

On Futuristic Query Processing in Data Streams

Charu C. Aggarwal

IBM T. J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
charu@us.ibm.com

Abstract. Recent advances in hardware technology have resulted in the ability to collect and process large amounts of data. In many cases, the collection of the data is a continuous process over time. Such continuous collections of data are referred to as *data streams*. One of the interesting problems in data stream mining is that of *predictive query processing*. This is useful for a variety of data mining applications which require us to estimate the future behavior of the data stream. In this paper, we will discuss the problem from the point of view of *predictive summarization*. In predictive summarization, we would like to store statistical characteristics of the data stream which are useful for estimation of queries representing the behavior of the stream in the future. The example utilized for this paper is the case of selectivity estimation of range queries. For this purpose, we propose a technique which utilizes a local predictive approach in conjunction with a careful choice of storing and summarizing particular statistical characteristics of the data. We use this summarization technique to estimate the future selectivity of range queries, though the results can be utilized to estimate a variety of futuristic queries. We test the results on a variety of data sets and illustrate the effectiveness of the approach.

1 Introduction

A number of technological innovations in recent years have facilitated the automated storage of data. For example, a simple activity such as the use of credit cards or accessing a web page creates data records in an automated way. Such dynamically growing data sets are referred to as data streams. The fast nature of data streams results in several constraints in their applicability to data mining tasks. For example, it means that they cannot be re-examined in the course of their computation. Therefore, all algorithms need to be executed in only one pass of the data. Furthermore, if the data stream evolves, it is important to construct a model which can be rapidly updated during the course of the computation. The second requirement is more restrictive, since it needs us to design the data stream mining algorithms while taking temporal evolution into account. This means that standard data mining algorithms on static data sets cannot be easily modified to create a one-pass analogue for data streams. A number of data mining algorithms for classical problems such as clustering and classification have been proposed in the context of data streams in recent years [1–8, 14].

An important problem in data stream computation is that of query selectivity estimation. Such queries include, but are not limited to problems such as selectivity estimation of range queries. Some examples of such queries are as follows:

- Find the number of data points lying in the range cube \mathcal{R} . (Range Query)
- For a target point \bar{X} , find the number of data points within a given radius r . (Radius Query)

A more general formulation of the above queries is to find the number of data points which satisfy a user-specified set of constraints \mathcal{U} . While this includes all standard selectivity estimation queries, it also allows for a more general model in which the selectivity of arbitrary constraints can be determined. For example, the constraint \mathcal{U} could include arbitrary and non-linear constraints using some combinations of the attributes. This model for selectivity estimation is significantly more general than one which supports particular kinds of queries such as range queries.

Consider an aggregation query on a data stream for a given window of time (T_1, T_2) . While the query processing problem has been explored in the context of data streams [6, 7, 10, 11, 13, 15], these methods are designed for processing of *historical* queries. These correspond to cases in which T_1 and T_2 are less than the current time t_0 . In this paper, we examine the problem of *predictive* query estimation. In the predictive query estimation problem, we attempt to estimate the selectivity of queries in a *future* time interval by making use of the current trends of the data stream. Thus, the generic data stream predictive selectivity estimation problem is defined as follows:

Definition 1. *Estimate the number of points in a data stream in the **future** time interval (T_1, T_2) , which satisfy the user-specified set of constraints \mathcal{U} .*

We note that predictive query processing is a significantly more difficult problem than historical query processing. This is because the historical behavior of the stream is already available, whereas the future behavior can only be estimated from the evolution trends in the data stream. This creates significant challenges in deciding on the nature of the summary information to be stored in order to estimate the responses to predictive queries. Some work has been done on performing high-level regression analysis to data cubes, but this work is designed for finding unusual trends in the data, and cannot be used for estimation of the selectivity of arbitrary user queries.

In order to solve the predictive querying problem, we use an approach in which we utilize local regression analysis in conjunction with storage of the summary covariance structure of different data localities. The local predictive approach stores a sufficient amount of summary statistics that it is able to create effective predictive samples in different data localities. These predictive samples can then be used in order to estimate the accuracy of the underlying queries. The sizes of the predictive samples can be varied depending upon the desired level of accuracy. We will show that such a local approach provides significant

advantages over the technique of global regression. This is because the latter cannot generate the kind of refined summary constructed by the local approach. The refined summary from the local approach provides the ability to perform significantly superior estimation of the future data points. Thus, this approach is not only flexible (in terms of being able to handle arbitrary queries) but is also more effective over a wide variety of data sets. Furthermore, the summaries can be processed very efficiently because of the small size of the data stored. Thus, the paper presents a flexible, effective and efficient approach to predictive data summarization.

This paper is organized as follows. In the next section, we will discuss the overall framework for the approach. We will also discuss the summary statistics which are required to be stored in order to implement this framework. In section 3, we will discuss the algorithms in order to create the summary statistics, and the process of performing the estimation. The empirical sections are discussed in section 4. Section 5 contains the conclusions and summary.

2 The Overall Summarization Framework

In order to perform predictive selectivity estimation, we need to store a sufficient amount of summary statistics so that the overall behavior of the data can be estimated. One way of achieving this goal is the use of histograms in order to store the summary information in the data. While traditional methods such as histograms and random sampling are useful for performing data summarization and selectivity estimation in a static data set, they are not particularly useful for *predicting* future behavior of high dimensional data sets. This is because of several reasons:

- (1) Histograms are not very effective for selectivity estimation and summarization of multi-dimensional sets. It has been estimated in [12] that for higher dimensional data sets, random sampling may be the only effective approach. However, random sampling is not very effective for predictive querying because the samples become stale very quickly in an evolving data stream.
- (2) Since the data may evolve over time, methods such as histograms are not very effective for data stream summarization. This is because when the behavior of the data changes substantially, the summary statistics of the current histograms may not effectively predict future behavior.
- (3) In this paper, we propose a very general model in which queries of *arbitrary* nature are allowed. Thus, the geometry of the queries is not restricted to particular kinds of rectangular partitions such as range queries. While summarization methods such as histograms are effective for rectangular range queries, they are not very effective for arbitrary queries. In such cases, random sampling is the only effective approach for static data sets. However, our empirical results will show that the random sampling approach is also not very useful in the context of an evolving data stream.

