2D-STR: Reducing Spatio-Temporal Traffic Datasets by Partitioning and Modelling

Liam Steadman[1], Nathan Griffiths[1], Stephen Jarvis[1], Stuart McRobbie[2] and Caroline Wallbank[2]
[1]*Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK*
[2]*TRL, Wokingham, RG40 3GA, UK*
{*l.steadman, nathan.griffiths, s.a.jarvis*}*@warwick.ac.uk*, {*smcrobbie, cwallbank*}*@trl.co.uk*

Keywords: Spatio-Temporal Data, Data Reduction, Data Partitioning

Abstract: Spatio-temporal data generated by sensors in the environment, such as traffic data, is widely used in the transportation domain. However, learning from and analysing such data is increasingly problematic as the volume of data grows. Therefore, methods are required to reduce the quantity of data needed for multiple types of subsequent analysis without losing significant information. In this paper, we present the 2-Dimensional Spatio-Temporal Reduction method (2D-STR), which partitions the spatio-temporal matrix of a dataset into regions of similar instances, and reduces each region to a model of its instances. The method is shown to be effective at reducing the volume of a traffic dataset to $<5\%$ of its original volume whilst achieving a normalise root mean squared error of $<5\%$ when reproducing the original features of the dataset.

1 INTRODUCTION

Spatio-temporal data generated by sensors in the environment is used widely in many domains. In the transportation domain, traffic data measures the speed and flow of a traffic network at a specific location and time. Analysing and learning from traffic data allows us to understand the relationship between different roads or areas of a road network, as well as understand the temporal characteristics of the network. Common tasks in analysing traffic data include: (i) imputing missing instances, instances between sensors or time intervals; (ii) identifying unusual behaviours, such as sensors that perform unexpectedly or periods of time wherein instances do not fit expected trends; (iii) calculating statistics, such as high and low values for a time period or calculating the variance within a time period from the expected trend; (iv) comparing time periods or sensors, for example performing a month-on-month time series analysis; and (v) predicting future instances.

The volume of traffic data in the transportation domain has increased significantly in recent years, and this presents several challenges for data scientists. Processing larger spatio-temporal datasets requires more processing time and is sometimes infeasible entirely. Therefore, methods are required to reduce the volume of the data to be processed whilst minimising the error introduced in later analysis or modelling. The aim is not necessarily to compress the data, since this may require decompression before the data can be used again. Rather, we aim to summarise the data in a manner such that analysis and processing can be performed directly on the summarised data.

In this paper, we propose a new dataset reduction method, 2-Dimensional Spatio-Temporal Reduction (2D-STR), for decreasing the volume of a traffic dataset whilst minimising the information lost in the process. The method uses the variance within the data to partition the dataset, then models each resulting region using an appropriate modelling technique. The algorithm iteratively trades information loss for quantity of data output until an objective function is minimised. 2D-STR is designed to be extensible given the uses of the reduced dataset in further analysis, and is evaluated using data recorded from traffic count sensors within a motorway network in England.

The remainder of this paper is structured as follows. Section 2 reviews existing methods for reducing the volume of a traffic dataset. Section 3 formalises the problem to be solved and introduces the notation used in this paper, and Section 4 describes the proposed data reduction method. Sections 5 and 6 evaluate the effectiveness of 2D-STR on a set of traffic datasets taken from locations in England and compares it with other techniques. Finally, Section 7 concludes the paper and gives some future directions of the work.

2 RELATED WORK

It is often infeasible to process a spatio-temporal dataset in its raw form because of the volume of data present. Therefore, several techniques exist for reducing the quantity of data in order to facilitate faster modelling and analysis. In this section, we give an overview of existing methods for achieving this aim.

2.1 Selection and Engineering of Features and Instances

Many existing methods for reducing datasets focus on removing or retaining a subset of features or instances from the original dataset. Feature selection techniques can be separated into three categories. First, filter methods rank features in the dataset according to a relevancy criterion, such as Shannon entropy, and remove any features which score below a defined threshold. Second, wrapper methods use search algorithms to find the optimal subset of features according to an objective function. The third category, embedded methods, incorporate feature selection as part of the training process of machine learning techniques. Several feature selection techniques for real-valued data exist, and a number of reviews of these can be found in related literature (Xue et al., 2016; Li et al., 2017). Similar to feature selection, instance selection techniques have been surveyed in the context of different application domains (Garcia et al., 2012; Acampora et al., 2016; Mukahar and Rosdi, 2018).

In contrast to selection techniques, feature engineering techniques (often referred to as *feature extraction*) project the original features of a dataset onto a new feature space, often of a differing dimensionality. Some feature engineering techniques map the original dataset onto a smaller set of engineered features, thus reducing the volume of the dataset. The best mapping is that which optimises an objective criterion, such as explained variance or accuracy when combined with modelling. Feature extraction methods can be grouped into linear algorithms, such as Principal Components Analysis (PCA) (Pearson, 1901) and Linear Discriminant Analysis (LDA) (Martinez and Kak, 2001), and non-linear algorithms, such as Isomap and Locally Linear Embedding (LLE) (Tenenbaum et al., 2000; Roweis and Saul, 2000).

Instance engineering techniques (often referred to as *feature abstraction* or *prototyping*) create a smaller set of *prototype* instances which represent the original instances. Prototyping has been shown to be effective in reducing the size of training sets required for tasks such as k-Nearest Neighbour classification (Ougiaroglou and Evangelidis, 2012).

Whilst effective at reducing the volume of a dataset, feature and instance selection techniques may remove data and patterns that are significant for subsequent analysis. They fail to capture the spatial and temporal nuances of the data and require the user to have knowledge of the modelling or analysis they will perform ahead of time. Instead, it may be more beneficial to retain information about all features and instances. Furthermore, the features output by extraction techniques are often incomprehensible to humans and require mapping back to the original feature space. The instances resulting from instance engineering may not exist in the original dataset and this may cause problems in later processing. Finally, selection and engineering techniques do not take advantage of the spatio-temporal correlations in the data.

2.2 Data Sketching

Data sketching techniques create query-specific summaries using a fixed number of passes over the data. Many of these techniques require just one pass over the data and so are very fast to compute.

Many sketching techniques focus on counting items, such as the Count-Min sketch and its adaptation for real-valued data (Cormode and Muthukrishnan, 2005; Sisovic et al., 2018), and determining membership, such as the Bloom Filter (Bloom, 1970). Other techniques, such as the HyperLogLog (HLL) algorithm, focus on answering cardinality queries (Flajolet et al., 2007). Most sketching techniques do not consider the spatial and temporal nature of the data and do not support analysis questions such as those presented in Section 1.

