*k*D-STR: A Method for Spatio-Temporal Data Reduction and Modelling

LIAM STEADMAN, NATHAN GRIFFITHS, and STEPHEN JARVIS, The University of Warwick, United Kingdom
MARK BELL, SHAUN HELMAN, and CAROLINE WALLBANK, TRL, United Kingdom

Analysing and learning from spatio-temporal datasets is an important process in many domains, including transportation, healthcare and meteorology. In particular, data collected by sensors in the environment allows us to understand and model the processes acting within the environment. Recently, the volume of spatio-temporal data collected has increased significantly, presenting several challenges for data scientists. Methods are therefore needed to reduce the quantity of data that needs to be processed in order to analyse and learn from spatio-temporal datasets. In this paper, we present the k-Dimensional Spatio-Temporal Reduction method (kD-STR) for reducing the quantity of data used to store a dataset whilst enabling multiple types of analysis on the reduced dataset. kD-STR uses hierarchical partitioning to find spatio-temporal regions of similar instances, and models the instances within each region to summarise the dataset. We demonstrate the generality of kD-STR with 3 datasets exhibiting different spatio-temporal characteristics and present results for a range of data modelling techniques. Finally, we compare kD-STR with other techniques for reducing the volume of spatio-temporal data. Our results demonstrate that kD-STR is effective in reducing spatio-temporal data and generalises to datasets that exhibit different properties.

CCS Concepts: • **Information systems → Spatial-temporal systems**; Decision support systems; Data mining.

Additional Key Words and Phrases: spatio-temporal data, data reduction, partitioning, modelling

ACM Reference Format:
Liam Steadman, Nathan Griffiths, Stephen Jarvis, Mark Bell, Shaun Helman, and Caroline Wallbank. 2020. *k*D-STR: A Method for Spatio-Temporal Data Reduction and Modelling. *ACM/IMS Trans. Data Sci.* 1, 1, Article 1 (January 2020), 32 pages. https://doi.org/10.1145/3439334

1 INTRODUCTION

Spatio-temporal data generated by sensors in an environment is widely used in many domains, and analysing or learning from such data allows us to understand processes in the environment. In spatio-temporal datasets, correlations between nearby sensors and time intervals can be exploited to better model or predict trends [45]. For example, in the transportation domain, Lv *et al.* used spatial and temporal correlations to create a traffic flow prediction model using a deep learning architecture [29]. Common tasks in analysing or learning from spatio-temporal data include: (i) imputing missing instances at locations or times not sampled; (ii) identifying unusual behaviours, such as sensors that

Authors' addresses: Liam Steadman, L.Steadman@warwick.ac.uk; Nathan Griffiths, Nathan.Griffiths@warwick.ac.uk; Stephen Jarvis, S.A.Jarvis@warwick.ac.uk, The University of Warwick, Coventry, Warwickshire, United Kingdom, CV4 7AL; Mark Bell, MBell@trl.co.uk; Shaun Helman, SHelman@trl.co.uk; Caroline Wallbank, CWallbank@trl.co.uk, TRL, Crowthorne House, Nine Mile Ride, Wokingham, Berkshire, United Kingdom, RG40 3GA.

2577-3224/2020/1-ART1
https://doi.org/10.1145/3439334

perform unexpectedly or time periods wherein instances do not fit expected trends; (iii) calculating summary statistics over features or calculating the variance from the expected trend within a time period; (iv) comparing time periods or sensors, for example performing a month-on-month time series analysis; and (v) predicting future instances. These tasks are all aided by the correlations present in spatio-temporal data.

In recent years, the volume of spatio-temporal datasets has increased significantly, presenting several challenges for data scientists. Such increases in data volume require more computational time and memory to process and analyse the data. Often this task can become infeasible, and so methods are required to reduce the volume of data to be processed whilst minimising the error introduced in later analysis or modelling. The aim is not necessarily to compress the data, since many analysis tasks cannot be achieved on compressed data. Rather, the aim is to summarise the data in a way that allows analysis to be performed directly on the summarised data.

In previous work [43], we introduced the 2-Dimensional Spatio-Temporal Reduction (2D-STR) method for reducing the quantity of data used when storing a spatio-temporal dataset with a single spatial dimension, such as a link-based representation for transport infrastructure. 2D-STR partitions the spatial and temporal domains into *regions*, and models the instances within each region to summarise the data. Beginning with a single region, 2D-STR trades improved reconstruction accuracy and increased storage overhead to meet the user's desired level of reduction. This is achieved by either increasing the number of regions, or increasing the accuracy of the model in one of the regions of the reduced dataset. When 2D-STR's heuristic is minimised, the set of regions and models is output, and the data can be reconstructed using these models.

However, 2D-STR cannot be applied directly to datasets containing two or more spatial dimensions since it is ambiguous how sensors and time intervals should be grouped into partitions. Therefore, this paper makes three main contributions: (i) we introduce kD-STR, which provides a solution for this issue and generalises the reduction process for k spatio-temporal dimensions; (ii) we demonstrate the generality of kD-STR for datasets exhibiting different spatio-temporal characteristics, and analyse the effect of increasing data sizes and different spatial referencing systems; (iii) we provide an analysis of the time and memory complexity of kD-STR.

Whilst other methods for reducing spatio-temporal datasets exist, many remove information that may be of importance in later processing. Existing model-based reduction techniques overcome this issue, however they ignore the variability and autocorrelation between instances in space and time when partitioning the data. Methods such as ISABELA require entire partitions of instances be reconstructed and processed before any raw data can be retrieved from the reduced dataset [24]. Furthermore, methods such as IDEALEM replace entire partitions with links to other partitions [25, 52]. In contrast, kD-STR uses the variability of instances in time and space to partition the dataset. By using the locations and times of instances as predictor features in the modelling process, kD-STR allows for faster reconstruction of raw data from the reduced form, overcoming the issue presented by ISABELA. Compared to IDEALEM and other methods, kD-STR stores information about all instances in the data. The properties of kD-STR therefore provide multiple improvements over the state of the art.

The remainder of this paper is structured as follows. Section 2 reviews existing methods for reducing the volume of spatio-temporal datasets. Section 3 formalises the problem of reducing a dataset in a manner that enables analysis to be performed on the reduced data, and introduces the notation used in the paper. Section 4 describes the proposed approach, kD-STR. Sections 5 and 6 evaluate the effectiveness of kD-STR on three sources of data exhibiting different spatio-temporal characteristics, and compare its performance with existing reduction techniques. Finally, Section 7 discusses the impact of these results, and Section 8 concludes the paper and gives future directions of this work.

2 RELATED WORK

The quantity of data present in many spatio-temporal datasets makes them difficult or infeasible to process in their raw forms. To facilitate faster modelling and analysis, several techniques exist for reducing the quantity of data that needs to be processed. Such techniques aim to minimise the difference between analysis performed on, or models created from, the original dataset and the reduced dataset. Other techniques result in smaller reduced datasets, yet the reduced data output is only useful for answering specific questions. In this section, we summarise the existing methods for reducing the quantity of data to be processed in a spatio-temporal dataset.

2.1 Feature Selection and Extraction Techniques

Many existing methods for reducing datasets focus on removing a subset of features, thereby reducing the quantity of data stored for each instance. Feature selection methods, which choose a subset of features from the original dataset, can be separated into three categories. First, filter methods rank features according to a relevance criterion, such as Shannon entropy, and remove any features that score below a defined threshold. However, choosing such a threshold is often dataset specific and can be time consuming. Second, wrapper methods use search algorithms to find the optimal subset of features according to an objective function [8]. Third, embedded methods aim to incorporate feature selection as part of the training process of machine learning. Embedded methods evaluate different fixed-size subsets of features to find the subset that consistently yields the smallest classification error.

Several feature selection techniques for real-valued data have been evaluated and compared in the context of different domains. Meskina compared the results of building machine learning models on full datasets versus the same datasets after they had been reduced by the FOCUS [2] and RELIEF [22] filtering methods [31]. FOCUS iteratively searches increasingly larger combinations of features until a combination is found that accurately separates two target classes, whilst RELIEF ranks features using a statistical relevance measure and selects the fewest ranked features that are sufficient to separate classes. Meskina found the outputs of both methods achieved similar accuracy rates for Support Vector Machine, Naïve Bayes and k-nearest neighbour classification compared to using the full dataset. However, the time required for classification was remarkably lower [31]. In a similar evaluation, Christopher and Balamurugan found correlation based feature selection achieves 97.8% of the accuracy achieved with the entire dataset using 12 of the original 181 features [11]. Several other feature selection techniques for real-valued data exist, and a number of reviews of these can be found in the literature [8, 26, 27, 47, 53].

In contrast to feature selection, extraction (or engineering) methods project the original features onto a new feature space, often of a differing dimensionality. The best mapping is that which optimises an objective criterion, such as explained variance or accuracy, when combined with modelling. Linear feature extraction algorithms include Principal Components Analysis (PCA) [37] and Linear Discriminant Analysis (LDA) [30]. Whilst PCA maximises the inter-feature variance for created features, LDA minimises the variance within a class and maximises the variance between classes. Assumptions made by PCA often require adaptations for spatio-temporal data, and these have been discussed in literature [15]. Non-linear algorithms, sometimes referred to as manifold learning, map high-dimensional datasets to lower dimensions such that the mapping reflects the structural features of the dataset. Examples such as Isomap, which uses a geodesic distance measure between instances, and Locally Linear Embedding (LLE) [39], which improves on Isomap by reducing the computation required, have been reviewed and compared in literature [28].

Whilst effective at reducing the volume of a dataset, feature selection and extraction techniques do not take advantage of the correlations and patterns present in spatio-temporal data. Furthermore, they may remove features that are significant for subsequent analysis or fail to capture nuances that may be of interest in later processing [21]. Removing features requires the user to have knowledge of the analysis they are to perform ahead of time. Instead, it may be more beneficial to retain information about all features. Therefore, whilst feature selection and extraction techniques are widely used, methods that take advantage of the correlations present in spatio-temporal data and retain information about all features may be more useful.

2.2 Instance Selection and Abstraction Techniques

Instance selection techniques choose a subset of instances from the original dataset that are sufficient for later processing tasks. For example, in a classification task they choose and retain only those instances that are sufficient to accurately classify unseen data. Several important and established algorithms currently exist for this task, such as the IB3 incremental algorithm [1]. In IB3, a set of output (i.e., selected) instances $\mathcal{S}$ is initialised to contain a single instance chosen from the set of input instances. Then, the remaining instances in the input set are considered in turn. If an instance being considered, x, and its nearest neighbour in $\mathcal{S}$ have different class labels, x is added to $\mathcal{S}$. However, if the nearest neighbour to x in $\mathcal{S}$ has the same class label, x is disregarded and not added to $\mathcal{S}$. After all input instances are considered, $\mathcal{S}$ contains the instances that are sufficient for classifying instances similar to the input instances. In the spatio-temporal domain, Whelan *et al.* have used the k-medoids clustering technique to reduce a dataset to k instances [49, 50].

