were able to show evidence that records in the human illness data could potentially be linked to the records in the food supply data from the USDA inspections in the establishments. Finding such linkages through comprehensive searches can lead to faster recalls, better inspection policies and safer environment for food consumers ([8], [13]).

- **Realtime Biosurveillance Project: A pilot program**

  We have successfully deployed TCWI in a pilot program in India and Sri Lanka. The TCWI is used to monitor daily feeds of patient visits, looking for emerging disease outbreaks ([36], [43], [44], [45]).

- **U.S. Air Force maintainence and supply data**

  The TCWI has also been used by the U.S. Air Force to analyze maintainence records of fleets of aircraft. Using TCWI, the users look for emergence of systematic patterns of breakdown so that preventative measures can be taken quickly in order to minimize the downtime of planes.

In summary, we have successfully applied TCWI to find complex anomalous patterns in a variety of transactional data across spatio-temporal and categorical dimensions. We have received favorable feedback from all our TCWI users. The realtime response to ad-hoc user queries helps clients quickly browse the data for investigation purposes and also to execute statistical data mining algorithms. The TCWI and the T-Cube has resulted in multiple publications: [8], [12], [13], [14], [15], [16], [26], [32], [34], [35], [36], [37], [38], [39], [43], [44], [45].

5 Proposed Work

Section 3 shows the results of applying DAD algorithm to microarray analysis and syndromic surveillance. We now present our proposed work towards the thesis, grouped into three categories: new directions, statistical improvements, and empirical evaluation plans.

5.1 New Directions

- **Tracking clusters over time**

  We envision that DAD algorithm will be executed at regular intervals (e.g. on a daily basis) against temporal datasets. The users would like to re-execute the cluster detection process when new data becomes available. As currently designed, our algorithm can be executed independently of the clusters reported in the past.

  However, it may be advantageous to consider clusters identified in the past when determining current model. For instance, we could use clusters reported yesterday to seed DAD's optimization for today's data. It may then converge in fewer iterations, than if executed without any prior information, if the world does not change drastically from day to day. We propose to investigate introduction of smoothing constraints on DAD cluster definitions so that the daily models could evolve smoothly over time, while still accurately reflecting the current data. We will compare the algorithm performance in both temporally dependent and independent modes of operation.

  Our interactions with the potential users of DAD suggest that they would like to track evolution of clusters over time in order to maintain high levels of situation awareness. We will research how to enable DAD to label currently identified clusters as for instance newly emerging, previosly known and escalating, previously known and diminishing, merging, splitting, etc. We envision using metrics derived from the overlap score and cluster intensity to perform such classification of evolving clusters.

- **Constrained cluster search**

  Currently DAD cluster search is fairly unconstrained. The only restriction is that the clusters must be in the form of sub-tensors defined along the categorical dimensions in the data. We now present a few scenarios where further constraining the search space makes practical sense.

  Imagine a dataset with a large number of categorical dimensions, say $20 - 100$. Anomalous clusters defined over all the dimensions may be difficult to comprehend for users. Hence, constraints on the number of anomalous dimensions reported by DAD clusters is one possible constraint.

  It is sometimes important to constrain the number of affected values along each dimension. For example, in the case of microarray analysis, clusters containing thousands of genes provides little or no insight as to which subset of genes should be further analyzed for indications of biological markers. The spatial scan algorithm tries to enforce graphical constraints along spatial dimensions in the data so that the anomalous clusters are defined over geographically contiguous regions. We plan to investigate how to modify DAD algorithm to respect such useful constraints when analyzing real-world datasets.

- **Scale-up DAD for large datasets**

  DAD's execution time is directly proportional to the number of data dimensions. We have shown the performance of DAD on the Sri Lanka data that contains 3 dimensions (Location, Disease and Time) with a few tens of values along each dimension. It takes 2-3 seconds to find the anomalous clusters in this dataset. As currently stated, DAD algorithm cannot process large datasets with tens of dimensions and thousands of values along each dimension in a reasonable time. Some example datasets that we would like to run DAD on include: Air Force and the U.S. Department of Agriculture datasets.

  We plan to scale-up DAD in order to handle more realistic datasets. We will explore ways to prune the search space, possibly using heuristics, by trading off detection accuracy for execution time.

5.2 Statistical Improvements

- **DAD with non-constant intensity models**

  Currently DAD assumes that clusters have constant intensities. In the domain of syndromic surveillance, this assumption is often violated at least in the temporal dimension. Typically, the disease rate has a unimodal temporal pattern: starts with a positive slope, reaches a peak and then slowly phases out with a negative slope. Disease outbreaks often have uneven disease rate distributions over spatial regions.

  We will propose models that fit non-constant intensity clusters. We will first start by relaxing the constant intensity assumption in the temporal dimension for the purpose of syndromic surveillance.