The overall approach in this paper emphasizes *predictive pseudo-data generation*. The essential idea in predictive pseudo-data generation is to store a

sufficient amount of summary statistics so that representative pseudo-data can be generated for the *future* interval (T_1, T_2) . The summary statistics include such parameters as the number of data points arriving, the mean along each dimension as well as relevant second order statistics which encode the covariance structure of the data. While such statistics are stored on a historical basis, they are used to estimate the corresponding statistics for any future time horizon (T_1, T_2) . Such estimated statistics can then be used to generate the sample pseudo-data records within the desired horizon (T_1, T_2) . We note that while the sample records (which are generated synthetically) will not represent the true records within the corresponding future time horizon, their aggregate statistics will continue to reflect the selectivity of the corresponding queries. In other words, the aggregation queries can be resolved by determining the number of pseudo-data points which satisfy the user query. The advantage of using pseudo-data is that it can be leveraged to estimate the selectivity of arbitrary queries which are not restricted to any particular geometry or form. This is not the case for traditional methods such as histograms which work with only a limited classes of queries such as rectangular range queries.

We will now describe the statistics of the data which are maintained by the stream summarization algorithm. The summary statistics consist of the first order statistics as well as the co-variance structure of the data. In order to introduce these summary statistics, we will first introduce some further notations and definitions. Let us consider a set of N records denoted by \mathcal{D} , each of which contains d dimensions. The records in the database \mathcal{D} are denoted by $\overline{X}_1 \dots \overline{X}_N$. The dimensions of each individual record \overline{X}_i are denoted by $(x_i^1 \dots x_i^d)$. For a subset of records \mathcal{Q} from the database \mathcal{D} , we define the summary statistics $\overline{Stat}(\mathcal{Q}) = (\overline{Sc}(\mathcal{Q}), \overline{Fs}(\mathcal{Q}), n(\mathcal{Q}))$, which defines the complete covariance structure of \mathcal{Q} . Specifically, $\overline{Sc}(\mathcal{Q})$ corresponds to the second order statistics of \mathcal{Q} , $\overline{Fs}(\mathcal{Q})$ corresponds to the first order structure, and $n(\mathcal{Q})$ corresponds to the number of data points. Each of these statistics are defined as follows:

(1) Product Sum (Second Order Covariance) Statistics: For each pair of dimensions i and j , we store the sum of the product for the corresponding dimension pairs. For the sake of convention (and to avoid duplication), we assume that $i \leq j$. The product sum for the dimension pairs i, j and record set \mathcal{Q} is denoted by $Sc_{ij}(\mathcal{Q})$. The corresponding value is defined as follows:

$$Sc_{ij}(\mathcal{Q}) = \sum_{k \in \mathcal{Q}} x_i^k \cdot x_j^k \quad (1)$$

The second order statistics is useful in computing covariance structure of the data records in \mathcal{Q} . We note that a total of $d \cdot (d + 1)/2$ values (corresponding to different values of i and j) need to be maintained in the vector $\overline{Sc}(\mathcal{Q})$.

(2) First Order Statistics: For each dimension i we maintain the sum of the individual attribute values. Thus, a total of d values are maintained. The value for the dimension i is denoted by $Fs_i(\mathcal{Q})$, and is defined as follows:

$$Fs_i(\mathcal{Q}) = \sum_{k \in \mathcal{Q}} x_i^k \quad (2)$$

We denote the vector $(Fs_1(\mathcal{Q}) \dots Fs_d(\mathcal{Q}))$ by $\overline{Fs(\mathcal{Q})}$.

(3) Zero Order Statistics: The zero order statistics $n(\mathcal{Q})$ contains one value and is equal to the number of records in \mathcal{Q} .

Thus, the total number of values which need to be stored in the vector $\overline{Stat(\mathcal{Q})}$ is equal to $d^2/2 + 3 \cdot d/2 + 1$. We make the following observations about the statistics which are stored:

Observation 21 *Each of the statistical values in $\overline{Stat(\mathcal{Q})}$ can be expressed as a linearly separable and direct sum of corresponding functional values over individual records.*

Observation 22 *The covariance C_{ij} between the dimensions i and j can be expressed in the following form:*

$$C_{ij} = Sc_{ij}(\mathcal{Q})/n(\mathcal{Q}) - Fs_i \cdot Fs_j/(n(\mathcal{Q}) \cdot n(\mathcal{Q})) \quad (3)$$

The first observation is important because it ensures that these statistical values can be efficiently maintained in the context of a data stream. This is because $\overline{Stat(\mathcal{Q})}$ can be computed as the simple arithmetic sum over the corresponding functional values over individual records. The second observation is important because it ensures that the covariance between the individual dimensions can be computed in terms of the individual statistical values. Thus, the statistical values provide a comprehensive idea of the covariance structure of the data. This is achieved by the method of principal component analysis. Since we will use this technique in our paper, we will discuss this method in detail below.

Let us assume that the covariance matrix of \mathcal{Q} is denoted by $\mathcal{C}(\mathcal{Q}) = [C_{ij}]$. Therefore, C_{ij} is equal to the covariance between the dimensions i and j . This covariance matrix is known to be positive-semidefinite and can be diagonalized as follows:

$$\mathcal{C}(\mathcal{Q}) = P(\mathcal{Q}) \cdot \Delta(\mathcal{Q}) \cdot P(\mathcal{Q})^T \quad (4)$$

Here the columns of $P(\mathcal{Q})$ represent the orthonormal eigenvectors, whereas $\Delta(\mathcal{Q})$ is a diagonal matrix which contains the eigenvalues. The eigenvectors and eigenvalues have an important physical significance with respect to the data points in \mathcal{Q} . Specifically, the orthonormal eigenvectors of $P(\mathcal{Q})$ represent an axis system in which the second order correlations of \mathcal{Q} are removed. Therefore, if we were to represent the data points of \mathcal{Q} in this new axis system, then the covariance between every pair of dimensions of the transformed data set would be zero. The eigenvalues of $\Delta(\mathcal{Q})$ would equal the variances of the data \mathcal{Q} along the corresponding eigenvectors. Thus, the orthonormal columns of the matrix $P(\mathcal{Q})$ define a new axis system of transformation on \mathcal{Q} , in which $\Delta(\mathcal{Q})$ is the new covariance matrix.

We note that the axis system of transformation represented by \mathcal{Q} is a particularly useful way to regenerate a sample of the data from the distribution represented by these statistics. This is because of the pairwise (second-order) independence between the dimensions of the transformed system. As a result, the data values along each of the transformed dimensions can also be generated

independently of one another.¹ Thus, the covariance matrix serves the essential purpose of summarizing the hidden structure of the data.