Like feature extraction, instance abstraction techniques create a smaller set of new *prototype* instances which represent the original instances but do not necessarily exist in the original dataset. Prototyping has been shown to create reduced training sets for tasks such as k-nearest neighbour classification, and training models on these reduced datasets is demonstrably faster with minimal effects on classification accuracy [35]. A simple example, the Prototypes for Nearest Neighbour (PNN) algorithm, is a supervised method which iteratively creates weighted prototypes that are expected to achieve approximately the same classification accuracies as the original instances [9]. Another example, the Decision Surface Mapping (DSM) algorithm, selects instances randomly from the set of original instances to become prototypes [20]. Each of the instances in the original dataset are then considered and classified according to the prototypes. When an instance is incorrectly classified, the DSM algorithm rewards the nearest neighbour of the correct class by moving it closer to the considered instance whilst the nearest overall neighbour is moved further away. The Learning Vector Quantization (LVQ) family of algorithms operate in a similar fashion to the k-means algorithm [23]. Instead of updating prototypes only when a misclassification is made, the LVQ algorithm updates prototypes even when an instance is correctly classified. Comparisons of instance abstraction techniques can be found in the literature [46].

Like feature selection methods, both instance selection and abstraction techniques remove instances from the dataset that may be of significance in later processing. For example, removing instances may increase inaccuracies in imputation tasks or in the calculation of statistics. Furthermore, querying the removed instances may no longer be possible. Therefore, techniques such as these restrict the analysis that can be performed on the reduced data they output.

2.3 Data Sketching

The techniques discussed until now focus on removing or prototyping features and instances. In contrast, data sketching techniques create query-specific summaries of the data using a fixed number of passes over the data. Many sketching techniques focus on counting items, such as the Count-Min sketch and its adaptation for real-valued data [13, 42]. Another, the Bloom Filter, answers membership questions using hash tables [6]. The HyperLogLog (HLL) algorithm uses a probabilistic counter to answer cardinality questions and is sufficiently efficient to be used with very large quantities of data [19]. However, these 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 [4]. Furthermore, Tai *et al.* presented a sketching method for building linear classifiers over a spatio-temporal dataset [44]. By building linear classifiers over the temporal stream of each sensor, correlated features can be identified while also permitting analysis of the stream's instances. However, this methodology destroys features which are not heavily weighted by the linear classifier.

Sketching techniques are limited in the analyses or later modelling they permit [12]. They are created for specific queries and, since the original dataset is destroyed after the sketch is created, it is not possible to reconstruct the data for other analyses. Most do not take advantage of the correlations present in spatio-temporal data, and many require knowledge of which features will be of interest before the sketch is created. This may not be known ahead of time.

2.4 Data Reduction using Modelling

Whilst the techniques discussed above result in the loss of instances or features, some techniques exist for reducing a dataset using statistical modelling. The IDEALEM algorithm partitions a data stream into blocks of a fixed size [25, 52]. Key statistical properties about these blocks, such as min, max and average values, are then used to identify those blocks which are statistically similar. For each set of similar blocks, the raw data of one block is retained along with statistics about the block and where it repeats in the data stream. By processing each of its prototype blocks, IDEALEM allows us to identify unusual temporal periods that do not fit expected trends. It also enables comparison of different sensors or time periods, retains information about all features, and allows for the faster generation of statistics compared to the original data stream. However, replacing blocks with links to a similar block introduces error. Furthermore, since the method does not consider the spatial nature of spatio-temporal data, IDEALEM does not permit spatial imputation without further modelling.

Similar to IDEALEM, the ISABELA algorithm partitions each feature into fixed size spatial windows and sorts the observed values into ascending order within each window [24]. A B-spline curve is then fitted to each window and the parameters of the curve stored using temporal encoding. Since the instances in each window are stored in ascending order by value, a mapping back to the temporal ordering also needs to be stored. This is not necessary with many real-world sensor datasets, such as traffic data, which are more smooth and cyclic than the data motivating ISABELA[1]. Furthermore, the reduced representation given by ISABELA makes imputation of values impossible without mapping the data back into temporal order. Statistics can be generated over the data if the temporal period of interest is exactly covered by a window, otherwise mapping back into temporal order is again required. In the same way, identifying unusual spatial or temporal regions is partially supported.

Deep autoencoders have also been used to model the temporal features of spatio-temporal datasets [48]. The Sparse Autoencoder (SAE) has been used to reliably estimate missing data in spatio-temporal sensor datasets [51]. This fitting of a summary, which minimises the root mean square error (RMSE) over instances in the discrete spatial and temporal dimensions, is able to impute missing values given other instances from the same time interval. It may be possible to adapt this approach to incorporate multiple time intervals, e.g., the whole dataset, and store the autoencoder weights for the purposes of reproducing the dataset. However, autoencoder weights are incomprehensible in analysis and so prevent manual analysis of the reduced dataset.

[1]In other work, discrete cosine transforms have been shown to effectively reduce similar datasets [5].

In the domain of traffic dataset analysis, Pan *et al.* have proposed a two-part algorithm that summarises a spatio-temporal traffic sensor dataset [36]. Their method creates a spatio-temporal signature of the dataset using a technique such as wavelet decomposition, and a set of outliers that fall outside an acceptable error margin of this signature. Whilst this technique is good at capturing the cyclic and seasonal natures of some datasets, it performs poorly on datasets containing irregular patterns or many outliers. For example, in road traffic data, instances from national holidays (temporal domain) and areas of construction work (spatial domain) are known to deviate from regular traffic cycles, and so will be labelled as outliers. The algorithm will only retain some of these outlier instances due to its probabilistic nature. Therefore, reconstruction of these spatial and temporal regions may be poor.

2.5 Discussion

Reduction techniques that partition and model the data offer a clear advantage over selection and prototyping/extraction techniques, as well as data sketching techniques. However, IDEALEM [25, 52] and the algorithm proposed by Pan *et al.* [36] remove entire partitions of instances, removing information that may be significant in later processing. The ISABELA method overcomes this issue by storing a model for each partition, yet retrieving the data requires many instances be reconstructed and re-organised back into the spatial and temporal domains [24]. This adds significant computational overhead for data retrieval. Furthermore, existing methods use fixed-size partitioning methods that require the user to select an appropriate partition size prior to reduction, ignoring the spatio-temporal dependencies in the data. This can reduce the storage efficiency of the reduced data and require more instances or coefficients be stored to capture the information present in the data.

Therefore, we argue that a new method is required that overcomes these issues. There are four requirements for this method. First, it must use the variability of the data in space and time to partition the instances. Second, the method must capture information about all instances and features. Third, it must allow data to be retrieved without requiring multiple instances be imputed and processed. Finally, the method must permit some analysis using the reduced data. All of these requirements are met by the *k*D-STR algorithm presented in this paper.

3 SPATIO-TEMPORAL DATA REDUCTION

To capture information that may be needed for later analysis, many datasets sample spatial and temporal processes at high frequency and resolution. This often results in datasets that contain regions of low variability where the process being sampled exhibits little change. For example, many weather datasets are sampled at 15 minute intervals, yet the weather they report can exhibit low change for several hours at a time. Therefore, such spatio-temporal datasets are very large but contain high spatial and temporal autocorrelation.

To decrease the quantity of data to be analysed or modelled in these cases, we can group similar adjacent instances together to form spatio-temporal regions. Each of these regions can then be reduced to a model of the instances within the region. This reduces both the number of instances in the reduced dataset, since each region becomes a single model in the reduced dataset, and the quantity of data used to store the information within each region. Whilst IDEALEM [25, 52], ISABELA [24], and the method presented by Pan *et al.* [36] take a similar approach, they use either single or fixed-sized regions defined over the spatial and temporal domains. Instead, by forming regions where the dataset exhibits little change, we argue that less complex models can be used to accurately represent the data. The nuances of the original instances can be maintained, and answering queries on the reduced dataset is still supported. In this section, we formalise this approach and the notation used in this paper.

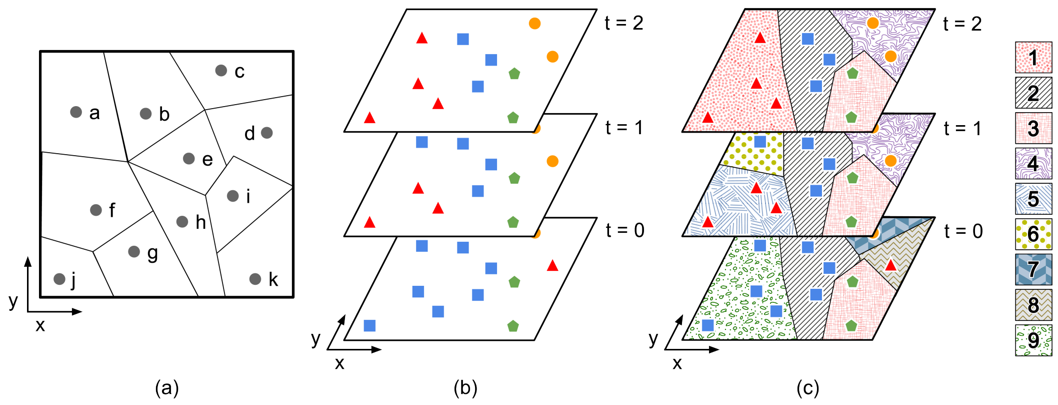


Fig. 1. (a) Sensors in a spatio-temporal dataset are shown at their spatial locations with the space decomposed into Voronoi polygons [3]. (b) Instances recorded at these sensors, coloured by their similarity. (c) The instances have been grouped into spatio-temporal regions, where each region is defined by the union of the Voronoi polygons of its constituent sensors, and a starting and ending time interval.

A spatio-temporal dataset D is a set of instances generated by a set of synchronous or asynchronous sensors. Each instance in D is recorded at a location $s \in S^{\mathcal{D}}$ at time $t \in T$. Therefore, we can reference each instance in D using the notation $d_{t,s}$. Here, T is the continuous 1-dimensional temporal domain and $S^{\mathcal{D}}$ is the continuous $\mathcal{D}$-dimensional spatial domain, thus the spatio-temporal space is k dimensional, where $k = 1 + \mathcal{D}$. In this work, we assume that each sensor exists at a unique location and records at most one instance at any given time, thus only one instance can exist at a given time and location (t, s)[2].

Each instance $d_{t,s}$ is a vector of values over the set of features F, i.e., $d_{t,s} = \langle d^1_{t,s}, ..., d^{|F|}_{t,s} \rangle$. For example, in a weather dataset these features may be rainfall, temperature and humidity. For generality we assume that these features are real-valued. Therefore, the dataset is a mapping from the k-dimensional spatio-temporal space to the $|F|$-dimensional feature space, $D : T \times S^{\mathcal{D}} \rightarrow \mathbb{R}^{|F|}$. Techniques exist for representing binary and categorical features as real-valued data, and appropriate partitioning algorithms can be used for binary and categorical features.

To reduce the dataset D, we wish to find the set $R = \{r_1, ..., r_{|R|}\}$ of non-overlapping regions in the $T \times S^{\mathcal{D}}$ space. Each region $r_i \in R$ is defined by a bounding spatial polygon P_i in the $S^{\mathcal{D}}$ space, a beginning time t_b and ending time t_e, and so may span multiple sensors and time intervals. The subset of instances in D that exist within the spatial and temporal bounds of region r_i is denoted by D_i, where $D_i = \{d_{t,s} \in D \mid inside(s, P_i),\ t_b \leq t \leq t_e\}$ and $\langle P_i, t_b, t_e \rangle = r_i$. Here, the function $inside(s, P_i)$, is used to indicate that location s is within the bounding polygon P_i of region r_i.