- **Computing significance score of clusters**

  In the work done so far we assume that either the number of clusters K is given in advance or is found using AIC regularization (Eqn. 12). We have yet to find support for such discrete AIC regularization in the literature. Balancing between overfitting the noise (more clusters) and underfitting the data (fewer clusters) is very important in every domain where DAD will be applied.

  There are two approaches that can avoid reporting clusters generated by chance. In the first approach, we can use randomization using Monte-Carlo simulations: sample data from the baseline distribution (assuming zero clusters) and compare anomalous clusters in the sampled data to the ones in the real data. This will help us compute significance of the clusters. Since each execution of DAD is computationally expensive, computing significance using randomization may be practically infeasible on realistically sized datasets. Alternately, we can borrow ideas from the structured multiple hypothesis testing literature ([1], [4], [11]) that use Bonferroni correction to compute significance scores. It is a challenge to derive closed form solutions to compute significance scores of hypotheses that can take overlapping sub-tensor structures in multidimensional spaces.

- **DAD under Poisson distribution**

  Most of the scoring functions derived in this proposal assume a Gaussian distribution for the model variables. The parameters of the distribution are either computed using historic data (syndromic surveillance) or are given as input to the model (microarray analysis). We will like to extend DAD to Poisson distribution. In the syndromic surveillance literature, both the Gaussian and Poisson models are popular. To formalize, in Sect. 3 DAD finds overlapping clusters using NNLS optimization assuming $X_i \sim N(\mu_i, \sigma_i^2)$. We plan to derive appropriate scoring functions for $X_i \sim Pois(\lambda_i)$.

5.3 Empirical Evaluation Plans

- **DAD versus MBSS**

  Section 3.2 compared DAD and WSARE on the syndromic surveillance data. DAD outperforms WSARE both in the detection rate and the time-to-detection. Similarly, we will evaluate DAD against one other popular biosurveillance algorithm: MBSS [30]. We believe that DAD will have better explanatory power as compared to MBSS in the presence of spatially non-colocated and overlapped disease outbreaks.

- **DAD user feedback**

  We work with clients from the U.S. Department of Agriculture and the Air Force to analyze their multidimensional categorical datasets. We plan to execute DAD algorithm on these datasets and get user feedback on the utility of anomalous clusters found by the DAD algorithm in their domains, to help us guide further development of the algorithm.

- **Biological significance of gene-sample biclusters**

In the context of sample-variable association, DAD clusters appear to differ from LAS clusters. LAS inventors at the University of North Carolina closely work with a group of biologists. We plan to send the DAD cluster definitions to them to get qualitative feedback. It might turn out that DAD clusters, along with better likelihood, also provide complementary to LAS insight into the biological processes.

5.4 Thesis Timeline

Table 5 lists the milestones and expected effort allocation until thesis defense.

Table 5: Thesis timeline

Description	**2010**						**2011**				
	J	**A**	**S**	**O**	**N**	**D**	**J**	**F**	**M**	**A**	**M**
Proposal											
Track clusters over time											
Constrained cluster search											
Scaling-up DAD											
Evaluate DAD versus MBSS											
Integrate DAD into TCWI											
Statistical improvements											
DAD user feedback											
Thesis writing											
Job search											
Defense & Graduation											

6 Conclusion

Detection of complex anomalous patterns is a challenging task faced by analysts in many application domains. In this work, we presented a new algorithm, Disjunctive Anomaly Detection (DAD), designed for that task. DAD is unique in its ability to accurately identify multiple overlapping events affecting multiple dimensions of categorical data. We evaluated DAD in the context of two application domains: sample-variable association for microarray analysis in cancer research, and disease outbreak detection in public health surveillance.

We compared DAD with LAS (Large Average Submatrices [40]) in an attempt to find overlapping biclusters in the microarray data. DAD outperformed LAS on both synthetic and real-world datasets. DAD could fit the data more accurately using substantially less complex models than LAS.

We also compared DAD with WSARE (What's Strange About Recent Events [47]) in the context of the disease outbreak detection problem. DAD significantly outperformed WSARE both in terms of detection accuracy and timeliness when exposed to synthetically injected disease outbreaks. DAD clusters matched true cluster definitions more precisely than WSARE. In most cases

the detection accuracy of DAD was 20% to 30% higher than WSARE. DAD reliably detected emerging disease outbreaks 1 to 2 days earlier than WSARE.

This work also presents the first known algorithm evaluation framework for finding multiple overlapping disease outbreaks. We hope that our framework will lay the foundation to develop new algorithms that are suited for modelling and detecting overlapped anomalous clusters.