This structural description can be used to estimate and generate *future* samples of the data. In order to do so, we use the historical statistics in order to estimate the future statistics. The aim of this approach is to effectively re-generate the data samples, while taking into account the evolution of the data. In the next section, we will discuss the details of the approach and its application to the predictive query estimation problem. In order to actually store the statistics, we use both a *global* and a *local* predictive approach. In the global approach, the summary statistics of the *entire* data are stored at regular intervals. Let us denote the data points which have arrived till time t by $\mathcal{DS}(t)$. As each data point \overline{X}_t arrives, we add the corresponding values of $Fs(\{X_t\})$ and $Sc_{ij}(\{X_t\})$ to $\overline{Fs(\mathcal{DS}(t))}$ and $\overline{Sc(\mathcal{DS}(t))}$ respectively. The value of $n(\mathcal{DS}(t))$ is incremented by one unit as well. Thus, the additivity property of the statistics ensures that they can be maintained effectively in a fast stream environment.

In order to improve the accuracy of prediction, we use a *local* approach in which the prediction is performed separately on each data locality. In the local predictive approach, the statistics are maintained separately for each data locality. In other words, the data stream $\mathcal{DS}(t)$ is segmented out into q local streams which are denoted by $\mathcal{DS}_1(t)$, $\mathcal{DS}_2(t)$, \dots $\mathcal{DS}_q(t)$ respectively. We note that the statistics for each local segment are likely to be more refined than the statistics for the entire data stream. This results in more accurate prediction of the future stream summaries. Correspondingly, we will show that the selectivity results are also more accurate in the local approach. We note that the local predictive approach degenerates to the global case when the value of q is set to 1. Therefore, we will simply present the predictive query estimation method for the local case. The global case can be trivially derived from this description.

The process of maintaining the q local streams is illustrated in Figure 1. The first step is to create the initial set of statistics. This is achieved by storing an initial portion of the stream onto the disk. The number of initial data points stored on disk is denoted by *Init*. A k -means algorithm is applied to this set of points in order to create the initial clusters. Once the initial clusters $\mathcal{DS}_1(t) \dots \mathcal{DS}_q(t)$ have been determined, we generate the corresponding statistics $\overline{Stat(\mathcal{DS}_1(t))} \dots \overline{Stat(\mathcal{DS}_q(t))}$ from these clusters. For each incoming data point, we determine its distance to the centroid of each of the local streams. We note that the centroid of each local stream $\mathcal{DS}_i(t)$ can be determined easily by dividing the first order statistics $Fs(\mathcal{DS}_i(t))$ by the number of points $n(\mathcal{DS}_i(t))$. We determine the closest centroid to each data point. Let us assume that the index of the closest centroid is $min \in \{1, \dots, q\}$. We assign that data point to the corresponding local stream. At the same time, we update $\overline{Stat(\mathcal{DS}_{min}(t))}$ by adding the statistics of the incoming data point to it. At regular intervals of r , we also store the corresponding state of the statistics to disk. Therefore, the summary statistics at times $0, r, 2 \cdot r, \dots, t \cdot r \dots$ are stored to disk.

¹ This results in a second-order approximation which is useful for most practical purposes.

Algorithm *MaintainLocalStream*(Data Stream: $\mathcal{DS}(t)$,
TimeStamp: t);

begin
Store the first *Init* points from the data stream;
Apply k -means clustering to create l clusters;
Denote each cluster by $\mathcal{DS}_i(t)$ for $i \in \{1, \dots, q\}$;
Compute $\overline{Stat(\mathcal{DS}_i(t))}$ for $i \in \{1, \dots, q\}$;
for each incoming data point \overline{X} **do**
begin
Compute centroid of each $\mathcal{DS}_i(t)$ using $Stat(\mathcal{DS}_i(t))$;
Compute closest centroid index $min \in \{1, \dots, q\}$;
Assign \overline{X} to closest centroid and update
corresponding statistics $\mathcal{DS}_i(t)$;
end
end

Fig. 1. Local Stream Maintenance

3 The Predictive Query Estimation Method

In this section, we will discuss the predictive query estimation technique. Let us assume that the user wishes to find a response to the query \mathcal{R} over the time interval (T_1, T_2) . In order to achieve this goal, a statistical sample of the data needs to be generated for the interval (T_1, T_2) . This sample needs to be sufficiently predictive of the behavior of the data for the interval (T_1, T_2) . For this purpose, we also need to generate the summary statistics which are relevant to the future interval (T_1, T_2) .

Let us assume that the current time is $t_0 \leq T_1 < T_2$. In order to generate the statistical samples in the data, we utilize a history of length $T_2 - t_0$. In other words, we determine p evenly spaced snapshots in the range $(t_0 - (T_2 - t_0), t_0)$. These p evenly spaced snapshots are picked from the summary statistics which are stored on disk. In the event that the length of the stream is less than $(T_2 - t_0)$, we use the entire stream history and pick p evenly spaced snapshots from it. Let us assume that the time stamps for these snapshots are denoted by $b_1 \dots b_p$. These snapshots are also referred to as the *base snapshots*. Then, we would like to generate a *functional form* for $\overline{Stat(\mathcal{DS}_i(t))}$ for all values of t that are larger than t_0 . In order to achieve this goal, we utilize a local regression approach for each stream $\mathcal{DS}_i(t)$. Specifically, each component of $\overline{Stat(\mathcal{DS}_i(t))}$ is generated using a polynomial regression technique.