In the spatio-temporal domain, methods have been proposed that combine instance selection data sketching with the Kalman filter to track large-scale spatio-temporal processes (Berberidis and Giannakis, 2015; Berberidis and Giannakis, 2017). Furthermore, Tai *et al.* presented a sketching method for building linear classifiers over a spatio-temporal dataset (Tai et al., 2018). This method destroys features which are not heavily weighted by the linear classifier and so prevents analysis of all features. Sketching techniques are still limited in the analyses or later modelling they support (Cormode et al., 2012). They are created specifically for particular queries and since the original dataset is destroyed after the sketch is created, it is not possible to recreate the data for other analyses.

2.3 Data Reduction by Modelling

Whilst the techniques discussed above result in the loss of instances or features, some techniques have

been investigated for reducing a dataset using statistical modelling. The IDEALEM algorithm partitions a data stream into blocks of a fixed size (Wu et al., 2017). Key statistical properties about these blocks, such as min, max and average values, are then used to identify similar blocks which can be reduced to the same model. By processing each of its prototype blocks, IDEALEM allows us to identify unusual temporal periods that do not fit expected trends. In the same way, generating statistics over the stream is faster when compared to processing the raw dataset. IDEALEM also enables an element of time series analysis and comparison of different sensors or time periods. However, since prototype blocks are retained in the form of raw data, and the method only considers the temporal nature of spatio-temporal data, IDEALEM does not permit spatio-temporal imputation.

Similar to IDEALEM, the ISABELA algorithm partitions each feature into fixed size spatial windows and sorts the instances into ascending order within each window (Lakshminarasimhan et al., 2011). A B-spline curve is then fitted to each window and the parameters for the curve of each window stored using temporal encoding. ISABELA permits the generation of statistics for given temporal and spatial periods, provided that those periods cover the spatial and temporal windows used by ISABELA exactly and, in the same way, partially permits identifying unusual spatio-temporal regions. However, since the instances in each window are stored in ascending order by value, a mapping from this ordering to the temporal ordering also needs to be stored. Many sensor datasets, such as traffic datasets, are more smooth and cyclic than the scientific data ISABELA was designed for, and so can be modelled effectively without needing to be sorted (Birvinskas et al., 2012).

Deep autoencoders have also been used to model the temporal features of spatio-temporal datasets (Wang et al., 2016). The Sparse Autoencoder (SAE) has been used to reliably estimate missing data in spatio-temporal sensor datasets (Wong et al., 2014). This fitting of a summary, which minimises the root mean square error (RMSE) over instances in both the discrete spatial and temporal dimensions, is able to impute missing values given other instances from the same time. It may be possible to adapt this approach to incorporate multiple time instances, e.g. the whole dataset, and store the autoencoder weights for the purposes of reproducing the dataset. However, autoencoder weights can be difficult to interpret and so prevent manual analysis of the reduced dataset.

In the domain of traffic dataset analysis, Pan *et al.* have proposed a two-part algorithm that summarises a spatio-temporal traffic sensor dataset (Pan et al., 2010). Their method creates a spatio-temporal signature of the dataset using a technique such as wavelet decomposition, and probabilistically stores separately the outliers that fall outside an acceptable error margin of this signature. Whilst this technique accounts for the cyclic and seasonal natures of traffic data, it performs poorly in reducing regions containing many outliers. For example, instances from national holidays (temporal domain) and areas of construction work (spatial domain) which are known to break regular traffic cycles will be labelled as outliers. Therefore, such instances will need to be stored in their raw form thereby hindering the reduction of data volume. The algorithm permits analysis of unusual periods, by examining the signatures and outliers of the period, comparison of time series, and statistics to be generated from the data in its reduced format.

In our review of literature these are the only examples of reduction by modelling for spatio-temporal data, and the technique presented by Pan *et al.* is the only example for traffic data. This suggests that the topic of reducing a dataset to a set of spatio-temporal regions and models has yet to be explored deeply.

3 SPATIO-TEMPORAL DATASET REDUCTION

Many spatio-temporal datasets, such as traffic datasets, contain spatial areas and temporal periods which exhibit low amounts of change. That is, the instances within these areas and periods have low variance when compared to the variance of the entire dataset. For example, a traffic sensor may record similar average speeds and vehicle counts for several consecutive 15-minute windows throughout the day. To decrease the quantity of data to be analysed or modelled in these cases, we can group similar consecutive instances together to form spatio-temporal regions in which the data exhibits low variance. We do not wish to lose information of any features, sensors or time intervals, however we do wish to decrease the quantity of data present in the resulting dataset. It is desirable to support many types of analysis or modelling, and multiple passes of the dataset are permitted. By identifying regions of similar instances in the dataset and reducing these regions to models, these requirements are met. The nuances of the original instances can be maintained and answering queries on the dataset is still supported. This section formalises this approach and the notation used in this paper.

A traffic dataset D is a set of instances in the $T \times S$ space, where T is a set of discrete time intervals and S is a discrete set of spatially fixed sensors along the

road. We assert that D may contain missing instances, i.e $|D| \leq |T| \cdot |S|$, and that T and S are ordered, thus it is possible define ranges over them. We can view D as a 2-dimensional matrix (Figure 1(a)), where columns represent the ordered set of discrete sensors and rows represent the ordered set of discrete time intervals. By permitting missing instances we also allow for sensors to be asynchronous.

Each instance $d_{t,s}$ is a vector of values over $F \in \mathbb{N}$ features, $d_{t,s} = (d_{t,s}(1), ..., d_{t,s}(F))$. For example, these features may be vehicle count and average vehicle speed. In this work, for the purposes of generality, we assume the non-referencing features in D are real-valued. Thus, $D : T \times S \rightarrow \mathbb{R}^F$. Techniques exist for representing binary and categorical features as real-valued data, and appropriate clustering algorithms can be used for binary and categorical features.

We wish to find the set R of non-overlapping regions in the $T \times S$ space, where each region R_i is a rectangle (Figure 1(d)). Each region R_i has a defined beginning and ending time, t_b and t_e, and a beginning and ending sensor, s_b and s_e. Furthermore, we use r_i to denote the subset of instances from dataset D that belong to region R_i, $r_i = \{d_{t,s} \in D | t_b \leq t \leq t_e, s_b \leq s \leq s_e : (t_b, t_e, s_b, s_e) = R_i\}$.