We also presented a very efficient data structure, T-Cube, to represent multi-dimensional categorical time series data. Experimental results show that T-Cube outperforms query response time of other commercially available tools for time series retrieval by 1-2 orders of magnitude. With the help of T-Cube, we have analyzed millions of time series in a matter of minutes looking for unusual patterns. We have successfully applied T-Cube to diverse transactional data sets (public health monitoring, food supply data, Air Force maintenance records).

In summary, we believe that the DAD algorithm should be useful in many application domains where there is a need to find complex anomalies in multidimensional categorical data.

APPENDIX

A Additive model

Consider a set of n random variables: $X_1, \ldots, X_n$. Assume a subset $S^\star$ of these variables have $\mathrm{N}(\alpha^\star, \sigma^2)$ distribution ($\alpha^\star \geq 0$) and others have $\mathrm{N}(0, \sigma^2)$ distribution. Given $\{x_1, \ldots, x_n, \sigma\}$ where x_i is a sample from X_i, we need to recover the best estimates of $S^\star$ and $\alpha^\star$.

We can compute $S^\star$ using maximum likelihood estimate (MLE) as follows:

$$S^\star = \arg\max_S f_S \tag{A-1}$$

where f_S is the likelihood of data assuming the variables in subset S have mean α (s. t. $\alpha \geq 0$) and all other variables have mean zero. Now for a particular subset S, we can find the value of α that maximizes f_S.

$$\begin{aligned}
f_S &= \max_{\alpha \geq 0} \sum_{i \in S} \left(-\frac{(x_i - \alpha)^2}{2\sigma^2} - \log(\sqrt{2\pi\sigma^2})\right) + \sum_{i \notin S} \left(-\frac{x_i^2}{2\sigma^2} - \log(\sqrt{2\pi\sigma^2})\right) \\
&= \max_{\alpha \geq 0} \sum_{i \in S} \left(-\frac{(x_i - \alpha)^2}{2\sigma^2}\right) - \sum_{i \in S} \left(-\frac{x_i^2}{2\sigma^2}\right) + \sum_{i \in S} \left(-\frac{x_i^2}{2\sigma^2}\right) + \sum_{i \notin S} \left(-\frac{x_i^2}{2\sigma^2}\right) - n \log(\sqrt{2\pi\sigma^2}) \\
&= \max_{\alpha \geq 0} \sum_{i \in S} \left(-\frac{(x_i - \alpha)^2}{2\sigma^2} + \frac{x_i^2}{2\sigma^2}\right) + \sum_{i=1}^{n} \left(-\frac{x_i^2}{2\sigma^2}\right) - n \log(\sqrt{2\pi\sigma^2}) \\
&= \max_{\alpha \geq 0} \sum_{i \in S} \frac{1}{2\sigma^2}(-(x_i - \alpha)^2 + x_i^2) + M \\
&= \max_{\alpha \geq 0} \sum_{i \in S} \frac{1}{2\sigma^2}(2\alpha x_i - \alpha^2) + M
\end{aligned} \tag{A-2}$$

where M is a constant. Taking derivative of Eqn. A-2 with respect to α, we get $\alpha = \frac{x_S}{|S|}$ (where $x_S = \sum_{i \in S} x_i$) if $x_S > 0$ and $\alpha = 0$ if $x_s \leq 0$. Substituting α back into Eqn. A-2 gives

$$f_S = \sum_{i \in S} \frac{1}{2\sigma^2}\left(2\frac{x_s}{|S|}x_i - \left(\frac{x_s}{|S|}\right)^2\right) + M = \frac{x_s^2}{2\sigma^2|S|} + M \tag{A-3}$$

when $x_S > 0$ and $f_S = M$ when $x_S \leq 0$. Note that solving Eqn. A-1 for all possible subsets S is generally infeasible as the number of subsets S is exponential in n. Fortunately since f_S uses the sum x_S as the sufficient statistics, we can rearrange all x_i's in the descending order of their magnitude and then assign the first K (where K varies from 1 to n) elements that maximize (A-3) in $S^\star$ (see [28], [29] for details of proof). Hence, we can find the best subset $S^\star$ in time linear in n. Note the corresponding MLE for $\alpha^\star$ will then be $x_{S^\star}/|S^\star|$.

B Multiplicative model

Consider a set of n random variables: $X_1, \ldots, X_n$. Assume a subset $S^\star$ of these variables have $\mathrm{N}(b_i(1+\alpha^\star), \sigma_i^2)$ distribution ($\alpha^\star \geq 0$) and others have $\mathrm{N}(b_i, \sigma_i^2)$ distribution.

Given $\{x_1, \ldots, x_n; b_1, \ldots, b_n; \sigma_1, \ldots, \sigma_n\}$ where x_i is a sample from X_i, we need to recover $S^\star$ and $\alpha^\star$.