The generation of the zeroth order and first order statistics from $\overline{Stat(\mathcal{DS}_i(t))}$ is done slightly differently from the generation of second order statistics. A bursty data stream can lead to poor approximations of the covariance matrix. This is because rapid changes in the covariance values could occur due to either changes in the speed of arriving data points, or due to changes in inter-attribute correlations. In order to improve the accuracy further, we use the *correlation matrix*

Algorithm *EstimateQuery(Local Statistics: $\overline{\text{Stat}(\mathcal{DS}_i(b_j))}$),*
Query Interval: (T_1, T_2) , Query: \mathcal{U});
begin
 Derive $\eta(i, b_j), \mu_k(i, b_j), \sigma_k^2(i, b_j), \phi_{kl}(i, b_j)$ from
 $\overline{\text{Stat}(\mathcal{DS}_i(b_j))}$;
 Use local polynomial regression to generate
 functional forms $H(\eta, i, t), H(\mu, i, t), H(\sigma, i, t)$,
 and $H(\phi, i, t)$ for each stream i ;
 $s(i, T_1, T_2) = \int_{t=T_1}^{T_2} \frac{H(\eta, i, t)}{b_2 - b_1} dt$;
 Generate $s(i, T_1, T_2)$ pseudo-points for each stream
 using statistics η, μ, σ and ϕ ;
 Let $f(i, \mathcal{U})$ be the fraction of data points satisfying
 predicate \mathcal{U} from data stream \mathcal{DS}_i ;
 $ES(\mathcal{U}) = \sum_{i=1}^q s(i, T_1, T_2) \cdot f(i, \mathcal{U})$;
report($ES(\mathcal{U})$);
end

Fig. 2. The Query Estimation Algorithm

for the period between $(t_0 - (T_2 - t_0), t_0)$ as a more usable predictor of future behavior. We note that the *correlation matrix* is far less sensitive to the absolute magnitudes and rate of arrival of the data points, and is therefore likely to vary more slowly with time. The correlation between the dimensions i and j for a set of data points \mathcal{Q} is denoted by $\theta_{ij}(\mathcal{Q})$ and is essentially equal to the scaled covariance between the dimensions. Therefore, if $Cov_{ij}(\mathcal{Q})$ be the covariance between the dimensions i and j , we have:

$$\theta_{ij}(\mathcal{Q}) = \frac{Cov_{ij}(\mathcal{Q})}{\sqrt{Cov_{ii}(\mathcal{Q}) \cdot Cov_{jj}(\mathcal{Q})}} \quad (5)$$

We note that unlike the covariance, the correlation matrix is scaled with respect to the absolute magnitudes of the data values, and also the number of data points. This ensures that the correlation between the data points remains relatively stable for a bursty data stream with noise in it. The value of $\theta_{ij}(\mathcal{Q})$ lies between 0 and 1 for all $i, j \in \{1, \dots, d\}$.

The local predictive approach works on each local stream $\mathcal{DS}_i(t)$ separately, and determines the values of certain statistical variables at the base snapshot times $b_1 \dots b_p$. These statistical variables are as follows:

(1) For each local stream $\mathcal{DS}_i(t)$ and $j \in \{1 \dots p - 1\}$ we determine the number of data points arriving in the time interval $[b_j, b_{j+1}]$. This can be derived directly from the summary statistics stored in the snapshots, and is equal to $n(\mathcal{DS}_i(b_{j+1}) - n(\mathcal{DS}_i(b_j)))$. We denote this value by $\eta(i, b_j)$.

(2) For each local stream $\mathcal{DS}_i(t)$, $j \in \{1 \dots p - 1\}$, and $k \in \{1 \dots d\}$, we determine the mean of the data points which have arrived in the time interval $[b_j, b_{j+1}]$. This can again be estimated from the summary statistics stored in the snapshots at $b_1 \dots b_p$. The corresponding value is equal to $(Fs_k(\mathcal{DS}_i(b_{j+1})) -$

$Fs_k(\mathcal{DS}_i(b_j))/(n(\mathcal{DS}_i(b_{j+1})) - n(\mathcal{DS}_i(b_j)))$. We denote this value by $\mu_k(i, b_j)$.
(3) For each local stream $\mathcal{DS}_i(t)$, $j \in \{1 \dots p-1\}$, and dimension $k \in \{1 \dots d\}$, we determine the variance of the data points which have arrived in the time interval $[b_j, b_{j+1}]$. This is estimated by using a two step process. First we compute the second order moment of dimension k in interval $[b_j, b_{j+1}]$. This second order moment is equal to $(Sc_{kk}(\mathcal{DS}_i(b_{j+1})) - Sc_{kk}(\mathcal{DS}_i(b_j)))/(n(\mathcal{DS}_i(b_{j+1})) - n(\mathcal{DS}_i(b_j)))$. We denote this value by $SquareMoment_{kk}(i, b_j)$. Then, the variance in interval $[b_j, b_{j+1}]$ is equal to $SquareMoment_{kk}(i, b_j) - \mu_k(i, b_j)^2$. We denote this variance by $\sigma_k^2(i, b_j)$.

(4) For each local stream $\mathcal{DS}_i(t)$, $j \in \{1 \dots p-1\}$, and dimension pairs $k, l \in \{1 \dots d\}$, we determine the correlation between these dimension pairs. The correlation is determined by the expression $(SquareMoment_{kl}(i, b_j) - \mu_k(i, b_j) * \mu_l(i, b_j))/(\sigma_k(i, b_j) * \sigma_l(i, b_j))$. The correlation is denoted by $\phi_{kl}(i, b_j)$.

For each of the statistical values $\eta(i, b_j)$, $\mu_k(i, b_j)$, $\sigma_k^2(i, b_j)$, and $\phi_{kl}(i, b_j)$, we have $(p-1)$ different instantiations for different values of k and l . Therefore, for each of the different values, we would like to define a functional form in terms of the time t . In the following discussion, we will discuss the general approach by which the functional form is that of the expression η . Let us assume that the functional form is determined by the expression $H(\eta, i, t)$. Note that the value of $H(\eta, i, t)$ refers to the number of data points in an interval of length $(b_2 - b_1)$ and starting at the point t for data stream $\mathcal{DS}_i(t)$. We also assume that this functional form $H(\eta, i, t)$ is expressed polynomially as follows:

$$H(\eta, i, t) = a_m \cdot t^m + a_{m-1} \cdot t^{m-1} + \dots + a_1 \cdot t + a_0 \quad (6)$$

The coefficients $a_0 \dots a_m$ define the polynomial function for $H(\eta, i, t)$. These coefficients need to be approximated using known instantiations of the function $H(\eta, i, t)$. The order m is chosen based on the number $(p-1)$ of known instantiations. Typically, the value of m should be significantly lower than the number of instantiations $(p-1)$. For a particular data stream $\mathcal{DS}_i(t)$, we know the value of the function for $(p-1)$ values of t which are given by $t = b_1 \dots b_{p-1}$. Thus, for each $j = 1 \dots (p-1)$, we would like $H(\eta, i, b_j)$ to approximate $\eta(i, b_j)$ as closely as possible. In order to estimate the coefficients $a_0 \dots a_m$, we use a linear regression technique in which we minimize the mean square error of the approximation of the known instantiations. The process is repeated for each data stream $\mathcal{DS}_i(t)$ and each statistical² variable η , μ_k , σ_k , and ϕ_{kl} . Once these statistical variables have been determined, we perform the predictive estimation process. As mentioned earlier, it is assumed that the query corresponds to the future interval (T_1, T_2) . The first step is to estimate the total number of data points in the interval (T_1, T_2) . We note that the expression $H(\eta, i, t)$ corresponds to the number of points for data stream i in an interval of length³ $(b_2 - b_1)$. Therefore,

² We note that the fitting method need not have the same order for all the polynomials.