The process of partitioning the $T \times S^{\mathcal{D}}$ space into regions, where $\mathcal{D} = 2$, is shown in Figure 1. In (a), sensors from a dataset are shown at their locations in the 2-dimensional spatial domain, which has been split into Voronoi polygons surrounding each sensor. In (b), instances recorded at the sensors have been colour coded by their similarity. For example, instances denoted by blue squares are more similar to each other than to instances denoted by red triangles, yellow circles and green pentagons. Finally, in (c), the instances from (b) have been grouped into 9 regions. Each region is

[2]In some scenarios a dataset may contain multiple instances recorded at the same location and time. For example, multiple sensors may be located at the same location and record instances at the same time. In these scenarios, multiple models may be output by the reduction process, thus allowing multiple response values to be modelled for the same location and time input. This is discussed further in Section 8.

Table 1. Notation used in this paper

Symbol	Definition
D	Original dataset containing instances in the spatio-temporal space $T \times S^{\mathcal{D}}$, with the number of instances in D denoted $\lvert D \rvert$
F	The set of real-valued features in D, excluding the referencing features T and $S^{\mathcal{D}}$
$d_{t,s}$	An individual instance in D
$d_{t,s}^f$	Value of $d_{t,s}$ for feature $f \in F$
R	A set of non-overlapping spatio-temporal regions in the $T \times S^{\mathcal{D}}$ space
r_i	An individual spatio-temporal region in R
P_i	The set of $\mathcal{D}$-dimensional coordinates defining the bounding spatial polygon of region r_i
D_i	Set of instances of dataset D contained in region r_i
M	Set of summary models of the instances in dataset D
m_j	Individual summary model, fitted over the a subset of instances in D, with the number of coefficients used to store m_j denoted $\lvert m_j \rvert$
$\langle R, M \rangle$	A reduction of dataset D
$e(D, \langle R, M \rangle)$	Error introduced after D is reduced to regions R and models M
$q(D, \langle R, M \rangle)$	Ratio of storage required for regions R and summary models M compared to the original dataset D
$h(D, \langle R, M \rangle)$	Objective function used to find the best reduction given parameter α, the constant that prioritises between $e(D, \langle R, M \rangle)$ and $q(D, \langle R, M \rangle)$

defined by a spatial polygon and start and end time. For example, region 4 exists in the top-right of the space, from time 1 to time 2. This partitioning method is applicable to any number of spatial dimensions, however we focus on 2 spatial dimensions for simplicity.

Each region r_i is associated with a model m_j fitted to D_i, and we refer to the set of models as M. Furthermore, we denote $\lvert m_j \rvert$ to be the number of coefficients used to store model m_j. In *k*D-STR, we assume that a single modelling technique is used, thereby removing the need to store which modelling technique was used for each region. We may form a model for each region, i.e., $\lvert R \rvert = \lvert M \rvert$, or associate a subset of regions with the same model, i.e., $\lvert R \rvert < \lvert M \rvert$. Finally, the term *reduction* is used to refer to the set of regions and models, denoted by the tuple $\langle R, M \rangle$. The notation used in this paper is summarised in Table 1.

When reducing D to $\langle R, M \rangle$ we wish to minimise the information lost. One way of measuring information loss is to reconstruct the dataset from $\langle R, M \rangle$ as D' and measure the difference between each instance $d_{t,s} \in D$ and the corresponding instance $d'_{t,s} \in D'$. A simple example is the mean absolute percentage error (MAPE) averaged across the dataset:

$$e_{\mathrm{MAPE}}(D, D') = \frac{1}{\lvert D \rvert \cdot \lvert F \rvert} \sum_{d_{t,s} \in D} \sum_{f \in F} \left| \frac{d_{t,s}^f - d'^f_{t,s}}{d_{t,s}^f} \right| \tag{1}$$

However, MAPE is undefined for instances containing feature values of 0, such as in rainfall datasets. Therefore, it is unsuitable as a measurement of error in the reduction of such datasets. An alternative measure is the normalised root mean square error (NRMSE) averaged across the dataset:

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

where,

$$\psi(f, D, D') = \sqrt{\frac{\sum_{d_{t,s} \in D} (d^f_{t,s} - d'^f_{t,s})^2}{|D|}} \tag{3}$$

and $range(f) = max_{t,s}(d^f_{t,s}) - min_{t,s}(d^f_{t,s})$. In general, we refer to the error introduced by reducing D to $\langle R, M \rangle$ as $e(D, \langle R, M \rangle)$. When using a difference metric between D and the reconstructed D', we say $e(D, \langle R, M \rangle) = e(D, D')$.

Whilst minimising the error introduced in reduction, we also wish to minimise the storage cost of the reduced dataset. To store an instance in the original dataset, a location point for each of the $\mathcal{D}$ spatial dimensions, a time in the temporal dimension, and a value for each of the F real-valued features must be stored. The storage cost of the original dataset is then the cost per instance multiplied by the number of instances:

$$storage(D) = |D| \cdot (|F| + k) \tag{4}$$

In the case of the reduced dataset, the storage required for each region is a start and end time in the temporal dimension, the set P_i of points that define the bounding polygon of the region in the spatial domain, and the set m_j of coefficients required to store the model of the region. Each point in the set P_i requires a value for each of the $\mathcal{D}$ spatial dimensions to be stored. Therefore, the storage cost of the reduced dataset is the sum of region costs as shown in Equation 5, where $(|P_i| \cdot (k-1))$ is the number of values stored for the spatial boundary of a region r_i, and 2 values are used to store the beginning and end times of r_i. Furthermore, $|m_j|$ is the number of coefficients used to store the model m_j.

$$storage(\langle R, M \rangle) = \sum_{i=1}^{|R|} \big((|P_i| \cdot (k-1)) + 2\big) + \sum_{j=1}^{|M|} |m_j| \tag{5}$$

We use the ratio of the storage cost of the reduced and original datasets to define the storage ratio, as shown in Equation 6.

$$q(D, \langle R, M \rangle) = \frac{storage(\langle R, M \rangle)}{storage(D)} \tag{6}$$

Finally, in reducing a dataset we wish to balance the reduction in storage against the error introduced. The best balance is subjective and depends on the dataset being reduced as well as the kinds of analysis to be performed after reduction. Given the user's preference, we can use the objective function $h(D, \langle R, M \rangle)$ defined in Equation 7 to find the optimal reduction. Here, the parameter α is used to indicate the user's preference for the balance between reduction in storage and minimising the error introduced.

$$h(D, \langle R, M \rangle) = \alpha \cdot q(D, \langle R, M \rangle) \ + \ (1 - \alpha) \cdot e(D, \langle R, M \rangle) \tag{7}$$

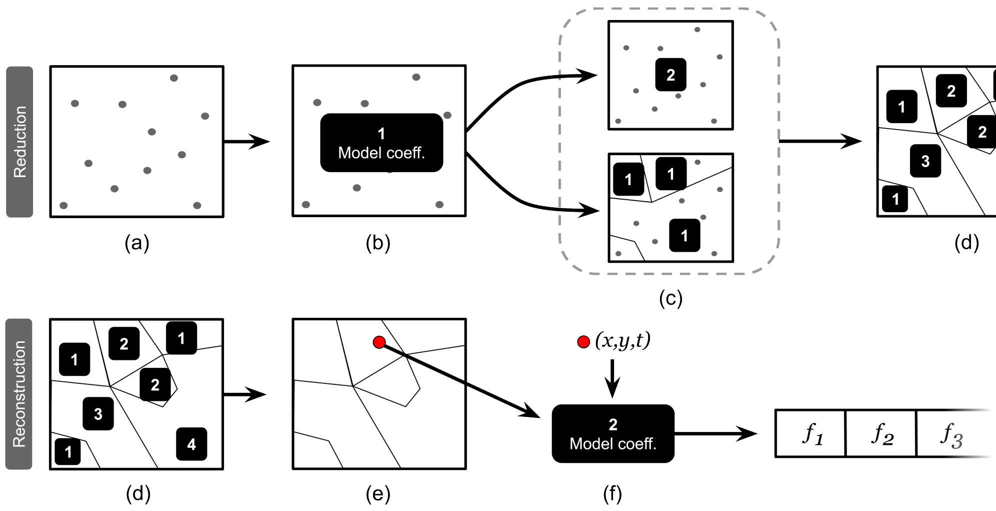


Fig. 2. Overview of kD-STR data reduction and reconstruction. (a) The input dataset. (b) Spatial and temporal domains partitioned into 1 region, and 1 model coefficient stored. (c) Number of model coefficients and number of regions repeatedly increased and tested until the objective function is minimised. (d) Regions and model coefficients output. (e) Reconstructing the feature values at a location and time – the containing region is selected and its model retrieved. (f) Location and time input into model and feature values reconstructed.

4 *k*D-STR: *k*-DIMENSIONAL SPATIO-TEMPORAL REDUCTION

In this paper, we introduce the k-Dimensional Spatio-Temporal Reduction method (kD-STR) for achieving the goal of reduction described in Section 3. As input, the method takes the dataset to be reduced, the parameter α, which indicates the user's preference for reduction in storage volume versus introduced error, and a preferred modelling technique. kD-STR is an iterative algorithm that begins by forming a partitioning tree over the dataset. Then, starting with a single region at the root of the tree, a model is fitted to the instances within the region to summarise the data. Next, kD-STR iteratively determines whether to partition the $T \times S^{\mathcal{D}}$ space into more non-overlapping regions, or to increase the model complexity of one of the existing regions with the aim of improving its accuracy. The decision taken at each step is that which minimises the objective function $h(D, \langle R, M \rangle)$, and the algorithm terminates when the objective function cannot be minimised further. An overview of this process is shown in Figure 2.

When kD-STR terminates, the algorithm outputs the set of regions and models, as shown in Figure 2(d). In this section, we describe each of the steps of kD-STR and the process of reconstructing the data from the reduced form.

4.1 Data Partitioning

By identifying spatio-temporal regions of similar instances, kD-STR is able to reduce a dataset to a set of regions and models. Whilst methods exist for creating a partition tree over a dataset, such as quadtree and octree decomposition [40], the number of new partitions introduced by these methods at each level of partitioning is fixed. Instead, it is more beneficial to use the variation of feature values over space and time to determine how many partitions are introduced at each level of the partition tree. kD-STR uses the difference between instances within the data itself to determine how many regions should be placed over the $T \times S^{\mathcal{D}}$ space at each level of partitioning.

Table 2. Example footfall data recorded at the 9 sensors (A-K) shown in Figure 1.

		Sensor										
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
Time step	**0**	252	278	148	193	279	248	267	296	45	241	58
	1	247	305	153	145	301	212	207	292	67	201	52
	2	210	296	139	134	299	199	192	287	39	189	46

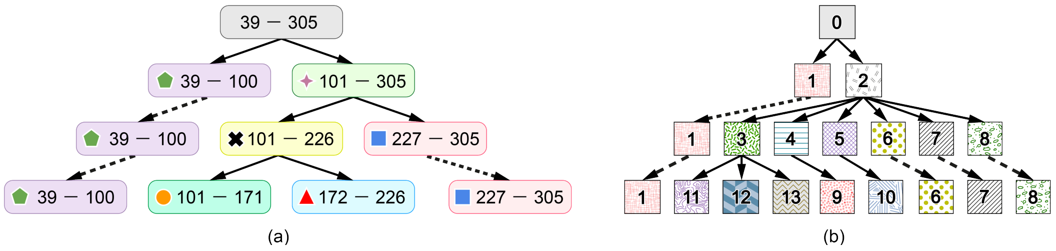


Fig. 3. (a) The cluster tree created by hierarchical clustering on the data in Table 2. Each node in the cluster tree shows the bounds defining the cluster. Solid arrows are used to show which clusters decompose into new clusters at each level of the tree, and dashed arrows are used to show clusters that remain the same. (b) The resulting partitioning tree.