Applying the technique from Appendix A, we can compute the scoring function f_S and MLE α for a given set S as

$$f_S = \frac{(\sum_{i \in S} y_i z_i)^2}{\sum_{i \in S} z_i^2} + M \tag{B-1}$$

$$\alpha = \frac{\sum_{i \in S} y_i z_i}{\sum_{i \in S} z_i^2} \tag{B-2}$$

where M is a constant, $y_i = \frac{x_i - b_i}{\sigma_i}$, $z_i = \frac{b_i}{\sigma_i}$ and $\sum_{i \in S} y_i z_i > 0$. When $\sum_{i \in S} y_i z_i \leq 0$ we get $f_S = 0$ and $\alpha = 0$. Also, similar to Appendix A, we can find the optimal set S^* in linear time by reordering the elements x_i's in decreasing order of x_i/b_i (see [28], [29] for details of the proof).

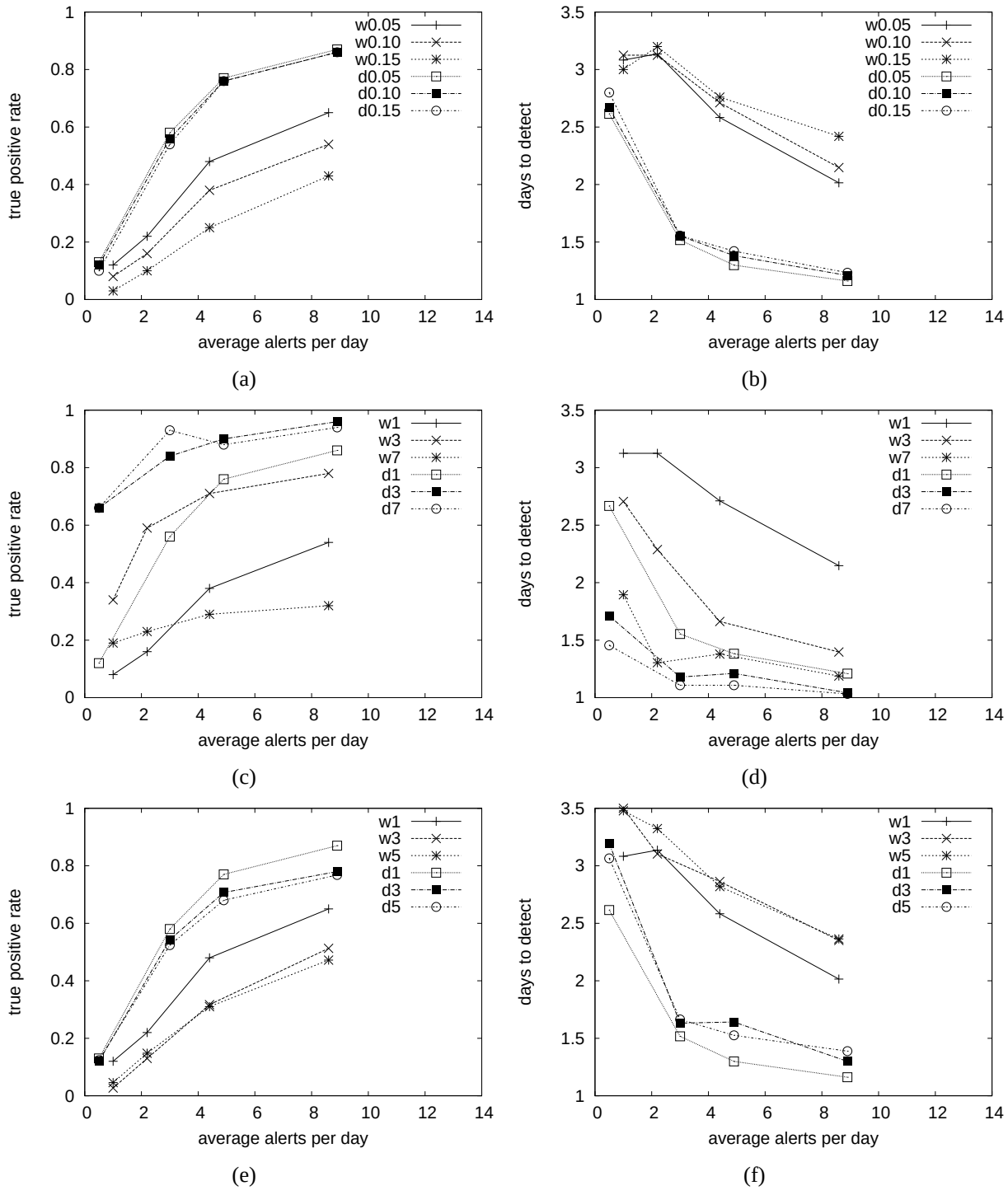


Figure 7: DAD vs. WSARE (a) Overlap ROC, (b) Overlap AMOC, (c) Cluster values ROC, (d) Cluster values AMOC, (e) Num clusters ROC, and (f) Num clusters AMOC