Each region R_i is associated with a model M_i which is fitted to the instances r_i. The reduction uses a single modelling technique, which is able to characterise the spatio-temporal nuances of the dataset, for all models. We refer to the set of summary models in the reduction as $M = (M_1, ..., M_{|R|})$, and denote $|M_i|$ to be the number of coefficients used to store model M_i. Finally, we use the term *reduction* to refer to a pair of regions and their models (R, M). The notation used in this paper is summarised in Table 1.

In reducing D to (R, M) we wish to minimise the information lost. We refer to the information lost by reducing D to reduction (R, M) as $e(D, (R, M))$. One method of measuring information loss is to recreate the dataset as D' using the set of region models (R, M), and then use an appropriate measure of the difference between each instance in D and its corresponding instance in D', i.e. $e(D, (R, M)) = e(D, D')$. A simple example is the Mean Absolute Percentage Error (MAPE) averaged across the dataset:

$$e_{\text{MAPE}}(D, D') = \frac{1}{|D| \cdot F} \sum_{d_{t,s} \in D} \sum_{j=1}^{F} \left| \frac{d_{t,s}(j) - d'_{t,s}(j)}{d_{t,s}(j)} \right| \tag{1}$$

An alternative measure is the Normalised Root Mean Square Error (NRMSE) averaged across the dataset:

$$e_{\text{NRMSE}}(D, D') = \frac{1}{F} \sum_{j=1}^{F} \frac{\psi(j, D, D')}{range(j)} \tag{2}$$

Table 1: Notation used in this paper

Symbol	Definition
D	Original dataset over the discrete ordered set of time intervals T and set of sensors S
F	The number of real-valued features in D, excluding the referencing features T and S
$d_{t,s}$	An individual instance in D
$d_{t,s}(j)$	Value of $d_{t,s}$ for feature j
R	A set of non-overlapping spatio-temporal regions on D
R_i	An individual spatio-temporal region in R
r_i	Set of instances of dataset D contained in region R_i
M	Set of summary models belonging to regions R of dataset D
M_i	Summary model of region R_i, fitted over the instances r_i, with the number of coefficients used to store M_i represented as $\|M_i\|$
(R, M)	A reduction of dataset D
$e(D, (R, M))$	Error introduced after D is reduced to regions R and their models M
$q(D, (R, M))$	Ratio of storage required for regions R and summary models M compared to the original dataset D
$h(D, (R, M))$	Objective function used to find the best reduction given parameter α, the constant that prioritises between $e(D, (R, M))$ and $q(D, (R, M))$

where,

$$\psi(j, D, D') = \sqrt{\frac{\sum_{t \in T} \sum_{s \in S} (d_{t,s}(j) - d'_{t,s}(j))^2}{|D|}}$$

and $range(j) = max_{t,s}(d_{t,s}(j)) - min_{t,s}(d_{t,s}(j))$. It is preferable to use the MAPE metric when the error of an instance relative to its original values is important. Conversely it is preferable to use the NRMSE metric when the error of an instance relative to the range of values observed for the feature is preferred.

Whilst minimising the information lost across D, we also wish to minimise the storage cost of the data. In the case of the original dataset this is given by the number of instances multiplied by the number of features including the spatial and temporal referencing features, as shown in Equation 3. In the case of the reduced dataset, the data output is a start and end value of each region in the spatial and temporal dimensions, along with the coefficients required to store the model for each region. Since a single modelling technique is

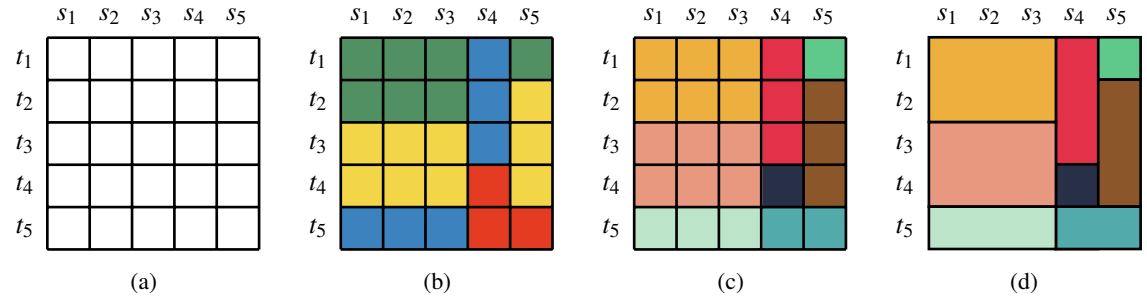


Figure 1: Stages of the hierarchical partitioning technique: **(a)** the raw data; **(b)** the data clustered into 4 clusters using hierarchical clustering (clusters coloured green, yellow, blue and red); **(c)** rectangles of contiguous clusters are found in the spatio-temporal feature space; **(d)** the resulting 8 regions, where instances within each region belong to the same cluster.

assumed to be used throughout the dataset there is no need to store the modelling technique used. The storage required for a reduced dataset is shown in Equation 4, and we use the quotient of these two measures to define the storage ratio, as shown in Equation 5.

$$\text{storage}(D) = |D| \cdot (|F| + 2) \tag{3}$$

$$\text{storage}((R,M)) = \sum_{i=1}^{|M|} |M_i| + (4 \cdot |R|) \tag{4}$$

$$\text{q}(D,(R,M)) = \frac{\text{storage}((R,M))}{\text{storage}(D)} \tag{5}$$

In data reduction, we require an objective function that guides the methodology and allows the user to indicate their preference for the trade-off between minimising storage cost and information loss. The objective function used in this paper is the sum of the information lost and the storage ratio, as used in (Guo et al., 2012):

$$\text{h}(D,(R,M)) = \text{e}(D,R) + \alpha \text{q}(D,R) \tag{6}$$

Here, α is a user-defined constant that weights the importance of storage cost against information loss.

4 2D-STR

The 2-Dimensional Spatio-Temporal Reduction algorithm, 2D-STR, is an iterative algorithm that begins with a single rectangular region over the $T \times S$ spatio-temporal space defined by D. A model, of the lowest order, is fitted to the instances in this region using a predefined modelling technique. On each iteration the algorithm decides whether to partition the $T \times S$ space into more non-overlapping rectangular regions, or increase the model complexity of one of the existing regions, thereby aiming to improve its accuracy. The decision made at each step is that which minimises the objective function $h(D,(R,M))$. The function uses the parameter α to weight the importance of storage reduction and model accuracy.