Each region is defined by a piece-wise linear polygon in the spatial domain and a beginning and ending time in the temporal domain.

To begin partitioning the dataset, the instances are first clustered into a set of clusters C using hierarchical agglomerative clustering in the feature space (further information on hierarchical clustering of features can be found in [33]). For example, a rainfall dataset can be clustered using the *total precipitation* feature, ignoring the spatial and temporal domains. By clustering instances in the feature space, rather than clustering in the $T \times S^{\mathcal{D}}$ space, *k*D-STR clusters together instances that have similar feature values regardless of when and where the instances were recorded. Hierarchical clustering is used as the clusters found in the feature space are not expected to be globular and the resultant *cluster tree* allows regions and models to be retained in some areas or time periods as the number of clusters required changes [18]. To illustrate this further, consider the dataset shown in Table 2 containing data from footfall sensors that record the number of people that walk through an area at three consecutive time steps. The data is hierarchically clustered using just the raw footfall count values. This results in the cluster tree shown in Figure 3(a), where each node in the tree shows the boundaries of the clusters in the footfall feature space.

After the clustering tree is formed for the feature space, regions in the spatio-temporal $T \times S^{\mathcal{D}}$ space are found for each level of the cluster tree. For a given level of the tree, each instance in the $T \times S^{\mathcal{D}}$ space is labelled with the cluster it has been grouped into. Then, homogeneous regions that are connected components belonging to the same cluster in the $T \times S^{\mathcal{D}}$ space are found. Since storing the bounding spatio-temporal polygon of each homogeneous region may require many coordinates to be stored, we assert that each region must be defined by a single start and end time to limit the shapes in the $T \times S^{\mathcal{D}}$ space that regions may take.

However, finding connected components that are limited in shape is not trivial. In previous work, 2D-STR found a connected component in 2 dimensional space (i.e., 1 spatial dimension and 1 temporal dimension) by selecting an instance at random as the start of a new region, and growing

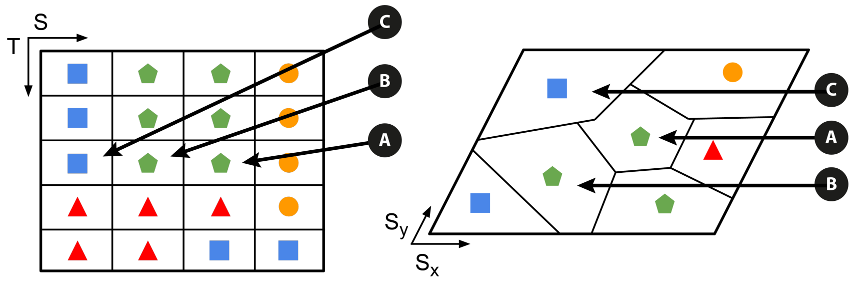


Fig. 4. (a) Instance **A** is chosen as the start of a new region. The region is expanded left to include instance **B** but not instance **C** as it belongs to a different cluster. (b) Instance **A** is chosen as the start of a new region. However, the region cannot be expanded left as both instances **B** and **C** are neighbours on the left of instance **A**.

the region as far as possible in each direction of each dimension [43]. That is, when the dataset is viewed as a 2-dimensional matrix, a region could be expanded beyond a beginning instance by moving as far left and right as possible, then as far up and down as possible. In the example shown in Figure 4(a), if instance **A** is chosen as the beginning instance of a new region, the region is first grown by extending the left boundary leftwards, adding instance **B** to the region, and then stopping as instance **C** belongs to a different cluster and cannot be added to the region. This process is only possible as there is a natural ordering to the sensors in the spatial and temporal dimensions. For the spatial dimension, sensors are naturally ordered by their position along the dimension. However, when the number of spatial dimensions is greater than 1, this natural ordering is not present and so it is ambiguous how to expand each region in the spatial domain. In the 3 dimensional example shown in Figure 4(b), suppose that instance **A** is again chosen as the start of a new region. If the region was expanded left in the S_x direction, both instances **B** and **C** would be added to the region. Adding instance **B** is acceptable but adding instance **C** would break the region's homogeneity. Therefore, the aim is to add instance **B** to the region without having to add instance **C**.

To overcome this issue, we first partition the spatial domain S into discrete polygons around each sensor using Voronoi partitioning [3] as shown in Figure 1(a). A similar action is also performed for the temporal domain T, forming a discrete timestep around each unique time present in D. By discretising the spatio-temporal dimensions, we can view the instances in D as a spatio-temporal graph, where each vertex is an instance and edges link vertices that are *adjacent*. We say two instances are adjacent if:

(i) they were recorded consecutively at the same sensor, or
(ii) they were recorded at the same time and the polygons surrounding their sensors are adjacent in the discretised spatial domain.

After discretising the spatial and temporal domains, *k*D-STR extends regions in a breadth-first manner. After an instance is chosen as the start of a new region, all instances that are adjacent to this instance in the spatial domain and belong to the same cluster are added to the region. In Figure 4(b), this would add instance **B** to the region but not instance **C**. Then, after all spatial neighbours of the initial instance are considered, the temporal boundary is extended by up to 1 timestep before and after the initial instance if doing so does not break the cluster homogeneity of the region. This process is repeated continuously, expanding the boundaries of the region by a depth of 1 neighbour to the existing instances in the region spatially, and 1 timestep before and after the region, until the spatial and temporal bounds of the region cannot be expanded any further.

Converting a level of the cluster tree into a level of the partition tree is complete when all instances in D are associated with a homogeneous region within that level of the partition tree. The result is a partition tree in which each instance is assigned to a region at each level of the tree. The relationship between clusters and regions is shown in Figure 3. The first four levels of the cluster tree are shown in subfigure (a), and the corresponding first four levels of the partitioning tree are shown in subfigure (b). Each level of the partitioning tree is also shown in Figure 5. At the root of the cluster tree, at level 1 with 1 cluster, all 33 instances shown in Table 2 are placed into a single region, namely region 0. On level 2 of the cluster tree, with 2 clusters, region 0 is decomposed into regions 1 and 2 in the partitioning tree. On level 3, with 3 clusters, the *101 – 305* cluster has been decomposed into 2 more clusters and region 2 has been decomposed into regions 3 to 8 accordingly. Finally, on level 4 of the cluster tree, the *101 – 226* cluster has been decomposed and region 3 decomposed into regions 11–13, region 4 into region 9, and region 5 into region 10. Note again that while some adjacent regions may contain instances that belong to the same cluster, a region must be defined by a bounding region, and a single start and end time. Thus, while regions 7 and 8 on $t = 0$ are adjacent and belong to the same cluster, region 7 also spans $t = 1$ and $t = 2$ while region 8 only spans $t = 0$. Therefore, these regions cannot be combined. Furthermore, whilst the sensors and time periods covered by regions 4 and 5 do not change, the parent cluster was decomposed and so they were replaced by regions 9 and 10 respectively.

By using hierarchical clustering, we ensure that only some regions in the partitioning tree are decomposed between levels. This results in some regions remaining unchanged throughout multiple levels of the partitioning tree, and this can be exploited in the reduction process by retaining models of these regions during reduction. Furthermore, the models of regions that are replaced but retain the same sensors and time periods can also be retained.

4.2 Region Modelling

After partitioning D into a hierarchy of regions, a technique is required to model the instances within each region. We may form a model for the instances within each region or, since each region is associated with a cluster, we may form a model per cluster and link all regions belonging to the same cluster to the same model. To maximise the utility of the reduction, we require the ability to reconstruct the instances of the original dataset from the models output. Furthermore, we wish to enable the imputing of instances in spatial and temporal locations that have not been sampled in the dataset, but have nearby sensors or time periods.

For each region r_i or cluster c_i in each level of the partitioning tree, a model m_j is fitted to the instances within the region or cluster. The spatial and temporal values of the instances are used as the independent or predictor values of the model, whilst the feature values of the instances are used as the dependent or response values of the model. That is, $m_j : \mathbb{R}^k \rightarrow \mathbb{R}^{|F|}$. For example, in the case of polynomial regression (PLR) when $k = 3$, the model m_j may be:

$$d^f_{t,s} = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 t + \epsilon \tag{8}$$

where x and y are the spatial coordinates of $d_{t,s}$, i.e., $(x, y) = s$. Here, the feature values in the dataset D are modelled as linear outputs of the spatial and temporal positions of each instance. Thus, the modelling step computes and stores the model coefficients $\beta_0, \beta_1, \beta_2$ and β_3.

However, in many cases, the feature values may not be linearly dependent on the spatial and temporal positions of the instances, and so adding interaction terms to the model may improve its accuracy. For PLR, this is achieved by increasing the *degree* of the model, although this increases the number of model coefficients stored. Thus, on each iteration, *k*D-STR tests the change in $h(D, \langle R, M\rangle)$ (Equation 7) when the degree of the model is increased by 1.

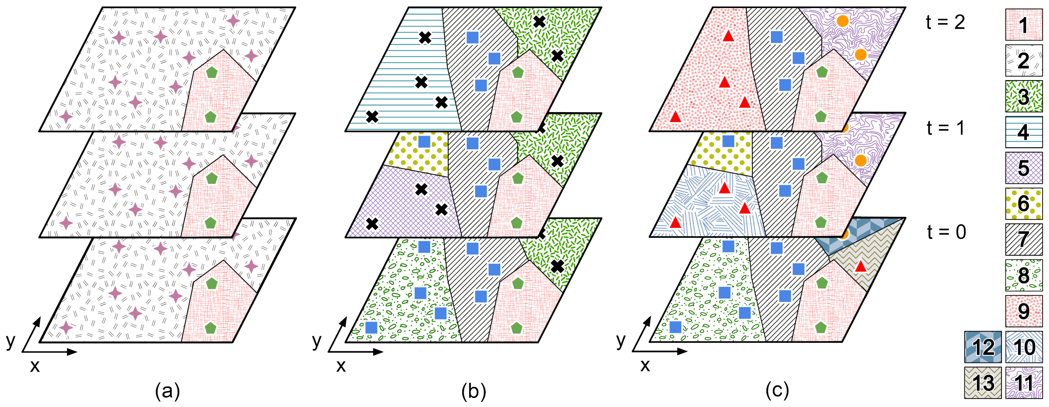


Fig. 5. The spatio-temporal regions formed by the data partitioning process, when applied to the data shown in Table 2. Subfigures (a), (b) and (c) show the partitions at levels 2, 3 and 4 of the partitioning tree respectively. Subfigure (c) is also shown in Figure 1(c), and regions 1–9 in Figure 1(c) correspond to regions 1 and 6–13 here.