For the *zeroth*, first order, and second order statistics, we used second order, first order and *zeroth* order polynomials respectively. This turns out to be more useful in a bursty data stream in which these parameters can vary rapidly.

³ Since the intervals are evenly spaced, we note that $(b_j - b_{j-1})$ is equal to $(b_2 - b_1)$ for each value of $j \in \{1, \dots, (p-1)\}$.

the number of data points $s(i, T_1, T_2)$ in stream i for the interval (T_1, T_2) is given by the following expression:

$$s(i, T_1, T_2) = \int_{t=T_1}^{T_2} \frac{H(\eta, i, t)}{b_2 - b_1} dt \quad (7)$$

The value of $(b_2 - b_1)$ is included in the denominator of the above expression, since the statistical parameter η has been estimated as the number of data points lying in an interval of length $(b_2 - b_1)$ starting at a given moment in time. Once the number of data points in the time interval (T_1, T_2) for each stream $\mathcal{DS}_i(t)$ have been estimated, the next step is to generate $N_{samp}(i)$ sample points using the statistics η , μ , σ , and ϕ . The value of $N_{samp}(i)$ is chosen proportionally to $s(i, T_1, T_2)$ and should at least be equal to the latter. Larger values of $N_{samp}(i)$ lead to greater accuracy at the expense of greater computational costs. We will discuss the process of generating each sample point slightly later. Each of these sample points is tested against the user-defined query predicate, and the fraction of points $f(i, \mathcal{U})$ which actually satisfy the predicate \mathcal{U} from data stream $\mathcal{DS}_i(t)$ is determined. The final estimation $ES(\mathcal{U})$ for the query \mathcal{U} is given by the sum of the estimations over the different data streams. Therefore, we have:

$$ES(\mathcal{U}) = \sum_{i=1}^q s(i, T_1, T_2) \cdot f(i, \mathcal{U}) \quad (8)$$

It now remains to describe how each sample point from stream i is generated using the summary statistics.

The first step is to generate the time stamp of the sample point from stream $\mathcal{DS}_i(t)$. Therefore, we generate a sample time $t_s \in (T_1, T_2)$ from the relative density distribution $\eta(i, T)$. Once the sample time has been determined, all the other statistical quantities such as mean, variance, and correlation can be instantiated to $\mu_k(i, t_s)$, $\sigma_k^2(i, t_s)$, and $\phi_{kl}(i, t_s)$ respectively. The covariance $\sigma_{kl}(i, t_s)$ between each pair of dimensions k and l can be computed as:

$$\sigma_{kl}(i, t_s) = \sqrt{\sigma_k^2(i, t_s) \cdot \sigma_l^2(i, t_s)} \cdot \phi_{kl}(i, t_s) \quad (9)$$

The equation 9 relates the covariance with the statistical correlation by scaling appropriately with the product of the standard deviation along the dimensions k and l . This scaling factor is given by $\sqrt{\sigma_k^2(i, t_s) \cdot \sigma_l^2(i, t_s)}$. Once the covariance matrix has been computed, we generate the eigenvectors $\{\bar{e}_1 \dots \bar{e}_d\}$ by using the diagonalization process. Let us assume that the corresponding eigenvalues are denoted by $\{\lambda_1 \dots \lambda_d\}$ respectively. We note that λ_i denotes the variance along the eigenvector \bar{e}_i . Since the eigenvectors represent the directions of zero correlation⁴, the data values can be generated under the independence assumption in the transformed axis system denoted by $\{\bar{e}_1 \dots \bar{e}_d\}$. We generate the data in each such dimension using the uniform distribution assumption. Specifically, the

⁴ We note that the eigenvectors represent the directions of zero *second-order* correlation. However, a second-order approximation turns out to be effective in practice.

offset from the mean $\overline{\mu(i, t_s)}$ of stream $\mathcal{DS}_i(t)$ along $\overline{e_j}$ is generated randomly from a uniform distribution with standard deviation equal to $\sqrt{\lambda_j}$. While the uniform distribution assumption is a simplifying one, it does not lead to an additional loss of accuracy. Since each data stream $\mathcal{DS}_i(t)$ represents only a small locality of the data, the uniform distribution assumption within a locality does not affect the *global* statistics of the generated data significantly. Once the data point has been generated using this assumption, we test whether it satisfies the user query constraints \mathcal{U} . This process is repeated over a number of different data points in order to determine the fraction $f(i, \mathcal{U})$ of the data stream satisfying the condition \mathcal{U} . The overall process of query estimation is illustrated in Figure 2. It is important to note that the input set of constraints \mathcal{U} can take on any form, and are not restricted to any particular kind of query. Thus, this approach can also be used for a wide variety of problems that traditional selectivity estimation methods cannot solve. For example, one can use the pseudo-points to estimate statistical characteristics such as the mean or sum across different records. We note that we can reliably estimate most first order and second order parameters because of the storage of second-order covariance structure. A detailed description of these advanced techniques is beyond the scope of this paper and will be discussed in future research. In the next section, we will discuss the effectiveness and efficiency of the predictive summarization procedure for query selectivity estimation.

4 Empirical Results

We tested our predictive summarization approach over a wide variety of real data sets. We tested our approach for the following measures:

- (1) We tested the accuracy of the estimation procedure. The accuracy of the estimation was tested in various situations such as that of a rapidly evolving data stream or a relatively stable data stream. The aim of testing different scenarios was to determine how well these situations adjusted to the predictive aspect of the estimation process.
- (2) We tested the rate of summarization of the stream processing framework. These tests determine the workload limits (maximum data stream arrival rate) that can be handled by the pre-processing approach.
- (3) We tested the efficiency of the query processing approach for different data sets. This is essential to ensure that individual users are able to process offline queries in an efficient manner.