4.1 Data Partitioning

To reduce a dataset, 2D-STR first partitions the dataset into regions of similar instances. Whilst methods exist for partitioning a dataset, such as quadtree and octree (Samet, 1984), the number of new partitions introduced by these methods at each level of decomposition is fixed. Instead, it is more beneficial to use the variance within the data to determine how many partitions are introduced at each level. 2D-STR uses a novel technique, based on the variance within the data, to partition the data.

First, the instances in the dataset D are clustered using hierarchical agglomerative clustering in the feature space (Figure 1(b)). By clustering instances using the F features in the dataset rather than clustering in the $T \times S$ space, 2D-STR finds instances that have similar feature values regardless of when and where the instances were generated. Hierarchical clustering is used as the clusters found in the feature space are not expected to be globular and the resultant hierarchical tree permits fast retrieval as the number of clusters required changes (Fahad et al., 2014).

After clustering, each instance in the $T \times S$ spatio-temporal matrix is labelled with the cluster it has been grouped into. To partition the $T \times S$ spatio-temporal matrix into homogeneous rectangular regions, that is regions containing instances belonging to the same cluster, a Monte Carlo algorithm is used (Figure 1(c)) (Lemley et al., 2016). On each iteration, the regioning algorithm randomly chooses an instance from the matrix which has yet to be placed into a region and uses it as a starting point for a new region. Starting with this instance and whilst ensuring cluster homogeneity, the algorithm iteratively adds instances with the same time value from each preceding sensor along

the road to the region. When an instance belonging to a different cluster is about to be added to the region, the algorithm stops adding preceding sensors. The same action is then repeated for sensors ahead of the starting instance along the road, and the beginning and ending time intervals of the region. This results in rectangular regions of the form depicted in Figure 1(d), and rectangular regions are required to ensure that the range of each region over space and time can be easily defined and stored.

As the number of clusters is increased, the hierarchical clustering decomposes just one branch of the cluster tree. As a result, many clusters are unchanged and retain the same instances. Therefore, the regions defined for these clusters remain the same. Retaining regions between iterations is useful as it allows the models for these regions to persist between iterations.

4.2 Region Modelling

After partitioning D into regions, a technique is required to model the instances within each region. A model M_i is fitted to the instances in each region using the spatial and temporal values of the instances in the region as the independent values of the model, and the values of the real-valued features as dependent values. To maximise the utility of the reduction, we require the ability to reconstruct the instances of the dataset after modelling is performed. However, if the type of analysis to be performed after reduction is already known and reconstruction is not required, more appropriate modelling techniques may be used.

In this paper, we consider two illustrative modelling techniques. First, we consider polynomial linear regression (PLR) because of its ability to explain traffic sensor data well whilst remaining easy to interpret and interpolate. Second, we consider 2-dimensional Discrete Cosine Transforms (DCT) since, like traffic datasets that have been reduced in previous work (Birvinskas et al., 2012), the traffic data we consider is cyclic and can be explained well using cosine waves. Each feature is modelled independently and all features are therefore reconstructable. In general, the modelling technique used should allow for variable model complexities, such as the number of terms used in regression or the number of coefficients stored for discrete cosine modelling.

4.3 Data Reduction

Before the algorithm can be initiated, a value for α must be chosen which weights model accuracy, or error, against the storage ratio in the objective function (Equation 6). 2D-STR is initiated by partitioning the instances, as described in Section 4.1, using a single cluster. Since there is just one cluster, only one region is created, R_1. This region is then modelled using the simplest form of the modelling technique used: in the case of polynomial regression a polynomial model of order 0 (simply a mean) is constructed for each feature; in the case of DCT only the first cosine coefficient is considered. After the model is fitted to the data the result of the objective function $h(D,(R,M))$ is calculated for this initialisation step.

After initialisation the algorithm iterates, at each step deciding whether to increment the number of clusters, and partition the $T \times S$ spatio-temporal space into more regions, or increase the complexity of one of the existing models. When the number of clusters is incremented, one of the existing clusters is split into two new clusters whilst the other clusters remain the same. The models for the regions in the unchanged clusters persist, improving efficiency, whilst new regions are found over the new clusters. Given a current set of regions and their models, (R,M), the steps taken on each iteration are as follows.

1. For each of the regions R_i in R:
   (a) Let M' be a duplicate of M
   (b) Let $M_i \in M$ be the current summary model fitted to instances in r_i and M'_i be a new model fitted to the instances in r_i with one degree of complexity more than M_i
   (c) Replace $M_i \in M'$ with M'_i
   (d) Calculate $h_1 = h(D,(R,M'))$
2. Increase the number of clusters by 1 and then:
   (a) Create the set of regions R' over the $T \times S$ space. Let M'' be a new set of models, where clusters over the $T \times S$ space that remain unchanged retain the same regions and models as in R, and clusters that are split gain new regions with these regions being marked as 'new'
   (b) For each of the regions $R_i \in R'$ marked as 'new':
      i. Fit model M_i to the instances r_i with degree 1
      ii. Add M_i to the set M''
   (c) Calculate $h_2 = h(D,(R',M''))$
3. If step 2 minimised the objective function more than step 1, i.e. $h_1 > h_2$, and $h(D,(R,M)) > h_2$, the reduction (R',M'') is carried forward. Otherwise, if $h_2 > h_1$ and $h(D,(R,M)) > h_1$ then (R,M') is carried forward to the next iteration. If $h(D,(R,M)) > h_1$, $h(D,(R,M)) > h_2$ and $h_1 = h_2$, the algorithm chooses arbitrarily

The algorithm stops when no future step can minimise the objective function further, i.e. $h_1 \geq h(D,(R,M))$ and $h_2 \geq h(D,(R,M))$.

	Apr '17	Sep '17	Nov '17	Dec '17
A30	81.0	79.1	71.1	85.9
A66	89.5	77.8	76.8	86.4
A69	88.1	82.3	82.2	89.8
M11	70.1	64.3	62.7	68.1
M1	83.0	86.5	78.5	80.0
M20	77.4	86.0	78.9	69.3
M23	91.3	89.2	79.9	85.7
M56	90.7	87.7	82.8	93.6

Table 2: Percentage completeness of the datasets used for evaluating the performance of 2D-STR.