In this paper, we consider three illustrative modelling techniques. First, we consider multivariate polynomial regression (PLR) owing to its ability to explain data that is spatially and temporally autocorrelated. The model equation used for 1 degree of freedom is shown in Equation 8. More information about PLR can be found in literature [41]. Second, we consider multidimensional discrete cosine transform (DCT) approximation for its ability to model periodic data using few coefficients. By removing low-weighted coefficients and storing just those that are weighted highly, the original data can be reproduced with reasonable accuracy. For a detailed description of multidimensional DCT approximation and data reconstruction, we refer the interested reader to [38].

Finally, we consider decision tree regression (DTR) owing to the interpretability of the models output. The DTR method forms a decision tree over the spatial and temporal features and, for each leaf node in the tree, computes a regression tree for the instances belonging to that node. Furthermore, the method uses the *maximum depth* of the tree as a method for controlling the number of regression models computed and regression coefficients stored. We refer the interested reader to [7] for further information on DTR modelling.

For each of these techniques, we may form a model per region or a model per cluster, and so we refer to these as PLR-R and PLR-C, DCT-R and DCT-C, and DTR-R and DTR-C respectively. If the type of analysis to be performed after reduction is already known, more appropriate modelling techniques may be used that offer higher accuracy, smaller storage or permit faster answering of queries. Overall, we wish the storage requirements of each model to be less than the storage requirement of the original instances the model was fitted to, and for the modelling technique to use the independent and dependent variables described above.

4.3 Data Reduction

The kD-STR algorithm can be seen in Algorithm 1, in which a model is formed for each region, although the algorithm may be easily adapted to model on clusters instead. Prior to initialising kD-STR, a value for the parameter α must be chosen, where $0 \leq \alpha \leq 1$. A discussion on choosing a value for α is provided in Section 7.1.

After a partitioning tree has been formed for the dataset D (Algorithm 1, lines 1–3), and a value for α and a modelling technique have been chosen, the reduction steps of kD-STR are performed.

Algorithm 1: *k*D-STR

```
clusterTree = cluster($D$);
numberClusters = 1;
$R$ = findRegions($D$, clusterTree, numberClusters);
$M = \{\}$ ;                          // Initialise the set of models to the empty set
for $r_i$ in $R$ do
    $m_i$ = model($D_i$, 1) ;          // Model the data in region $r_i$ using the simplest complexity
    $M$.add($m_i$);
$h$ = heuristic($D, R, M$) ;          // Calculate heuristic for 1 region and simple model
// Now iterate until heuristic $h$ is minimised
do
    // First, check if increasing an existing model's complexity minimises $h$ further
    $h_1 = h$;
    for $r_i$ in $R$ do
        $M' = M$;
        $m_i'$ = model($D_i$, $m_i$.complexity + 1);
        $M'$.replace($m_i, m_i'$) ;                    // Replace $m_i$ with $m_i'$ in $M'$
        $h'$ = heuristic($D, R, M'$);
        if $h' < h_1$ then
            $h_1 = h'$;
            $M_{best} = M'$;
    // Second, check if increasing the number of regions minimises $h$ further
    $R'$ = findRegions($D$, clusterTree, numberClusters+1);
    $M'' = \{\}$;                     // Initialise the set of models to the empty set
    for $r_i$ in $R'$ do
        if $r_i$ in $R$ then
            $M''$.add($m_i$) ;                          // Add $m_i \in M$ to $M''$
        else
            $m_i''$ = model($D_i$, 1);
            $M''$.add($m_i''$);
    $h_2$ = heuristic($D, R', M''$);
    // Finally, if increasing the number of regions, or the complexity of an existing model,
       is more optimal than $h$, take that choice
    if $h_1 < h_2$ and $h_1 < h$ then
        $M = M_{\text{best}}$;
        $h = h_1$;
    else if $h_2 < h_1$ and $h_2 < h$ then
        $R = R'$;
        $M = M''$;
        $h = h_2$;
        numberClusters = numberClusters + 1;
while $h_1 < h$ or $h_2 < h$;
return $R$, $M$
```

*k*D-STR begins at the root of the partitioning tree, with all instances belonging to a single region. This region is then modelled using the simplest form of the modelling technique used (lines 5–7): 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 highest weighted cosine coefficient is considered; and, in the case of DTR the decision tree is limited to a depth of 1. Note that the value 1 is used in line 6 to indicate the simplest form of model. After the model is fitted to the data, the result of the objective function $h(D, \langle R, M \rangle)$ is calculated for this first reduction step (line 8).

After the first reduction step, the algorithm iterates. On each iteration, *k*D-STR decides whether to increase the complexity of one of the existing models (lines 11–18) or increase the number of regions in the partitioning of the $T \times S^{\mathcal{D}}$ space (lines 19–27). When the number of regions is increased, only some existing regions are decomposed. For regions that are not decomposed, the models for these regions persist thereby improving the efficiency of *k*D-STR (lines 22–23). For previous regions that are decomposed, new regions are found and new models are fitted to these regions (lines 24–26).

The *k*D-STR algorithm stops when the objective function cannot be minimised further, i.e., $h_1 \geq h(D, \langle R, M \rangle)$ and $h_2 \geq h(D, \langle R, M \rangle)$. When the algorithm stops the reduction is complete and the set of regions and models, $\langle R, M \rangle$, is output.

4.4 Data Reconstruction

To reconstruct an instance from the reduced dataset $\langle R, M \rangle$, the location and time of the instance is required, (t, s). First, the region containing (t, s) is selected (as shown in Figure 2(e)). Then, the selected region's model is retrieved and (t, s) is input into the model (as shown in Figure 2(f)). The output of the model is the reconstructed feature values of D at (t, s). For example, in the case of PLR, the instance can be reconstructed using Equation 8. In the case of DCT, the feature values are the weighted sum of cosines of the input location and time values, as discussed in [38]. In the case of DTR, the input location and time are parsed by the decision tree and the resulting regression coefficients retrieved. After this, a similar equation to Equation 8 is used to reconstruct the feature values, as discussed in [7]. Note that an instance at that location and time may not have been present in the input raw dataset prior to reduction, as all three methods provide continuous models that allow for data imputation. This allows the data at a location and time to be imputed directly without requiring multiple other instances be reconstructed as is required by some state of the art algorithms.

4.5 Analysis of Running Time and Memory

To understand the time and memory complexities of *k*D-STR, we consider the startup cost of clustering the dataset and the cost of each iteration of the algorithm separately. In the startup phase of the algorithm (line 1 of Algorithm 1), hierarchical clustering is performed. Whilst hierarchical agglomerative clustering runs in $O(|D|^3)$ time and requires $O(|D|^2)$ memory, a more efficient approximation has been demonstrated to reduce the time complexity to $O(|D|^2)$ [32]. Therefore, the startup phase of *k*D-STR requires $O(|D|^2)$ time and $O(|D|^2)$ space.

After clustering, the algorithm iterates in successive rounds, either increasing the number of regions in the reduced dataset or increasing the complexity of a model. To increase the number of regions, the number of clusters is increased by 1. Each instance is labelled with the ID of its cluster in the next level of the cluster tree, and homogeneous regions are found for each cluster at this next level. To create a region, an instance that has not been assigned to a region is chosen at random as the start of the next region, and other instances adjacent to the instance are added to the region if they belong to the same cluster. This process is repeated in rounds until no more instances can be added to the region. During this process, the boundary between any two adjacent instances is checked at most twice. First, the boundary is checked in one direction to see if the two instances belong to the same cluster. If they do not, the second instance is not added to the region. In this case, at a later time of processing the second instance will be added to a different region and the same boundary is checked again to see if the first instance can be added to the new region. Since that instance already belongs to a region it will not be added, but the adjacency edge between the two instances has now been considered twice. This process requires $O(x|D|)$ time, where x is the maximum number of instances that are adjacent to any single instance in D. Furthermore, this process requires $O(|D|)$ memory to store the cluster and partition ID of each instance, as well as a working list of the boundary instances of the region being expanded.

As well as increasing the number of regions on each iteration, the algorithm also considers increasing the complexity of an existing model. For example, in the case of polynomial linear regression (PLR), this requires $O(y^2|D|)$ time and $O(y^2)$ memory per model, where y is the number of coefficients calculated for the model. Discrete cosine transforms (DCT) can be performed in $O(|D|^2)$ time and $O(1)$ memory per model, although fast cosine transforms can be performed in $O(|D| \log |D|)$ time. Similarly, decision tree regression (DTR) can be performed in $O(k|D|^2)$ time and $O(k)$ memory per model. Therefore, the startup phase of *k*D-STR requires $O(|D|^2)$ time and $O(|D|^2)$ memory, whilst each iteration, in the case of PLR, requires $O((x + y^2|M|)|D|)$ time and $O(|D| + y^2|M|)$ memory.

5 EVALUATION METHODOLOGY

To evaluate *k*D-STR, we compared the rate of decrease in storage with the rate of increased error for three spatio-temporal datasets taken from different sources. To measure the error introduced by the reduction process, we considered the normalised root mean square error (NRMSE) as shown in Equation 2. The NRMSE metric was used as it indicates how well each feature is modelled by the reduction process as a percentage error of the range of original values. To measure the reduction in storage between the original and the reduced datasets, we used the storage ratio defined in Equation 6. This indicates how much smaller the reduced dataset is compared to the original dataset. The number of regions in the reduced dataset was also considered, since this indicates how many data points in the reduced dataset have to be processed for many types of query and analysis.

To evaluate the generality of *k*D-STR to a range of spatio-temporal data, we considered datasets that exhibit different distributions and characteristics in space and time. We considered three sources of data: air temperature sensors, traffic counting sensors and rainfall sensors. Each of these datasets exhibits different spatial and temporal variances, that is, the dissimilarity between nearby instances in space and time. Furthermore, while the air temperature and road traffic datasets continuously evolve, meaning that there is a predictable trend of increasing and decreasing feature values over space and time, the rainfall dataset is event driven. Rainfall events are localised to small groups of sensors at the same time, and there are discontinuities between no rainfall (a recording of 0 mm rainfall) and rainfall (non-zero values) between consecutive time intervals at the same sensor. We expected the reduction of datasets exhibiting higher variance in space to yield more regions in space than time, and vice versa. The characteristics of the three datasets used are shown in Table 3. Using these characteristics, other datasets may be likened to those tested. The spatio-temporal datasets used to evaluate *k*D-STR in this paper are as follows.

(1) **Air Temperature**: 12 samples of month-long air temperature data were collected from the Met Office Integrated Data Archive System (MIDAS) Land and Marine Surface Stations Dataset in the United Kingdom (UK) [34]. The samples used were taken from January to December 2017 and covered all Met Office temperature recording sensors in the UK, which recorded one instance per hour. The dataset contains three real-valued features: *temperature*

Table 3. Characteristics of the datasets used for evaluation

	Spatial Variance	Temporal Variance	Discontinuities in Space	Discontinuities in Time
Air Temperature	Low	Low	Low, nearby sensors recorded similar feature values	Low, small variations between adjacent time intervals
Traffic	Low	High	Nearby sensors on main carriageway recorded similar feature values at same time, but differed substantially from sensors on slip roads; road traffic collisions introduced discontinuities in space	Road traffic collisions introduced discontinuities in time
Rainfall	Changed over time	Low	Rainfall events are often localised to groups of nearby sensors at the same time	Varied substantially over time as rainfall events occurred

(°C), *wet bulb temperature (°C)* and *dew point (°C)*. The values of these features follow a daily trend of increasing and decreasing values. The samples used from this dataset contained between 240,201 and 266,197 instances each.