The accuracy of our approach was tested against two techniques:

- (1) We tested the technique against a random sampling approach. In this method, we estimated the query selectivity of \mathcal{U} corresponding to future interval (T_1, T_2) by using a sample of data points in the most recent window of size $(T_2 - T_1)$. This technique can work poorly in a rapidly evolving data stream, since the past window may not be a very good reflection of future behavior.
- (2) We tested the local technique against the global technique in terms of the quality of query estimation. The results show that the local technique was sig-

nificantly more effective on a wide variety of data sets and measures. This is because the local technique is able to estimate parameters which are specific to a given segment. This results in more refined statistics which can estimate the evolution in the stream more effectively.

4.1 Test Data Sets

We utilized some real data sets to test the effectiveness of the approach. A good candidate for such testing is the KDD-CUP'99 Network Intrusion Detection stream data set. The Network Intrusion Detection data set consists of a series of TCP connection records from two weeks of LAN network traffic managed by MIT Lincoln Labs. Each record can correspond to a normal connection, an intrusion or an attack. This data set evolves rapidly, and is useful in testing the effectiveness of the approach in situations in which the characteristics of the data set change rapidly over time.

Second, besides testing on the rapidly evolving network intrusion data stream, we also tested our method over relatively stable streams. The KDD-CUP'98 Charitable Donation data set shows such behavior. This data set contains 95412 records of information about people who have made charitable donations in response to direct mailing requests, and clustering can be used to group donors showing similar donation behavior. As in [9], we will only use 56 fields which can be extracted from the total 481 fields of each record. This data set is converted into a data stream by taking the data input order as the order of streaming and assuming that they flow-in with a uniform speed.

The last real data set we tested is the *Forest CoverType* data set and was obtained from the UCI machine learning repository web site [16]. This data set contains 581012 observations and each observation consists of 54 attributes, including 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables. In our testing, we used all the 10 quantitative variables.

4.2 Effectiveness results

We first tested the prediction accuracy of the approach with respect to the global approach and a random sampling method. In the sampling method, we always maintained a random sample of the history of the data stream. When a query was received, we used the random sample from the most recent history of the stream in order to estimate the effectiveness of the queries. The queries were generated as follows. First, we randomly picked $k' = d/2$ dimensions in the data with the greatest standard deviation. From these dimensions, we picked k dimensions randomly, where k was randomly chosen from $(2, 4)$. The aim of pre-selecting widely varying dimensions was to pick queries which were challenging to the selectivity estimation process. Then, the ranges along each dimension were generated from a uniform random distribution. In each case, we performed the tests over 50 such randomly chosen queries and presented the averaged results.

In Figure 3, we have illustrated the predictive error of the Network Intrusion data set with stream progression. In each group of stacked bars in the chart,

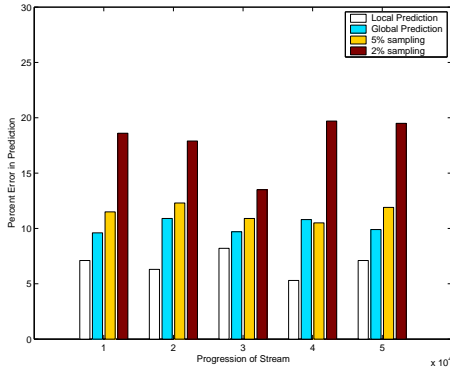


Fig. 3. Predictive Error of Different Methods with Stream Progression (Network Intrusion Data Set)

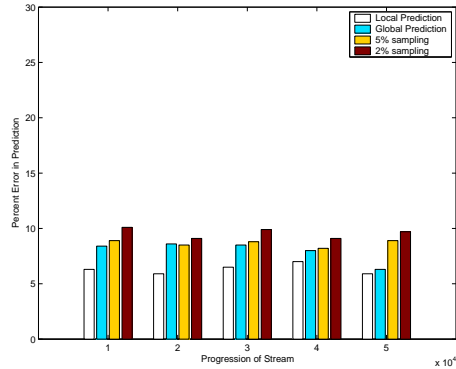


Fig. 4. Predictive Error of Different Methods with Stream Progression (Charitable Donation Data Set)

we have illustrated the predictive error of each method. Different stacks correspond to different time periods in the progression of the stream. The predictive accuracy is defined as the difference between the true and predictive selectivity as a percentage of the true value. On the X-axis, we have illustrated the progression of the data stream. The predictive error varied between 5% and 20% over the different methods. It is clear that in each case, the local predictive estimation method provides the greatest accuracy in prediction. While the local method is consistently superior to the method of global approach, the latter is usually better than pure random sampling methods. This is because random sampling methods are unable to adjust to the evolution in the data stream. In some cases, the 5% random sampling method was slightly better than global method. However, in all cases, the local predictive estimation method provided the most accurate result. Furthermore, the 2% sampling method was the least effective in all cases. The situations in which the 5% sampling method was superior to global method were those in which the stream behavior was stable and did not vary much over time.

In order to verify this fact, we also performed empirical tests using the charitable donation data set which exhibited much more stable behavior than the Network Intrusion Set. The results are illustrated in Figure 4. The stable behavior of the charitable donation data set ensured that the random sampling method did not show much poorer performance than the predictive estimation methods. However, in each case, the local predictive estimation method continued to be significantly superior to other techniques. In some cases, the 5% sampling method was slightly better than the global estimation method. Because of the lack of evolution of the data set, the sampling method was relatively more robust. However, it was still outperformed by the local predictive method in all cases.