5 EXPERIMENTAL METHODOLOGY

To evaluate the performance of 2D-STR, a set of 28 spatio-temporal datasets was used, consisting of month-long surveys of traffic counting sensors in England. The datasets consisted of samples from the A30, A66 and A69 trunk roads as well as the M1, M11, M20, M23 and M56 motorways. These roads were chosen for their differing spatial resolutions and non-uniformly distributed sensors, as well as their differing traffic characteristics. The datasets contained values from sensors located on slip roads and main carriageway, with differing quantities of the two types of road. Furthermore, the distribution of slip roads and main carriageway within the spatial feature space was different for each road. We chose samples from April, September, November and December 2017 to include a range of different traffic trends, i.e. public holidays, and the end of the summer season when many return to school or work from holiday. Each of the datasets consisted of 30 sensors sampled at 15 minute intervals, yielding datasets containing between 54,180 and 83,549 instances. The percentage completeness of the datasets, that is the number of instances compared to the number of sensors and time intervals, $|D|/(|S| \cdot |T|)$, can be seen in Table 2.

Each dataset contained 6 features, each exhibiting different trends across the spatial and temporal dimensions. The features were:

1. Count of vehicles of length 0 – 520 cm
2. Count of vehicles of length 521 – 660 cm
3. Count of vehicles of length 661 – 1160 cm
4. Count of vehicles of length 1160+ cm
5. Total count of all vehicles
6. Average speed of all vehicles

To measure the performance of 2D-STR, we consider NRMSE and storage ratio, as defined in Equations 2 and 5. NRMSE was used as it indicates how well each feature is recreated by the summary modelling. Since it measures the modelling error as a percentage difference over the range of a feature, rather than the feature-instance values themselves, NRMSE was preferred over MAPE. Using NRMSE better reflects the need to recreate each feature across all observed values, rather than allowing errors of smaller feature-instance values to dominate larger ones.

Four variations of 2D-STR were used: polynomial linear regression modelling on each region (PLR-R), polynomial linear regression modelling on each cluster (PLR-C), discrete cosine modelling on each region (DCT-R), discrete cosine modelling on each cluster (DCT-C). Furthermore, 5 values for the parameter α, which weights model accuracy against storage ratio in the objective function (Equation 6) were evaluated, namely $\alpha \in \{0.01, 0.1, 0.5, 1.0, 2.0\}$.

To compare 2D-STR with other reduction methods, we considered IDEALEM (with default parameter settings), DEFLATE (Deutsch, 1996) and PCA. The 2D-STR method is compared with IDEALEM since both permit statistical analysis and modelling to be performed without requiring a further transformation of the dataset. We compared our technique with DEFLATE owing to its use in popular compression algorithms, and PCA owing to its popularity as a feature engineering technique.

6 RESULTS AND DISCUSSION

In this section, we investigate the effect of the α parameter and choice of modelling technique, using the datasets introduced in Section 5, and compare the performance of 2D-STR with other techniques.

6.1 Analysis of Traffic Data

One aim of 2D-STR is to enable analysis of the reduced dataset without requiring further data transformations. In this section, we analyse the partitioning of the datasets introduced in Section 5 by 2D-STR. We suggest that 2D-STR is able to identify structural properties of the datasets.

2D-STR was found to distinguish between instances from daytime and nighttime when two clusters were selected. On motorways, 2D-STR also identified slip roads and placed them into the same cluster as the nighttime instances, regardless of the time that the slip road instance was recorded. As the number of clusters was increased, the daytime cluster was broken into smaller clusters as this cluster contained the highest variance. Successive clusters appeared around

times of high change on the main carriageway sensors, specifically the increase and decrease of total traffic volume around the beginning and ending of the working day. Instances from slip roads remained in the same cluster as nighttime instances.

It was observed that the number of regions was a product of the number of clusters selected by 2D-STR, the number of days in the data sample, and the number of slip roads and their position on the road. Thus, as the number of clusters was increased the number of regions grew proportional to the number of slip roads and number of days in the dataset. In particular, the difference in traffic volume and type between night and day creates regions that separate the nighttime and daytime of each new day in the dataset.

The position of regions in time was indicative of temporal trends in the datasets. Public holidays were observed to omit regions of high traffic volume, indicating unusual traffic volumes during daytime. Periods of differing traffic volumes were identified by the partitioning method and accidents were easily identifiable by sudden changes of high traffic volume regions to low traffic volume regions. Similarly, periods of high congestion or traffic appeared as separate clusters and regions.

6.2 Storage and Error Trade-off, and Choice of Modelling Technique

The NRMSE, storage ratio and number of iterations required to reach the stopping criterion, averaged over the datasets, are shown in Figure 2. Each boxplot shows the interquartile and median error values for NRMSE and storage metrics of the best reduction for each dataset. The whiskers on each boxplot show the minimum and maximum reported of all results.