(2) **Traffic**: A set of 28 samples of the WebTRIS traffic dataset was evaluated [17]. Each sample was a month-long survey of traffic counting sensors taken from several roads in England over April 2017, September 2017, November 2017 and December 2017. These samples were taken from the A30, A66 and A69 trunk roads and the M1, M11, M20, and M56 motorways. These roads were chosen for their differing spatial distributions and traffic characteristics. The traffic dataset contains six real-valued features: *count of vehicles of length 0m to 5.2m*, *count of vehicles of length 5.21m to 6.6m*, *count of vehicles of length 6.61m to 11.6m*, *count of vehicles of length 11.61m or greater*, *total count of vehicles* and *average speed (mph)*. Similar to the air temperature dataset, the traffic dataset also exhibits daily trends as well as weekly trends. Sensors on slip roads (entries and exits) are interspersed amongst main carriageway sensors, and exhibit much lower traffic counts compared to the main carriageway sensors. The samples used from this dataset contained between 54,180 and 86,042 instances each.

(3) **Rainfall**: 12 samples of month-long rainfall data were used from the Met Office Integrated Data Archive System (MIDAS) [34]. Again, the samples used were taken from January to December 2017 and covered all Met Office rainfall recording sensors in the UK at hourly intervals[3]. The dataset contained a single feature, *precipitation (mm)*. However, unlike the air temperature and traffic datasets there was not a temporal pattern that was expected to be observed every day. Instead, the dataset contains many instances of 0 mm rainfall, especially in the summer months. It was expected this property would make the dataset more efficient to reduce than the traffic and temperature datasets, since large spatio-temporal regions can be reduced to a mean value of 0 mm. The samples used from this dataset contained between 194,371 and 215,119 instances each.

[3]Note the stations that recorded rainfall were not the same as the stations that recorded air temperature, and so this dataset contains a different spatial distribution to the air temperature dataset.

We used 6 modelling techniques in our evaluation of *k*D-STR: 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), decision tree regression on each region (DTR-R) and decision tree regression on each cluster (DTR-C). We evaluated these methods on both the clusters and regions generated by the data partitioning stage to test the hypothesis that modelling on clusters would output few but complex models and modelling on regions would output many simple models.

Alongside the 6 modelling techniques tested on the presented datasets, 5 values of α were tested, namely, $\alpha \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$. Finally, to compare *k*D-STR with other reduction methods for spatio-temporal datasets, we also used IDEALEM [52], DEFLATE [16] and PCA adapted for spatio-temporal sensor [15, 37].

6 RESULTS

In this section, we discuss the results of applying *k*D-STR to the datasets discussed in Section 5. We compare the results of the different modelling techniques and the effects of the parameter α. Furthermore, we discuss reduction when multiple spatial referencing systems (SRSs) can be used and the effect of increasing the dataset size, as well as the spatial and temporal properties of the datasets that can be found using the partitioning of space and time by *k*D-STR.

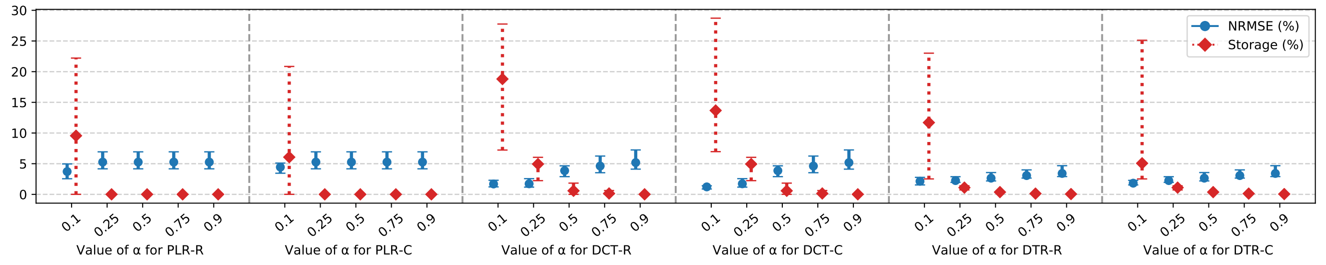
6.1 The Trade-off Between Reconstruction Error and Storage

For each of the three datasets evaluated, the mean and range of the reconstruction NRMSE and storage ratios of the *k*D-STR reduction are shown in Figure 6. A subfigure is shown per dataset, with each showing the results of using each of the 6 modelling techniques discussed in Section 5. Furthermore, each subfigure shows the results of the reduction process given the five values for the parameter α described in Section 5.

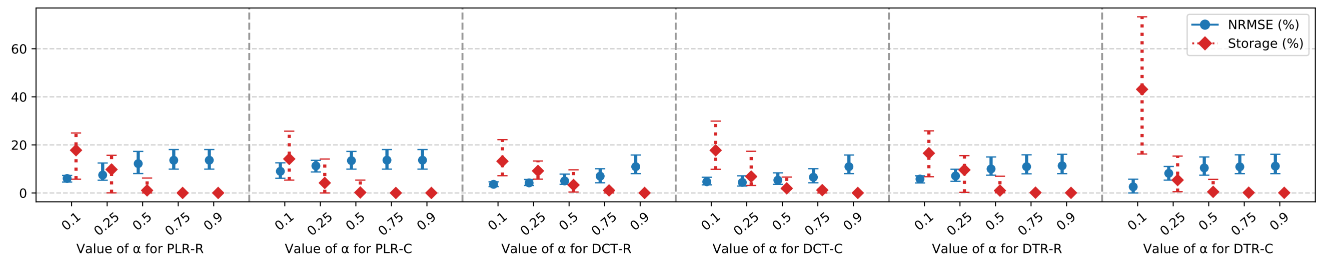
Several conclusions can be drawn from these results. In all cases, the reconstruction error introduced by the reduction was smaller for lower values of α than for higher values. Conversely, the storage ratio of data output was greater for lower values of α than for higher values. For DCT-R modelling on the air temperature dataset, the storage quantity used in the reduced datasets ranged from 7.2% to 27.8% when $\alpha = 0.1$, but decreased to 0.006% to 0.1% when $\alpha = 0.9$. Conversely, the error introduced ranged from 1.2% to 2.3% when $\alpha = 0.1$ but increased to 4.1% to 7.2% when $\alpha = 0.9$. A similar relationship between the error introduced and storage quantity used was observed across all of the datasets presented.

In some cases, particularly when modelling with polynomial regression, this trend plateaued after α increased beyond a certain value. For example, in the case of PLR-R and PLR-C on the air temperature and rainfall datasets, the value of α had little effect when $\alpha > 0.1$ and $\alpha > 0.5$ respectively. We attribute this to the cost of increasing the number of regions being too great, and the benefits of increasing the model complexity of the regions being insufficient to lower the objective function value. Furthermore, it was noted that as α was increased, increasing the number of clusters and regions to capture the spatio-temporal variance was not worthwhile, and so only a single region with a simple model was output. The exceptions to this were DCT and DTR modelling on the air temperature dataset and DCT modelling on the traffic dataset. As the value of α increased the reconstruction NRMSE observed also increased and did not appear to plateau.

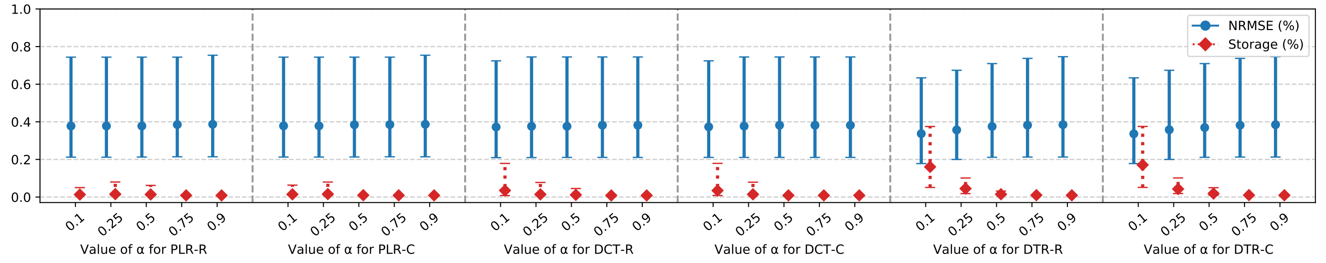
The range of reconstruction NRMSE and storage ratios also differed between datasets for the same modelling technique and value of α. The mean storage used was highest for the traffic dataset and lowest for the rainfall dataset. Similarly, the NRMSE achieved was highest for the traffic dataset and lowest for the rainfall dataset. We attribute this to the high temporal variance of the traffic dataset compared to the air temperature and rainfall datasets, and that the majority of rainfall instances



(a) Results of *k*D-STR on the air temperature dataset (12 1-month samples)



(b) Results of *k*D-STR on the traffic dataset (28 samples, collected over 4 months from 7 roads)



(c) Results of *k*D-STR on the rainfall dataset (12 1-month samples)

Fig. 6. Error introduced and storage used by *k*D-STR on the air temperature, traffic and rainfall datasets. Each sub-figure shows the results of *k*D-STR using the six modelling techniques evaluated for varying values of α.

had the same feature value (0 mm rainfall). Furthermore, for the traffic dataset, discontinuities in space resulted in boundaries between regions, often yielding a higher number of regions overall. This is discussed further in Section 6.6.

Finally, the relationship between storage and NRMSE differed between the modelling techniques tested. In most cases, as α increased, the storage used decreased in an exponential manner. However, the NRMSE appeared to follow a log-like pattern in some cases (PLR on temperature data, DTR on rainfall and traffic), quadratic pattern in some cases (DCT on traffic), arctan pattern in some cases (DCT on temperature, PLR on traffic) and a flat or linear pattern in others (DTR on temperature, PLR and DCT on rainfall). This suggests that the value of α used should be chosen after initial investigation of the data and, since similar relationships between NRMSE and storage ratio were observed across all samples of the same dataset given a particular modelling technique, the relationship between NRMSE, storage ratio and α is predictable once a subset of data samples have been tested.

6.2 Choice of Modelling Technique

From the results presented in Figure 6, several conclusions about the choice of modelling technique can be drawn. First, in most cases the choice between modelling on regions or clusters has little impact on the NRMSE introduced by the resulting reduction. The only exceptions noted are for the traffic dataset when $\alpha = 0.1$, where DTR-C reported slightly lower NRMSE than DTR-R, yet PLR-R reported slightly lower NRMSE than PLR-C.

Second, the storage used by the reduced dataset differs when modelling on regions versus clusters. In the case of the air temperature dataset, modelling on clusters yielded lower storage ratios on average than modelling on regions. For example, when $\alpha = 0.1$, DTR-R used 11.7% of the original storage volume on average whilst DTR-C used just 5.1%. The average number of regions output in the reduced dataset was lower when modelling on clusters (as shown in Table 4), though the mean number of coefficients stored per model was when modelling on clusters. This indicates the low spatial and temporal variance of the air temperature data can be more efficiently modelled with few complex models, rather than many simple models.