25	5	5	5
5	15	15	5
5	15	15	5
5	5	5	25

Figure 8: Zero-margin data

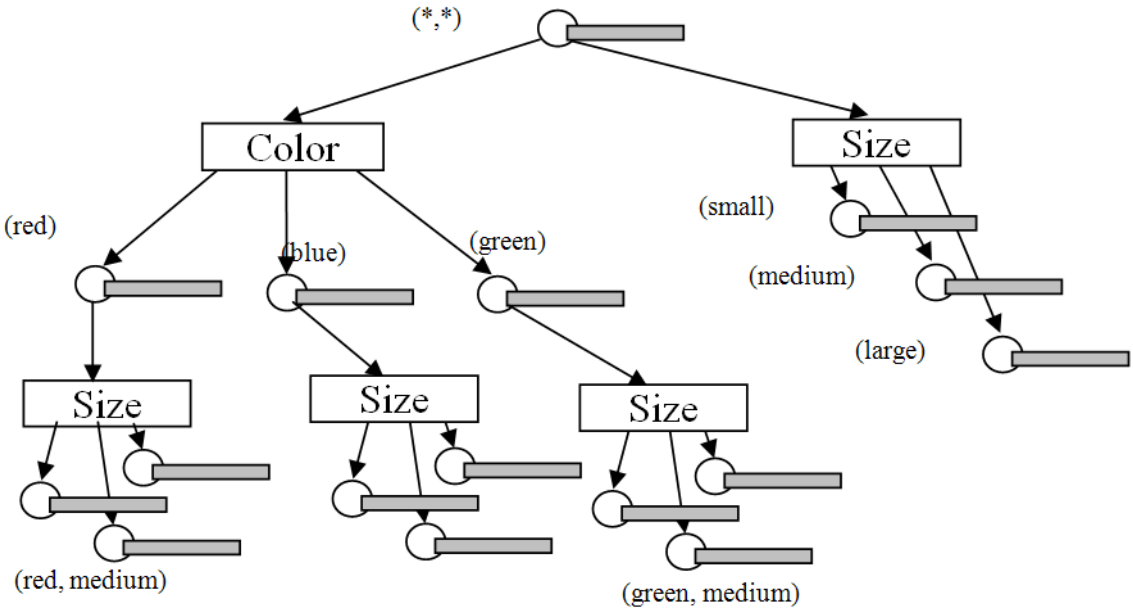


Figure 9: AD-tree for clothing retail dataset

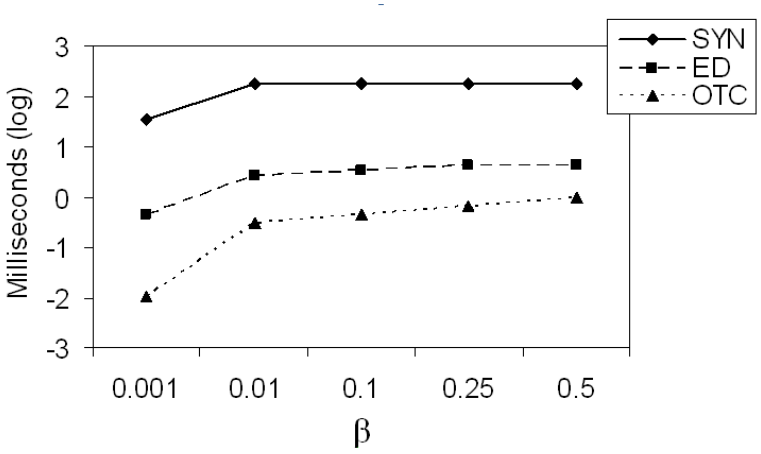


Figure 10: T-Cube response time on complex queries

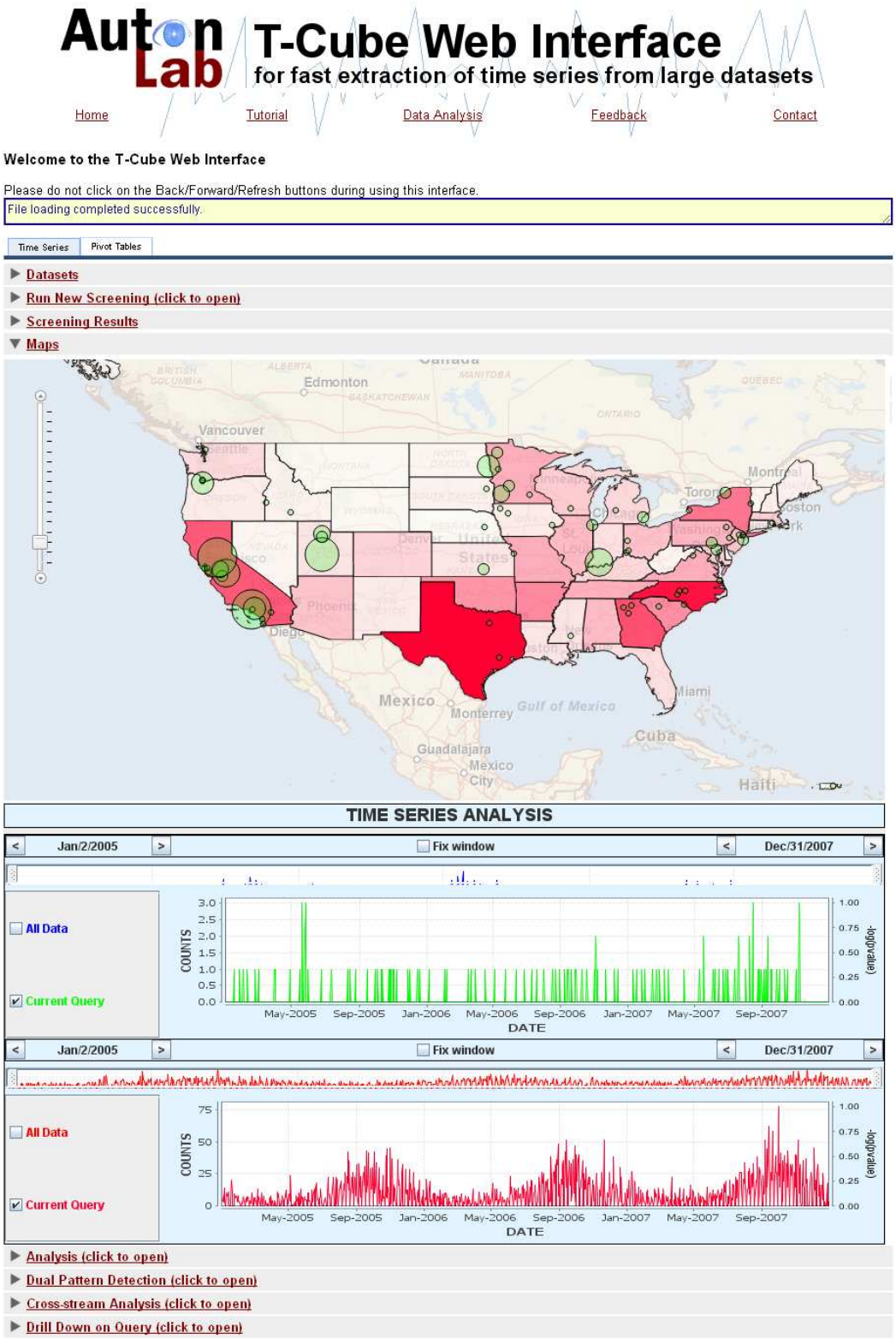


Figure 11: T-Cube Web Interface

Bibliography

[1] Louigi Addario-Berry, Nicolas Broutin, Luc Devroye, and Gabor Lugosi. On combinatorial testing problems. *The Annals of Applied Statistics*, Aug 2009.

[2] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. *SIGMOD Rec.*, 28(2):61–72, 1999.

[3] Brigham Anderson and Andrew W. Moore. Ad-trees for fast counting and for fast learning of association rules. In *Knowledge Discovery from Databases Conference*, 1998.

[4] Ery Arias-Castro, Emmanuel J. Candes, and Arnaud Durand. Detection of an anomalous cluster in a network. *The Annals of Applied Statistics*, Jan 2010.

[5] Amir Ben-Dor, Benny Chor, Richard Karp, and Zohar Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. *Journal of Computational Biology*, 10(3-4):373–384, June 2003.

[6] Arindam Bhattacharjee, William G. Richards, Jane Staunton, Cheng Li, Stefano Monti, Priya Vasa, Christine Ladd, Javad Beheshti, Raphael Bueno, Michael Gillette, Massimo Loda, Griffin Weber, Eugene J. Mark, Eric S. Lander, Wing Wong, Bruce E. Johnson, Todd R. Golub, David J. Sugarbaker, and Matthew Meyerson. Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the Natational Academy of Sciences, USA*, 98(24):13790–13795, 2001.

[7] Philip K. Chan and Matthew V. Mahoney. Modeling multiple time series for anomaly detection. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 90–97, Washington, DC, USA, 2005. IEEE Computer Society.