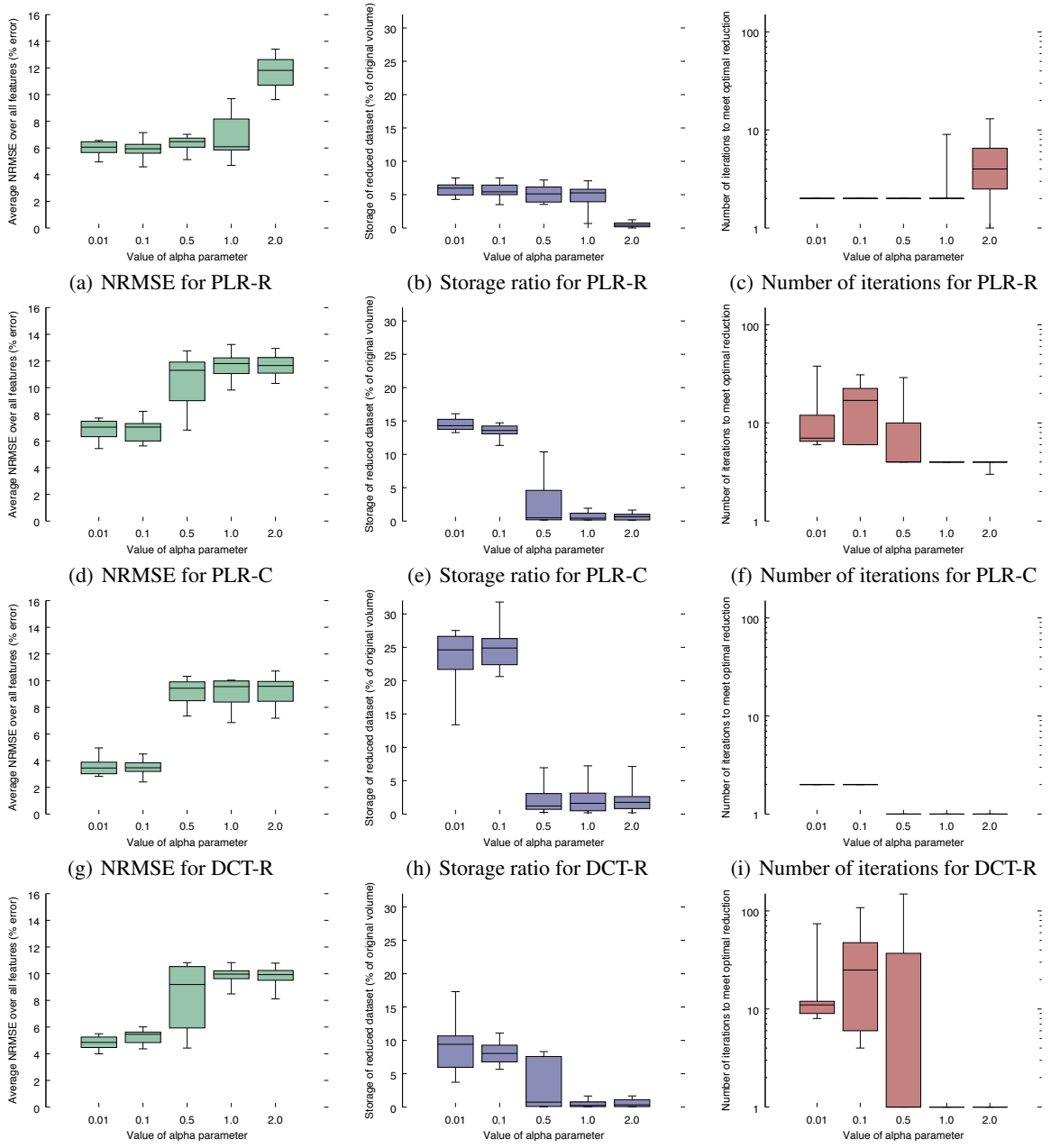
We can draw several conclusions regarding the application of 2D-STR to the datasets considered in this paper. First, more accurate summaries of the dataset occurred when α was small (e.g. $\alpha \in \{0.01, 0.1\}$) and model accuracy was favoured over storage volume. Conversely the opposite was true when α was large (e.g. $\alpha \in \{1, 2\}$). Second, DCT summaries (DCT-R and DCT-C) tended to yield more accurate summaries than polynomial regression (PLR-R and PLR-C) given the same α parameter value. Third, values for α in the middle of the range of those evaluated tended to give less consistent results, as indicated by the larger interquartile range and extreme values shown. Finally, the lower NRMSE values reported for modelling on regions (PLR-R and DCT-R) shows that modelling on regions yielded slightly more accurate representations than modelling on clusters.

As shown in Figure 2, summary modelling on regions (PLR-R and DCT-R) required significantly fewer iterations to reach the stop criterion. Modelling on clusters (PLR-C and DCT-C) required significantly more iterations, and the modelling process took longer owing to the larger number of instances within each model. It was noted that modelling on regions yielded fewer regions for lower α values than modelling on clusters because the latter method results in fewer models which in turn have a higher error when compared to their instances. Polynomial regression modelling (PLR-R and PLR-C) resulted in many models when α was small, each of which was simple, and few models when α was large, all of which were more complex. Discrete cosine modelling (DCT-R and DCT-C) was found to behave similarly when α was small, whilst at larger α values the best iteration was often the first, with one DCT coefficient stored for the entire dataset. For large values of α, neither increasing the number of regions nor increasing the complexity of the existing DCT model gave a lower storage ratio ($q(D,(R,M))$) than the first step. This result may help to explain why such models gave worse NRMSE scores but yielded very low storage values.

Overall, these results show that it is possible to reduce a spatio-temporal traffic sensor dataset significantly whilst incurring tolerable errors. In different scenarios, different choices of α and modelling technique may be desirable. When model accuracy is important, DCT modelling on either regions or clusters (DCT-R or DCT-C) may be preferable. These methods gave between 2.4% and 5.5% NRMSE when α was 0.01 and 0.1. When modelling on clusters (DCT-C) with $\alpha = 0.1$, less than 12% of the original data volume was required to reach these error rates. Whilst higher NRMSE values indicate that the modelling does not fit the instances so well, the smooth regressions output captures the low-frequency trends of the dataset well and removes some of the high-frequency fluctuations in the dataset. This smoothing effect can be used as a way of reducing noise in the dataset.

When few output models are desired, modelling on clusters (PLR-C and DCT-C) is useful. Polynomial regression modelling on clusters (PLR-C) yielded a maximum of 7 models per feature for the entire reduced dataset and DCT modelling (DCT-C) yielded a maximum of 12 models per feature. This may be particularly useful for applications where interpretation of the output models is desired.

A low number of regions can be found by using high α values across all of the modelling techniques evaluated. In particular, DCT modelling on regions (DCT-R) consistently yielded a low number of regions for $\alpha = 0.5, 1.0$ and 2.0 whilst the average NRMSE error was less than 10%. Again, this



(a) NRMSE for PLR-R (b) Storage ratio for PLR-R (c) Number of iterations for PLR-R

(d) NRMSE for PLR-C (e) Storage ratio for PLR-C (f) Number of iterations for PLR-C

(g) NRMSE for DCT-R (h) Storage ratio for DCT-R (i) Number of iterations for DCT-R

(j) NRMSE for DCT-C (k) Storage ratio for DCT-C (l) Number of iterations for DCT-C

Figure 2: Reduction results over traffic sensor datasets. Each sub-figure shows the results from 2D-STR using 5 values for the parameter α, which weights model accuracy against storage ratio. The left column reports NRMSE averaged across the features in the datasets. The middle column reports the quantity of data output by the reduction as a percentage of the original data volume. The right column reports the number of iterations used by the algorithm. Results are shown for polynomial regression on each region (PLR-R), polynomial regression on each cluster (PLR-C), discrete cosine modelling on each region (DCT-R) and discrete cosine modelling on each cluster (DCT-C).

would be particularly useful for interpretability, including fast visualisation of the whole dataset.

Finally, when a fast reduction is required, polynomial and DCT modelling on regions (PLR-R and DCT-R) is useful, particularly when $\alpha = 0.01$ or 0.1. These results showed small NRMSE error rates compared to other techniques and parameter settings evaluated, whilst requiring significantly fewer iterations.

As expected, the α parameter was shown to trade summary model accuracy for volume reduction at a

predictable rate. When $\alpha = 0.5$ we observed higher variance in each of the metrics used, indicating mixed results may be seen for this setting. We suggest that $\alpha = 0.5$ is specific to traffic data and may be different for other types of spatio-temporal data.