In contrast, DCT-C and DTR-C yielded a higher number of regions than DCT-R and DTR-R for the traffic dataset. More complex models were required to accurately capture the high temporal variance in the traffic data when modelling on clusters. This suggests that modelling on regions may be more efficient than modelling on clusters for datasets containing higher spatial or temporal variance, though this is more noticeable for DTR modelling than DCT. For the rainfall dataset, modelling on regions and clusters achieved approximately the same NRMSE and same storage ratio. However, we noted that when modelling on regions, each region stored one coefficient for its model, whereas when modelling on clusters, each region stored a single pointer to its cluster model. The difference in storage achieved by modelling on regions or clusters is therefore negligible for all three modelling techniques.

Third, the choice between polynomial linear regression, discrete cosine modelling and decision tree regression is dependent upon the user's preferences. For example, for both the air temperature and traffic datasets, DCT modelling was shown to achieve lower or similar NRMSE compared to PLR, although this came at the cost of increased storage overhead in many cases. However, the difference in NRMSE between the two techniques was quite small, with the largest difference occurring when $\alpha = 0.1$ on the traffic dataset. In this case, the average NRMSE reported was 4.7% for DCT-C and 9.1% for PLR-C. For the two NRMSE values reported, the storage used was 17.7% and 14.2% respectively. Decision tree regression, which yields easy to interpret models that can be output as a set of if-then rules, achieved similar NRMSE values on all three datasets when compared to DCT and PLR. However, this came at the cost of an increased storage overhead in the majority

Table 4. Average number of regions output by kD-STR, given differing values of α and the different modelling techniques evaluated.

	Air Temperature Dataset						Traffic Dataset					
α	PLR-R	PLR-C	DCT-R	DCT-C	DTR-R	DTR-C	PLR-R	PLR-C	DCT-R	DCT-C	DTR-R	DTR-C
0.1	4209	2683	8403	2688	4192	816	5163	4060	3792	4062	4037	4235
0.25	1	1	1	1	1	1	2743	1063	2638	768	2255	1029
0.5	1	1	1	1	1	1	268	42	562	43	153	43
0.75	1	1	1	1	1	1	3	1	3	3	3	3
0.9	1	1	1	1	1	1	3	3	3	3	1	3

	Rainfall Dataset					
α	PLR-R	PLR-C	DCT-R	DCT-C	DTR-R	DTR-C
0.1	3	3	3	3	3	3
0.25	3	3	3	3	3	3
0.5	3	1	3	1	1	3
0.75	1	1	1	1	1	1
0.9	1	1	1	1	1	1

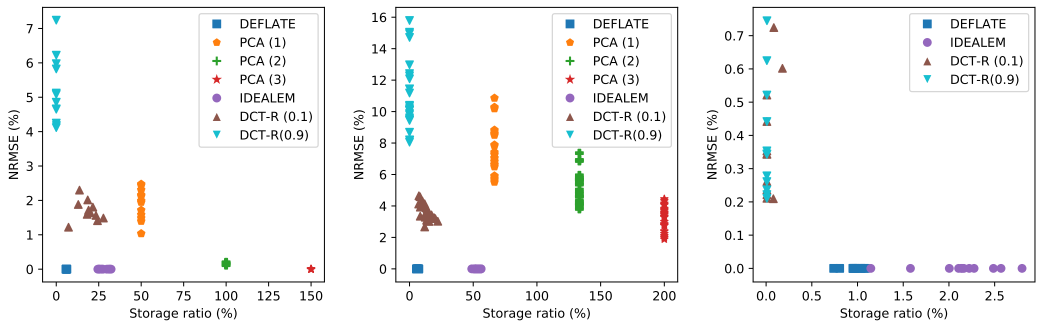
of cases. Therefore, when a model that is easy to interpret is required, DTR modelling is preferable, however more efficient modelling can be achieved by using PLR and DCT.

6.3 Comparison with Other Techniques

As discussed in Section 5, we compared kD-STR with the IDEALEM, PCA and DEFLATE methods. We used DEFLATE, a lossless compression algorithm, as a benchmark to show the reduction in storage achievable whilst losing no information. Whilst DEFLATE is able to compress the data to a form smaller than the raw dataset, the entire dataset must be decompressed before the data can be used for analysis. Thus, the user must have sufficient memory to store the decompressed raw dataset and cannot use the compressed data directly. In comparison, kD-STR allows the source data to be reconstructed by inputting the desired locations and times into the model(s) of the relevant regions in the reduced dataset. Furthermore, some analysis can be performed on the models output by kD-STR without needing reconstruct the data. Therefore, we include DEFLATE as an indicator of the potential reduction in storage that is achievable by lossless compression algorithms, rather than as a directly comparable reduction method. When evaluated on the datasets presented in this paper, DEFLATE reduced the data to between 0.7% and 7.2% of the original volume. DEFLATE reduced the air temperature datasets to an average of 6.0% of the original data volume, the traffic datasets to 6.3% and the rainfall datasets to 1.0%, as shown in Figure 7.

PCA, adapted for spatio-temporal data[4], was found to reduce the air temperature datasets to 50% of the original volume when 1 principal component was stored, and 100% (i.e., no reduction in storage) when 2 principal components were stored. An average of 1.9% and $3.95\times10^{-17}\%$ NRMSE was achieved respectively. On the traffic dataset, 1 principal component was found to produce an average error of 7.6% whilst using 67% of the original data volume, 2 principal components were found to produce an average error of 5.2% error whilst using 133% of the original data volume and 3 principal components again produced an average error of 3.2% error whilst using 200% of the original data volume. Whilst not shown on Figure 7(c), 1 principal component was also found to produce negligible NRMSE on the rainfall dataset whilst using 25% of the original data volume.

[4]We used the atmospheric science PCA adaptation, as discussed in [15], as this reflected the nature of data collected from sensors that are spatially and temporally autocorrelated.



(a) Air Temperature (b) Traffic (c) Rainfall

Fig. 7. Comparison of *k*D-STR with PCA, IDEALEM and DEFLATE on the air temperature, traffic and rainfall datasets. Note PCA is not shown for the rainfall dataset, though it achieved negligible NRMSE using 25% of the original dataset volume. For *k*D-STR, DCT-R was chosen to represent the range of reduction results seen.

IDEALEM was found to produce negligible NRMSE across all of the datasets evaluated. IDEALEM reduced the air temperature datasets to between 24.5% and 32.3% of the original data volume, the traffic datasets to between 48.8% and 56.2% and the rainfall datasets to between 1.1% and 2.8% of the original data volume.

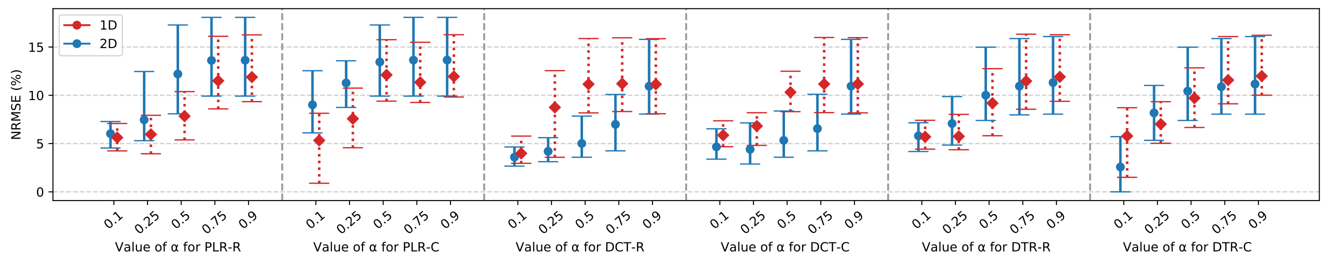
In comparison to these results, *k*D-STR was found to reduce the datasets to smaller volumes than IDEALEM and PCA, achieving storage ratios more similar to DEFLATE for both the air temperature and traffic datasets. Results for *k*D-STR using DCT modelling on regions (DCT-R) are shown to indicate the performance achieved by *k*D-STR compared to other techniques. When $\alpha = 0.1$, indicating a preference for minimal introduced error at the cost of increased storage used, *k*D-STR reduced the air temperature datasets to between 7.2% and 27.8%, the traffic datasets to between 7.2% and 22.2%, and the rainfall datasets to between 0.08% and 0.18% of the original dataset volumes. This was achieved by introducing an NRMSE of between 1.2% and 2.3%, 2.7% and 4.6%, and 0.2% and 0.8% accordingly. When $\alpha = 0.9$, indicating a preference for minimal storage at the cost of increased NRMSE, all datasets were reduced to less than 0.16% of the original volume. These reductions resulted in a NRMSE of between 4.1% and 7.2% for the air temperature datasets, 8.1% and 15.8% for the traffic datasets, and 0.2% and 0.8% for the rainfall datasets.

These results indicate that *k*D-STR is able to reduce the datasets evaluated to smaller storage volumes than IDEALEM and PCA, often with quite significant improvements. However, *k*D-STR may incur increased information loss compared to the other techniques evaluated.

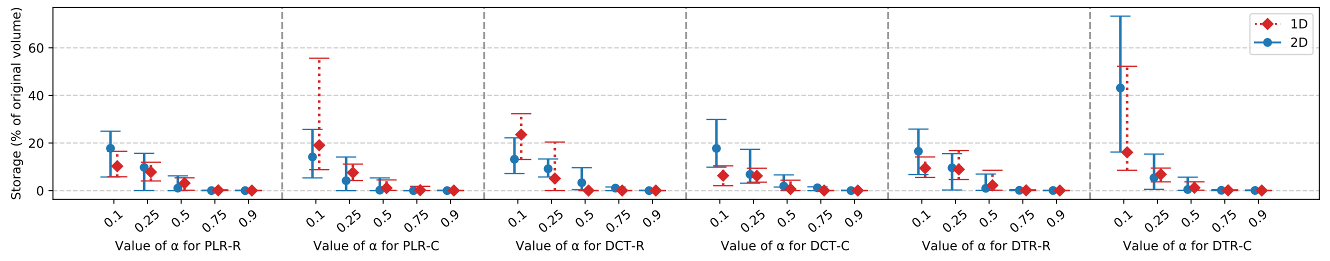
6.4 Choice of Spatial Referencing System

Some datasets can be referenced using multiple spatial domains. For example, transportation datasets that are sourced from roads can be referenced using a 2-dimensional spatial domain or a 1-dimensional spatial domain (or linear referencing system). When such datasets can be reduced using either of their spatial referencing systems (SRSs), the user must decide which they wish to use. Therefore, we compared the reduction of the 28 traffic dataset samples using *k*D-STR when 1-dimensional and 2-dimensional spatial referencing systems can be used, and refer to these as $k = 2$ and $k = 3$ respectively.