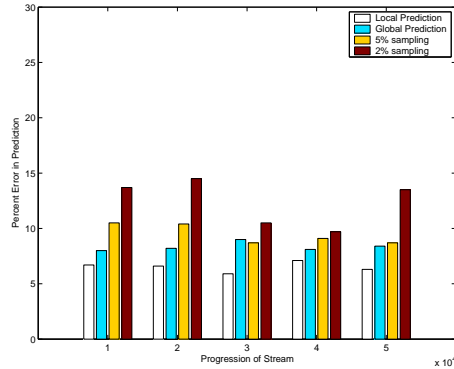


Fig. 5. Predictive Error of Different Methods with Stream Progression (Forest Cover Data Set)

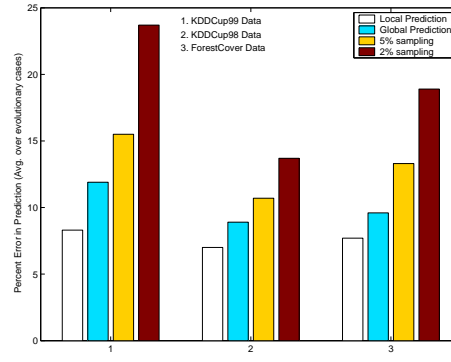


Fig. 6. Predictive Error of Different Methods with Stream Progression (High Evolution Cases)

Finally, the results for the forest cover data set are illustrated in Figure 5. This data set showed similar relative trends between the different methods. However, the results were less skewed than the network intrusion data set. This is because the network intrusion data set contained sudden bursts of changes in data behavior. These bursts correspond to the presence of intrusions in the data. These intrusions also show up in the form of sudden changes in the underlying data attributes. While the forest cover data set evolved more than the charitable donation data set, it seemed to be more stable than the network intrusion data set. Correspondingly, the relative performance of the sampling methods improved over that for the network intrusion data set, but was not as good as the charitable donation data set. As in the previous cases, the predictive estimation approach dominated significantly over other methods.

We also tested the effectiveness of the approach in specific circumstances where the data was highly evolving. In order to model such highly evolving data sets, we picked certain points in the data set at which the class distribution of the data stream showed a shift. Specifically, when the percentage presence of the dominant class in successive blocks of 1000 data points showed a change of greater than 5%, these positions in the data stream were considered to be highly evolving. All queries to be tested were generated in a time interval which began at a lag of 100 data points from the beginning of the shift. This ensured that the queries followed a region with a very high level of evolution. For each data set, ten such queries were generated using the same methodology described earlier. The average selectivity error over the different data sets was reported in Figure 6. Because of the greater level of evolution in the data set, the absolute error values are significantly higher. Furthermore, the random sampling method performed poorly for all three data sets. The results were particularly noticeable for the network intrusion data set. This is because the random sampling approach uses only the history of past behavior. This turned out to be poor surrogate in this

case. Since the random sampling approach relied exclusively on the history of the data stream, it did not provide very good results in cases in which the stream evolved rapidly. These results show that the predictive estimation technique was a particularly useful method in the context of a highly evolving data stream.

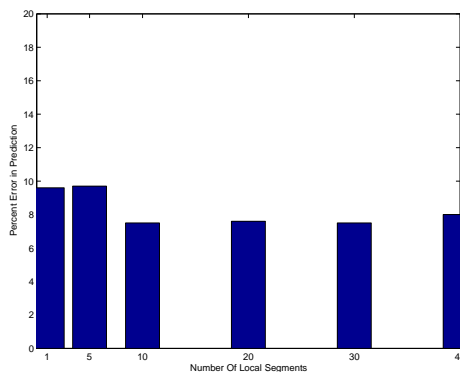


Fig. 7. Predictive Error with Increasing Number of Local Segments (Network Intrusion Data Set)

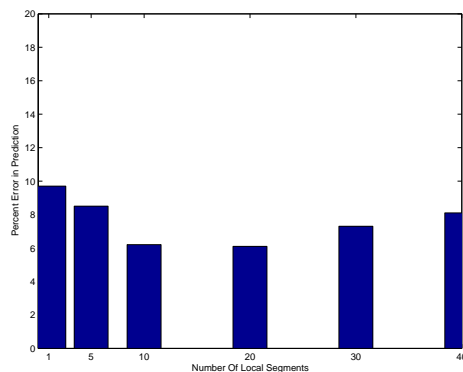


Fig. 8. Predictive Error with Increasing Number of Local Segments (Charitable Donation Data Set)

We also tested the effectiveness of the predictive estimation analysis with increasing number of segments in the data stream. The results for the network intrusion data set are presented in Figure 7. These results show that the error in estimation reduced with the number of segments in the data stream, but levelled off after the use of 7 to 8 segments. This is because the use of an increasing number of segments enhanced the power of data locality during the parameter estimation process. However, there was a limit to this advantage. When the number of clusters was increased to more than 20, the error rate increased substantially. In these cases, the number of data points from each cluster (which were used for the polynomial fitting process) reduced to a point which leads to a lack of statistical robustness. The results for the charitable donation data set are presented in Figure 8. While the absolute error numbers are slightly lower in each case, the trends are quite similar. Therefore, the results show that it is a clear advantage to use a large number of segments in order to model the behavior of each data locality.

4.3 Stream Processing and Querying Efficiency

In this section, we will study the processing efficiency of the method, and its sensitivity with respect to the number of segments used in the data stream. The processing efficiency refers to the online rate at which the stream can be processed in order to create and store away the summary statistics generated by the method. The processing efficiency was tested in terms of the number of

data points processed per second with stream progression. The results for the case of the network intrusion and charitable donation data sets are illustrated in Figure 9. On the X-axis, we have illustrated the progression of the data stream. The Y-axis depicts the processing rate of the stream in terms of the number of data points processed every minute. It is clear that the algorithm was stable throughout the execution of the data stream. The processing rate for the network intrusion data set was higher because of its lower dimensionality. Furthermore, the execution times were relatively small, and several thousand data points were processed per minute. This is because the stream statistics can be updated using relatively straightforward additive calculations on each point. We have drawn multiple plots in Figure 9 illustrating the effect of using the different data sets.