6.3 Comparison with Other Techniques

For the datasets considered in this paper, IDEALEM gave a high reduction in the data volume, achieving an average NRMSE of 3.6% for A roads and 4.4% for motorways. This reduction performed poorly in error, however, with an average storage ratio of 47.4% achieved across all features for A roads and 52.1% NRMSE across all features for motorways. These large NRMSE values may be attributed to the diverse temporal patterns exhibited within the files. Since IDEALEM replaces statistically similar temporal blocks with references to the first occurrence of a similar block, it fails to capture the nuances of traffic patterns that vary across different time periods.

A plot of storage ratio against NRMSE for IDEALEM can be seen in Figure 3. Similar NRMSE values and storage ratios were achieved for datasets sampled from the same road regardless of the time period. As seen above, IDEALEM performed better on most A roads compared to motorways, achieving lower NRMSE and storage ratios. Conversely, NRMSE and storage ratios were more varied for samples taken at the same time from different roads. This implies that the spatial nature of the data has a stronger influence on the storage ratio and NRMSE than the temporal nature, when using IDEALEM. Furthermore, it may indicate that, whilst the raw values of instances may change from month to month, the traffic patterns of each road remain the same across the different monthly observations in the datasets.

Results for DEFLATE show similar groupings as IDEALEM. Whilst the NRMSE for DEFLATE is always 0, since DEFLATE is a lossless compression method, A roads tend to result in lower storage ratios than motorways. DEFLATE compressed data from A roads to 17.4% of their original data volume on average, whereas data from motorways were reduced to 18.9%. This higher value for motorways may be indicative of a higher variance in the motorway data. Similar to IDEALEM, DEFLATE also shows that individual roads exhibit more similar compression ratios across different time periods than data taken from different roads during the same time period.

Several observations can be made regarding the results for PCA. First, like DEFLATE and IDEALEM, the temporal window used for sampling has little effect on the resulting NRMSE. Second, the road

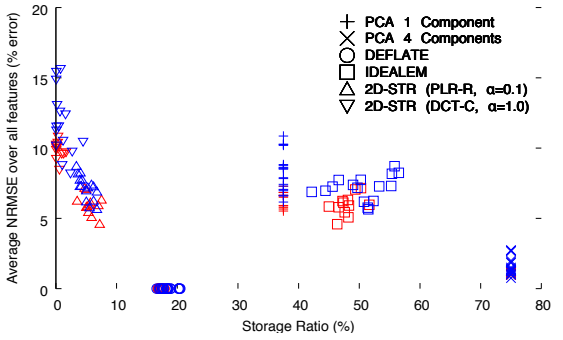
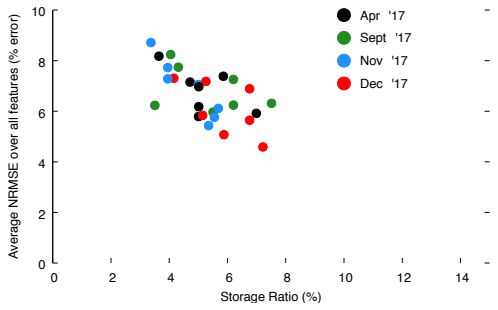


Figure 3: Comparison of storage ratio and NRMSE introduced by IDEALEM, PCA (1 and 4 components), DEFLATE, 2D-STR using polynomial regression on regions (PLR-R) when $\alpha = 1.0$ and 2D-STR using discrete cosine modelling on clusters (DCT-C) when $\alpha = 1.0$. These modelling methods and parameter values were chosen to represent the range of results observed. Results for data sampled from motorways are shown in blue and data sampled from A roads shown in red.

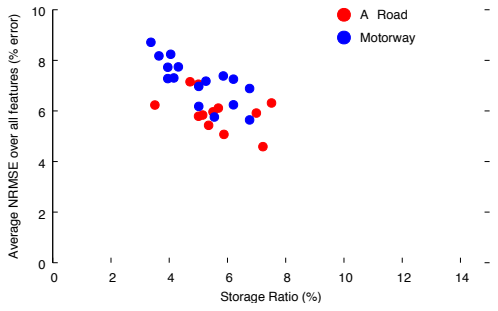
location has some effect on the error rate, with different time periods taken from the same road achieving similar error rates across all numbers of principle components (PCs). Using a single PC, PCA achieved a 6.3% error on A roads and 8.2% on motorways, whilst it achieved 1.2% and 1.8% error respectively when using 4 components. In achieving these error rates, PCA used 37.5% of the original data volume to store a single PC and 75% to store 4 PCs. Finally, as the number of components increased the error rate decreased significantly. When more than 5 components were chosen the error rate was 0 or negligible across all features and datasets. Furthermore, the spread of error rates also decreased as the number of components increased.

Figure 4 shows the NRMSE and storage used by 2D-STR when using polynomial regression on each region (PLR-R) with $\alpha = 0.1$. The class of road is shown to impact the reduction achieved, as observed with the previous techniques. Again, results were clustered more by the road than time period, and similar results were observed for all modelling techniques considered, except polynomial regression on regions when $\alpha = 1.0$ or 2.0. These results indicate that 2D-STR is capable of achieving NRMSE rates similar to those achieved by IDEALEM on 2-dimensional spatio-temporal data. It is capable of achieving similar rates to PCA when a small number of components is chosen, whilst achieving smaller storage ratios than both PCA and IDEALEM.

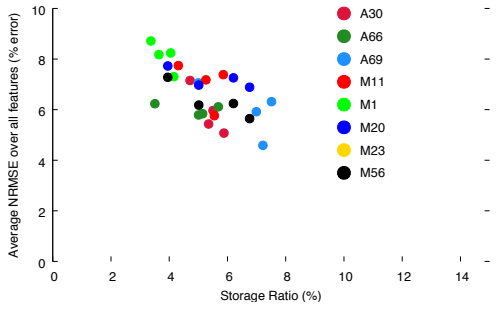
Finally, a comparison of the time taken to reduce the datasets is shown in Table 3. These results indicate that whilst 2D-STR is capable of reducing 2-dimensional spatio-temporal data to a smaller



(a) Coloured by month



(b) Coloured by road class



(c) Coloured by road

Figure 4: Comparison of storage ratio and NRMSE introduced by 2D-STR using polynomial regression on each region (PLR-R) with $\alpha = 0.1$. Here, (a) presents the results of these reductions coloured by the month the dataset was sampled from, (b) presents the results coloured by the type of road, and (c) presents the results coloured by the road the samples were taken from.

data volume than established methods, and achieves a smaller NRMSE in doing so, the computation time is much greater. This suggests that existing methods may be more useful when a faster reduction is required. However, 2D-STR is more effective when longer running times are permitted for reduction. It is important to note that the implementation of 2D-STR may be optimised further, and the running time of the algorithm reduced as a result.