[8] Lujie Chen, Artur Dubrawski, Michael Baysek, Linda Kelley, Adrienne Dunham, Mike Huchabee, Paula J. Fedorka-Cray, C. Jackson, and B. McGlinchey. Detecting linkages between human illness and salmonella isolates in food using a new tool for spatio-temporal analysis of multi-stream data (poster extended abstract). In *AMIA Annual symposium*, 2008.

[9] Haibin Cheng, Pang-Ning Tan, Christopher Potter, and Steven A. Klooster. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the SIAM International Conference on Data Mining*, pages 413–424, 2009.

[10] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[11] David Donoho and Jiashun Jin. Higher criticism for detecting sparse heterogeneous mixtures. *The Annals of Statistics*, 32(3):962–994, 2004.

[12] Artur Dubrawski, Michael Baysek, Shannon Mikus, Charles McDaniel, Bradley Mowry, Laurel Moyer, John Ostlund, Norman Sondheimer, and Timothy Stewart. Applying outbreak detection algorithms to prognostics. In *AAAI Fall Symposium on Artificial Intelligence in Prognostics*, November 2007.

[13] Artur Dubrawski, Lujie Chen, Maheshkumar Sabhnani, Paula J. Fedorka-Cray, Lynda Kelley, Peter Gerner-Smidt, Ian Williams, Mark Huckabee, and Adrienne Dunham. Discovering possible linkages between food-borne illness and the food supply using an interactive analysis tool. In *Advances in Disease Surveillance*, 2009.

[14] Artur Dubrawski, Maheshkumar Sabhnani, Michael Knight, Michael Baysek, Daniel Neill, Saswati Ray, Anna Michalska, and Nuwan Waidyanatha. T-Cube Web Interface in support of real-time bio-surveillance program (extended demo abstract). In *The 3rd IEEE/ACM International Conference on Information and Communication Technologies and Development (ICTD)*, April 2009.

[15] Artur Dubrawski, Maheshkumar Sabhnani, Saswati Ray, Michael Baysek, Lujie Chen, John Ostlund, and Michael Knight. Interactive manipulation, visualization and analysis of large sets of multidimensional time series in health informatics. In *Proceedings of the 3rd INFORMS Workshop on Data Mining and Health Informatics*, November 2008.

[16] Artur Dubrawski, Maheshkumar Sabhnani, Saswati Ray, Joseph Roure, and Michael Baysek. T-Cube as an enabling technology in surveillance applications. In *Advances in Disease Surveillance*, volume 4, page 6, 2007.

[17] Jerome H. Friedman and Jacqueline J. Meulman. Clustering objects on subsets of attributes (with discussion). *Journal Of The Royal Statistical Society Series B*, 66(4):815–849, 2004.

[18] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, March 1997.

[19] Jiawei Han and Mitcheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2 edition, March 2006.

[20] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing data cubes efficiently. In *Proceedings of ACM SIGMOD*, pages 205–216, Montreal, Canada, June 1996.

[21] Zhiyuan Hu, Cheng Fan, Daniel Oh, JS Marron, Xiaping He, Bahjat Qaqish, Chad Livasy, Lisa Carey, Evangeline Reynolds, Lynn Dressler, Andrew Nobel, Joel Parker, Matthew Ewend, Lynda Sawyer, Junyuan Wu, Yudong Liu, Rita Nanda, Maria Tretiakova, Alejandra Orrico, Donna Dreher, Juan Palazzo, Laurent Perreard, Edward Nelson, Mary Mone, Heidi Hansen, Michael Mullins, John Quackenbush, Matthew Ellis, Olufunmilayo Olopade, Philip Bernard, and Charles Perou. The molecular portraits of breast tumors are conserved across microarray platforms. *BMC Genomics*, 7(1):96, 2006.

[22] Paul Komarek and Andrew W. Moore. A dynamic adaptation of AD-trees for efficient machine learning on large data sets. In *Proceedings of the 17th International Conference on Machine Learning*, pages 495–502, 2000.

[23] Charles L. Lawson and Richard J. Hanson. *Solving least squares problems*. Prentice-Hall, Englewood Cliffs, 3 edition, 1974.

[24] Xiaolei Li and Jiawei Han. Mining approximate top-k subspace anomalies in multi-dimensional time-series data. In *Proceedings of International Conference on Very Large Data Bases (VLDBf07)*, 2007.

[25] Jinze Liu, Jiong Yang, and Wei Wang. Biclustering in gene expression data by tendency. In *CSB '04: Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference*, pages 182–193, Washington, DC, USA, 2004. IEEE Computer Society.

[26] Shannon Mikus, Artur Dubrawski, Norman Sondheimer, Laurel Moyer, Michael Baysek, John Ostlund, Timothy Stewart, and Bradley Mowry. Collective machine learning for early identification of logistics crises. In *Integrated Health Management Conference*, August 2007.

[27] Andrew W. Moore and Mary Soon Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, March 1998.