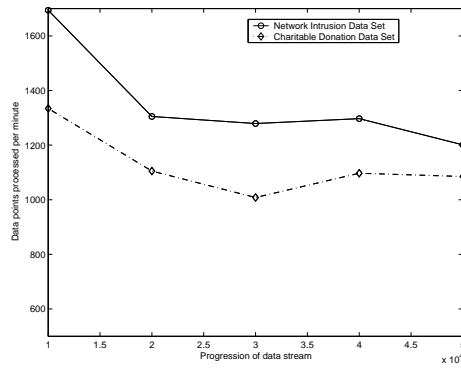


Fig. 9. Stream Processing Time with Data Stream Progress

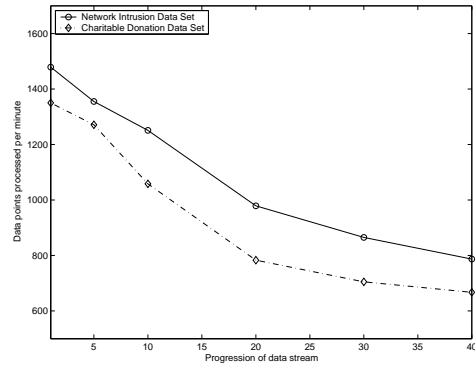


Fig. 10. Stream Processing Time with Increasing Number of Local Segments

In order to illustrate the effect of using different number of segments, we have illustrated the variation in processing rate with the number of stream segments in Figure 10. Both data sets are illustrated in this figure. As in the previous case, the lower dimensionality of the network intrusion data set resulted in higher processing efficiency. It is clear that the number of data points processed per second reduces with increasing number of segments. This is because of the fact that the time for finding the closest stream segment (in order to find which set of local stream segment statistics to update) was linear in the number of local stream segments. However, the majority of the time was spent in the (fixed) cost of updating stream statistics. This cost was independent of the number of stream segments. Correspondingly, the overall processing rate was linear in the number of stream segments (because of the cost of finding the closest stream segment), though the fixed cost of updating stream statistics (and storing it away) tended to dominate. Therefore, the results of Figure 10 illustrate that the reduction in processing rate with increasing number of stream segments is relatively mild.

Finally, we studied the efficiency of querying the data stream. We note that the querying efficiency depends upon the number of segments stored in the data

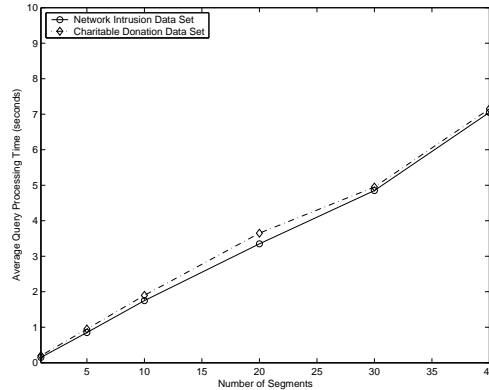


Fig. 11. Stream Query Time with Increasing Number of Local Segments

stream. This is because the statistics need to be estimated separately for each stream segment. This requires separate processing of each segment and leads to increased running times. We have presented the results for the Charitable Donation and Network Intrusion Data data set in Figure 11. In order to improve the accuracy of evaluation, we computed the running times over a batch of one hundred examples and reported the *average* running times per query on the Y -axis. On the X -axis, we have illustrated the number of stream segments used. It is clear that in each case, the running time varied between 0.3 and 8 seconds. A relevant observation is that the most accurate results for query responses is obtained when about 7-10 segments were used in these data sets. For these cases, the query response times were less than 2 seconds in all cases. Furthermore, we found the running time to vary linearly with the number of stream segments. The network intrusion and the charitable donation data sets showed similar results except that the running times were somewhat higher in the latter case. This is because of the higher dimensionality of the latter data set which increased the running times as well.

5 Conclusions and Summary

In this paper, we discussed a method for predictive query estimation of data streams. The approach used in this paper can effectively estimate the changes in the data stream resulting from the evolution process. These changes are incorporated in the model in order to perform the predictive estimation process. We note that the summarization approach in this paper is quite general and can be applied to arbitrary kinds of queries as opposed to simple techniques such as range queries. This is because the summarization approach constructs pseudo-data which can be used in conjunction with an arbitrary query. While this scheme

has been developed and tested for query estimation, the technique can be used for any task which requires predictive data summarization. We tested the scheme on a number of real data sets, and compared it against an approach based on random sampling. The results show that our scheme significantly outperforms the method of random sampling as well as the global approach. The strength of our approach arises from its careful exploitation of data locality in order to estimate the inter-attribute correlations. In future work, we will utilize the data summarization approach to construct visual representations of the data stream.

References

1. Aggarwal C. C.: A Framework for Diagnosing Changes in Evolving Data Streams, ACM SIGMOD Conference, (2003) 575–586.
2. Aggarwal C. C., Han J., Wang J., Yu P.: A Framework for Clustering Evolving Data Streams, VLDB Conference, (2003) 81–92.
3. Babcock B., Babu S., Datar M., Motwani R., Widom J.: Models and Issues in Data Stream Systems, ACM PODS Conference, (2002) 1–16.
4. Chen Y., Dong G., Han J., Wah B., Wang J.: Multi-Dimensional Regression Analysis of Time Series Data Streams, VLDB Conference, (2002) 323–334.
5. Cortes C., Fisher K., Pregibon D., Rogers A., Smith F.: Hancock: A Language for Extracting Signatures from Data Streams, ACM KDD Conference, (2000) 9–17.
6. Dobra A., Garofalakis M., Gehrke J., Rastogi R.: Processing Complex Aggregate Queries over Data Streams, ACM SIGMOD Conference, (2002) 61–72.
7. Dobra A., Garofalakis M., Gehrke J., Rastogi R.: Sketch Based Multi-Query Processing Over Data Streams, EDBT Conference, (2004) 551–568.
8. Domingos P., Hulten G.: Mining High-Speed Data Streams, ACM KDD Conference, (2000) 71–80.
9. Farnstrom F., Lewis J., Elkan C.: Scalability for Clustering Algorithms Revisited, ACM SIGKDD Explorations, Vol. 2(1), (2000) 51–57.
10. Gilbert A. C., Kotidis Y., Muthukrishnan S., Strauss M. J.: Surfing Wavelets on Streams: One-pass Summaries for Approximate Aggregate Queries, VLDB Conference, (2001) 79–88.
11. Gilbert A. C., Kotidis Y., Muthukrishnan S., Strauss M. J.: How to Summarize the Universe: Dynamic Maintenance of Quantiles. VLDB Conference, (2002) 454–465.
12. Gunopulos D., Kollios G., Tsotras V., Domeniconi C.: Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes. ACM SIGMOD Conference, (2000) 463–474.
13. Manku G. S., Motwani R.: Approximate Frequency Counts over Data Streams. VLDB Conference, (2002), 346–357.
14. O’Callaghan L., Mishra N., Meyerson A., Guha S., Motwani R.: Streaming-Data Algorithms For High-Quality Clustering, IEEE ICDE Conference, (2002) 685–696.
15. Vitter J., Wang M.: Approximate Computation of Multidimensional Aggregates of Sparse Data using Wavelets. ACM SIGMOD Conference, (1999) 193–204.
16. <http://www.ics.uci.edu/~mllearn>.