	Min	Avg	Max
2D-STR using PLR-R	<1	545	3,699
2D-STR using PLR-C	14	26	38
2D-STR using DCT-R	18.7	249	1,870
2D-STR using DCT-C	2,542	3,457	3,711
IDEALEM	<1	<1	<1
DEFLATE	<1	<1	<1
PCA	<1	<1	<1

Table 3: Running time (wall time, in minutes) of the algorithms presented in Section 6, over all of the datasets introduced in Section 5. Results are shown for 2D-STR when $\alpha \in \{0.01, 0.1, 0.5, 1.0, 2.0\}$.

7 CONCLUSION

In this paper, we proposed a novel method, 2D-STR, for reducing spatio-temporal traffic datasets by exploiting regions of similar instances. The effectiveness of 2D-STR has been demonstrated, in achieving a 96.3% reduction of the traffic sensor data whilst only introducing a small error ($< 3.7\%$), sufficiently low to make the resulting data useful for many traffic analysis tasks. We demonstrated 2D-STR on medium sized traffic datasets but we believe it could scale to much larger datasets and enable faster processing for queries and analysis. In comparison to other techniques, 2D-STR is found to reduce the dataset to sizes similar to the DEFLATE method whilst achieving NRMSE error rates similar to IDEALEM and PCA. 2D-STR is therefore shown to perform comparably to commonly used techniques whilst enabling a wider range of queries to be answered on the reduced data.

Working with reduced datasets makes analysis possible on modest hardware that would otherwise require greater computational resources. The interpolation nature of the modelling we have adopted in this paper permits many applications, such as spatio-temporal data linkage. Future extensions of this work could investigate reduction over the entire road network, larger datasets, and other modelling techniques. Furthermore, extensions of the technique for other types of data should be explored.

ACKNOWLEDGEMENTS

The lead author gratefully acknowledges funding by the UK Engineering and Physical Sciences Research Council (grant no. EP/L016400/1), the EPSRC Centre for Doctoral Training in Urban Science. The lead author also gratefully acknowledges funding by TRL.

REFERENCES

Acampora, G., Tortora, G., and Vitiello, A. (2016). Comparison of Multi-objective Evolutionary Algorithms for prototype selection in nearest neighbor classification. In *SSCI 2016*, pages 1–8.

Berberidis, D. and Giannakis, G. B. (2015). Data sketching for tracking large-scale dynamical processes. In *ACSSC 2015*, pages 345–349.

Berberidis, D. and Giannakis, G. B. (2017). Data Sketching for Large-Scale Kalman Filtering. *IEEE Transactions on Signal Processing*, 65(14):3688–3701.

Birvinskas, D., Jusas, V., Martisius, I., and Damasevicius, R. (2012). EEG Dataset Reduction and Feature Extraction Using Discrete Cosine Transform. In *2012 Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation*, pages 199–204.

Bloom, B. H. (1970). Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426.

Cormode, G., Garofalakis, M., Haas, P. J., and Jermaine, C. (2012). Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294.

Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

Deutsch, P. (1996). Deflate compressed data format specification version 1.3. Technical report.

Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S., and Bouras, A. (2014). A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279.

Flajolet, P., Fusy, É., Gandouet, O., and Meunier, F. (2007). Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156.

Garcia, S., Derrac, J., Cano, J., and Herrera, F. (2012). Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study. *IEEE Transactions on pattern analysis and machine intelligence*, 34(3):417–435.

Guo, T., Yan, Z., and Aberer, K. (2012). An Adaptive Approach for Online Segmentation of Multi-dimensional Mobile Data. In *Proceedings of the Eleventh ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 7–14. ACM.

Lakshminarasimhan, S., Jenkins, J., Arkatkar, I., Gong, Z., Kolla, H., Ku, S. H., Ethier, S., Chen, J., Chang, C. S., Klasky, S., Latham, R., Ross, R., and Samatova, N. F. (2011). ISABELA-QA: Query-driven analytics with ISABELA-compressed extreme-scale scientific data. In *SC 2011*, pages 1–11.

Lemley, J., Jagodzinski, F., and Andonie, R. (2016). Big Holes in Big Data: A Monte Carlo Algorithm for Detecting Large Hyper-Rectangles in High Dimensional Data. In *COMPSAC 2016*, pages 563–571.

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. (2017). Feature Selection: A Data Perspective. *ACM Comput. Surv.*, 50(6):94:1–94:45.

Martinez, A. M. and Kak, A. C. (2001). PCA versus LDA. *IEEE Transactions on pattern analysis and machine intelligence*, 23(2):228–233.

Mukahar, N. and Rosdi, B. A. (2018). Performance comparison of prototype selection based on edition search for nearest neighbor classification. In *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, ICSCA 2018, pages 143–146. ACM.

Ougiaroglou, S. and Evangelidis, G. (2012). Efficient Dataset Size Reduction by Finding Homogeneous Clusters. In *Proceedings of the Fifth Balkan Conference in Informatics*, pages 168–173. ACM.

Pan, B., Demiryurek, U., Banaei-Kashani, F., and Shahabi, C. (2010). Spatiotemporal Summarization of Traffic Data Streams. In *Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming*, pages 4–10. ACM.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260.

Sisovic, S., Bakaric, M. B., and Matetic, M. (2018). Reducing Data Stream Complexity by Applying Count-Min Algorithm and Discretization Procedure. In *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 221–228.

Tai, K. S., Sharan, V., Bailis, P., and Valiant, G. (2018). Sketching Linear Classifiers over Data Streams. In *Proceedings of the 2018 International Conference on Management of Data*, pages 757–772. ACM.

Tenenbaum, J. B., De Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323.

Wang, M., Li, H. X., and Shen, W. (2016). Deep autoencoder in model reduction of lage-scale spatiotemporal dynamics. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3180–3186.

Wong, L. Z., Chen, H., Lin, S., and Chen, D. C. (2014). Imputing Missing Values in Sensor Networks Using Sparse Data Representations. In *Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 227–230. ACM.

Wu, K., Lee, D., Sim, A., and Choi, J. (2017). Statistical data reduction for streaming data. In *2017 New York Scientific Data Summit (NYSDS)*, pages 1–6.

Xue, B., Zhang, M., Browne, W. N., and Yao, X. (2016). A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626.