[28] Daniel B. Neill. Fast and flexible outbreak detection by linear-time subset scanning. In *Advances in Disease Surveillance*, volume 5, page 48, 2008.

[29] Daniel B. Neill. Fast subset scan for spatial pattern detection. Technical report, Carnegie Mellon University, 2010.

[30] Daniel B. Neill and Gregory F. Cooper. A multivariate Bayesian Scan Statistic for early event detection and characterization. *Machine Learning Journal*, Nov 2009.

[31] Amela Prelić, Stefan Bleuler, Philip Zimmermann, Anja Wille, Peter Bühlmann, Wilhelm Gruissem, Lars Hennig, Lothar Thiele, and Eckart Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[32] Saswati Ray, Anna Michalska, Maheshkumar Sabhnani, Artur Dubawski, Michael Baysek, Lujie Chen, and John Ostlund. T-Cube Web Interface: A tool for immediate visualization, interactive manipulation and analysis of large sets of multivariate time series (poster extended abstract). In *AMIA Annual symposium*, 2008.

[33] Nick Roussopoulos, Yannis Kotidis, and Mema Roussopoulos. Cubetree: Organization of and bulk incremental updates on the data cube. In *Proceedings of the 1997 ACM SIGMOD Conference*, pages 89–99, 1997.

[34] Maheshkumar Sabhnani, Artur Dubrawski, and Jeff Schneider. Multivariate time series analyses using primitive univariate algorithms. In *Advances in Disease Surveillance*, volume 4, page 112, 2007.

[35] Maheshkumar Sabhnani, Artur Dubrawski, and Jeff Schneider. Detection of disjunctive anomalous patterns in multidimensional data. In *Advances in Disease Surveillance*, 2009.

[36] Maheshkumar Sabhnani, Artur Dubrawski, and Jeff Schneider. T-Cube Web Interface for real-time biosurveillance in sri lanka. In *Advances in Disease Surveillance*, 2009.

[37] Maheshkumar Sabhnani, Andrew W. Moore, and Artur Dubrawski. Rapid processing of ad-hoc queries against large sets of time series. In *Advances in Disease Surveillance*, volume 2, page 66, 2007.

[38] Maheshkumar Sabhnani, Andrew W. Moore, and Artur Dubrawski. T-Cube: A data structure for fast extraction of time series from large datasets. Technical Report CMU-ML-07-114, Carnegie Mellon University, Machine Learning Department, School of Computer Science, 2007.

[39] Maheshkumar Sabhnani, Daniel B. Neill, Andrew W. Moore, Artur Dubrawski, and Weng keen Wong. Efficient analytics for effective monitoring of biomedical security. In *Proceedings of the IEEE International Conference on Information and Automation*, December 2005.

[40] Andrey A. Shabalin, Victor J. Weigman, Charles M. Perou, and Andrew B. Nobel. Finding large average submatrices in high dimensional data. *The Annals of Statistics*, 3(3):985–1012, 2009.

[41] Sajid Siddiqi, Byron Boots, and Geoff Gordon. A constraint generation approach to learning stable linear dynamical systems. In *Advances in Neural Information Processing Systems*, December 2007.

[42] Public version of T-Cube Web Interface. https://tcube.autonlab.org/public/TCubeApp.ht

[43] Nuwan Waidyanatha, M. Ganesan, P. Weerakon, Gordan Gow, Artur Dubrawski, and Maheshkumar Sabhnani. Real-time biosurveillance pilot in India and Sri Lanka. In *4th Annual eASIA Conference*, December 2009.

[44] Nuwan Waidyanatha, S. Prashant, M. Ganesan, Artur Dubrawski, Lujie Chen, Michael Baysek, M. Careem, P. Damendra, and M. Kaluarachchi. Real-time biosurveillance pilot in India and Sri Lanka. In *IEEE Healthcom*, July 2010.

[45] Nuwan Waidyanatha, C. Weerasinghe, Artur Dubrawski, Maheshkumar Sabhnani, Michael Baysek, Saswati Ray, Steve Brudenell, Lujie Chen, M. Ganesan, P. Vincy, S. Prashant, and K. Janakiraman. T-Cube Web Interface as a tool for detecting disease outbreaks in real-time: A pilot in India and Sri Lanka. In *IEEE International Conference on Research, Innovation and Vision for the Future on Computing and Communication Technologies (submitted)*, 2010.

[46] Weekly Sri Lanka Epidemiological data. http://www.epid.gov.lk/wer.htm.

[47] Weng-Keen Wong, Andrew W. Moore, Gregory F. Cooper, and Michael M. Wagner. What's Strange About Recent Events (WSARE): An algorithm for the early detection of disease outbreaks. *Journal of Machine Learning Research*, 6:1961–1998, 2005.