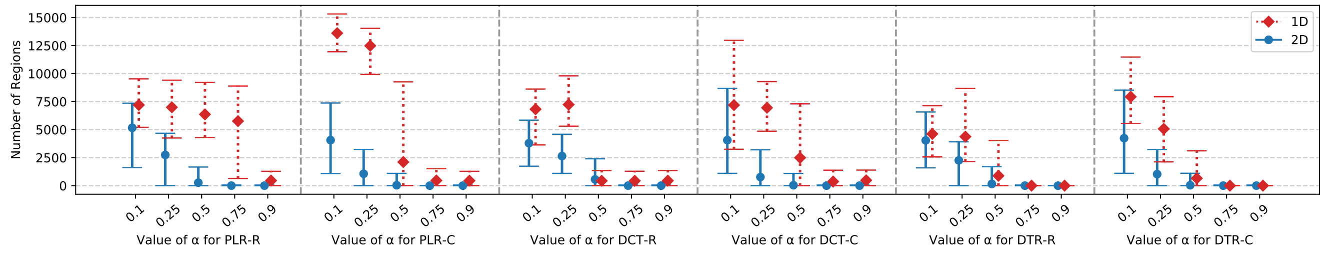
A comparison of the NRMSE incurred, storage used and number of regions output by *k*D-STR when applied to the traffic dataset using 1-dimensional and 2-dimensional spatial domains can be



(a) NRMSE



(b) Storage used



(c) Number of regions output

Fig. 8. Error introduced, storage used and number of regions output by kD-STR on the traffic dataset using a 1-dimensional and 2-dimensional spatial referencing system (SRS). Each sub-figure shows the results of kD-STR using the six modelling techniques evaluated for varying values of α.

seen in Figure 8. The relationship between NRMSE and storage used when $k = 2$ and $k = 3$ was similar for all modelling techniques used. However, the number of regions output when $k = 2$ was consistently higher than when $k = 3$. In some cases, for example DTR-R modelling, the number of regions output differed slightly whilst in others, for example PLR-C modelling, the number of regions output differed significantly. Furthermore, in some cases, notably PLR-R and DTR-R modelling, the higher number of regions used when $k = 2$, combined with fewer model coefficients stored per region, led to a lower or similar average NRMSE with a lower quantity of storage used. In most other cases a lower quantity of storage used correlated with a higher NRMSE achieved and a higher quantity of storage used correlated with a lower NRMSE achieved.

6.5 Impact of Dataset Size

To examine the effect of increasing the size of a dataset, we varied the sizes of the air temperature, traffic and rainfall datasets and investigated the impact on reduction. In the temporal domain, the dataset sizes were made larger by increasing the number of days in the dataset, thereby maintaining the spatio-temporal density while increasing the data size. The sizes tested were 0.25×, 0.5×, 1.0×, 1.5× and 2.0× the size of the data samples presented in Section 5 (corresponding to approximately 7.5, 15, 30, 45 and 60 days of data). In the spatial domain, we were only able to evaluate smaller samples (0.25×, 0.5× and 1.0×) as no further sensors existed.

When $\alpha = 0.1$, increasing the size of the air temperature dataset in the temporal domain increased the percentage of relative storage used in some cases. That is, the percentage of storage used compared to the raw dataset (Equation 6) increased. In other cases, increasing the size of the air temperature dataset in the temporal domain led to a decrease in the percentage of storage used. When the quantity of data increased, the number of partitions in the reduced dataset was proportionally higher and the NRMSE of the reduced data was lower. Similarly, when the quantity of data decreased, the number of partitions was proportionally lower and the NRMSE was higher. However, increasing the size of the air temperature dataset in the spatial domain decreased the storage used in all cases, and the NRMSE remained approximately the same. Furthermore, increasing the quantity of data in either domain decreased the storage used for the rainfall dataset and maintained approximately the same storage used, or lower, for the traffic datasets. Again, the NRMSE observed remained approximately the same. This can be seen in Figures 9 and 10, which show the effect of increasing the dataset size in the temporal and spatial domains respectively.

When $\alpha \in \{0.75, 0.9\}$, increasing the size of the dataset reduced the NRMSE in many cases and maintained approximately the same NRMSE in others. In most cases, the percentage of storage used relative to the raw data remained approximately the same as only 1 region was stored. However, in some cases when $\alpha = 0.75$, the number of regions was greater than 1 for smaller dataset sizes. Thus, as the size of the dataset increased, the number of regions decreased to 1, resulting in a decrease in storage used and an increase in NRMSE. For α values of 0.25 and 0.5, increasing the quantity of data in the dataset either led to an increase in the storage used and subsequent decrease in NRMSE, or a decrease in storage and increase in NRMSE.

Increasing the quantity of data in the temporal domain led to an approximately linear increase in the number of regions stored. This is demonstrated in Table 5, where increasing the quantity of air temperature data from 0.25x to 1.0x increased the number of regions from 2,503 to 9,618. However, increasing the quantity of data in the spatial domain led to a sub-linear increase in the number of partitions stored. We attribute this to the lower spatio-temporal variance of the datasets in the spatial domain requiring fewer regions and models.

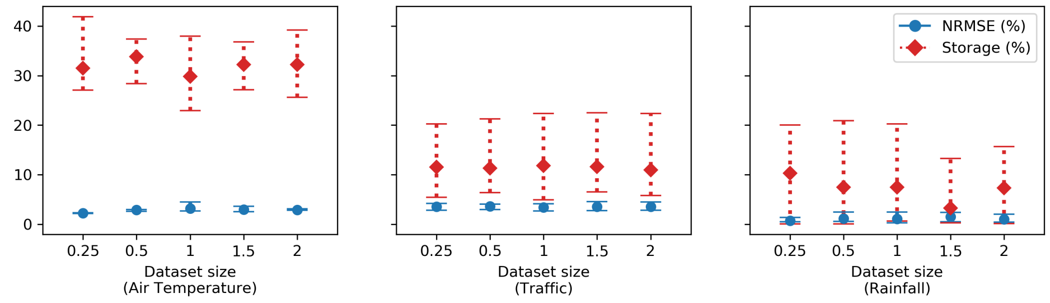


Fig. 9. Effect on NRMSE and storage used as the quantity of data in the temporal domain increases, with results shown for DCT-R modelling when $\alpha = 0.1$.

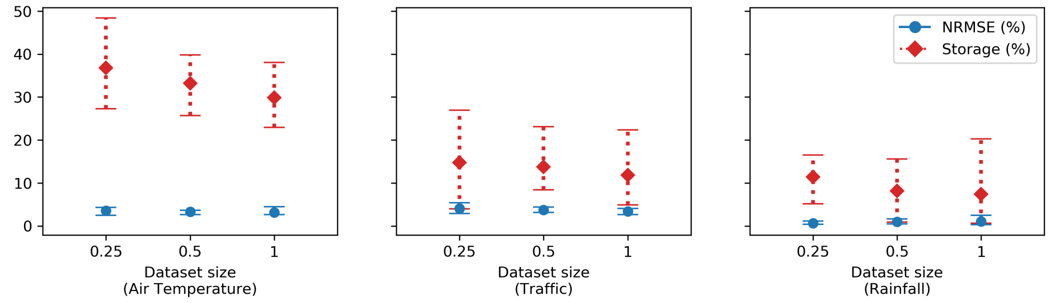


Fig. 10. Effect on NRMSE and storage used as the quantity of data in the spatial domain increases, with results shown for DCT-R modelling when $\alpha = 0.1$.

Table 5. Average number of regions stored as the quantity of data increases in the spatial and temporal domains. Results are shown for DCT-R modelling when $\alpha = 0.1$.

	Temporal Domain					**Spatial Domain**		
	0.25×	0.5×	1×	1.5×	2×	0.25×	0.5×	1×
Air Temperature	2503	5569	9618	15468	21247	3190	5499	9618
Traffic	848	1189	2448	1633	5091	1078	1429	2448
Rainfall	1662	3677	7494	10344	12881	2736	4300	7494

6.6 Spatial and Temporal Properties Discovered by *k*D-STR

The partitioning method used by *k*D-STR was found to identify several patterns and nuances within each of the three datasets tested. For example, for the air temperature dataset, several regions were created per day but few regions were created in space. We attribute this to the higher temporal variance in the dataset compared to its spatial variance. A similar partitioning in time occurred for the traffic dataset, however main carriageway and slip-road sensors were consistently placed in different regions during the daytime. This distinction is a result of the spatial discontinuity between the main carriageway and slip-roads which resulted in significant differences in the feature values recorded by the two types of sensor. From these two datasets, we therefore conclude that more regions were created in the spatial domain when the spatial domain has a higher variance than the temporal domain, and vice versa.

However, this conclusion did not hold for the rainfall dataset. Whilst the rainfall dataset experienced discontinuities at times of rainfall versus no rainfall, a maximum of 3 regions were introduced for each of the α values tested. We attribute this to the cost of increasing the number of regions being too high to be chosen by *k*D-STR. We therefore suggest that when the data contains a high enough number of discontinuities, for example a rainfall dataset that contains many occurrences of rainfall that are limited in their spatial area and duration in time, *k*D-STR will create few regions and attempt to model these discontinuities with complex models.

7 DISCUSSION

From the results presented in this section, several observations can be drawn. First, *k*D-STR is shown to be effective in reducing a spatio-temporal dataset to a set of regions of similar instances in space and time and forming a model of the instances within each of these regions. Furthermore, the parameter α is shown to be effective at allowing the user to decide between minimising the information lost and minimising the storage used. This is shown for a variety of datasets that exhibit different spatio-temporal characteristics, including different rates of variance in space and time, and different rates of continuous variability and sudden changes (events).

Second, *k*D-STR is shown to perform similarly when 1-dimensional and 2-dimensional spatial referencing systems are used for the same dataset. A similar relationship between NRMSE and storage is achieved both when $k = 2$ and $k = 3$, again showing that α can be used to control the rate of information lost and storage used. We believe this to be further indication of the generality of spatio-temporal datasets that can be reduced by *k*D-STR.

Third, the results shown in Section 6.1 indicate that the information lost and storage used are correlated more with the dataset than the method used to model the instances. This allows the user to select the modelling technique that is most applicable for their general form of analysis from a suite of possible modelling techniques. For example, if the user wishes to understand the relationships between feature values and the spatio-temporal dimensions, PLR may be preferable. However, when minimising NRMSE is more important it may be more beneficial to use DCT. Furthermore, when a more interpretable model is required for each region, DTR can be used.

Fourth, increasing the dataset size whilst maintaining approximately the same spatio-temporal variance and density of instances is shown to increase the size of the reduced dataset. When $\alpha = 0.1$, increasing the size of the dataset can lead to a decrease in storage whilst the NRMSE remains approximately the same. When $\alpha = 0.9$, increasing the size of the dataset can decrease the NRMSE whilst maintaining approximately the same storage ratio. In our results we noted that occasionally the percentage of storage used can increase as the quantity of data increases, though we attribute this to an increase in the number of regions stored as a result of increased spatio-temporal variance.

Finally, *k*D-STR is shown to achieve similar storage ratios to DEFLATE and to achieve smaller storage ratios than both PCA and IDEALEM. Whilst the lossless compression algorithm DEFLATE achieves 0% NRMSE, it does not permit analysis of the dataset without first requiring decompression back into the original dataset. In comparison, *k*D-STR permits multiple types of analysis on the reduced datasets without requiring a transformation back from the reduced state. *k*D-STR is also able to take advantage of the spatial nature of spatio-temporal data, unlike IDEALEM, and does not require a mapping from the stored models back to the original feature space like PCA.

7.1 Choice of value for α parameter

We use a parameter, α, in the objective function (Equation 7), to allow the user to indicate their preference for minimising the storage used by the reduced dataset and minimising the error incurred.

The parameter is bounded to be in the range $[0, 1]$ and must be determined before reducing the dataset. Intuitively, a value of $\alpha = 0$ reduces the impact of the storage metric to zero, and maximises the impact of the error metric. As a result, *k*D-STR would not stop iterating until the error metric $e(D, \langle R, M \rangle) = 0$. Such a perfect model may not be achievable and may be less efficient than storing the original dataset. Similarly, when a value of α =1 is chosen, both increasing the complexity of a model in M and decomposing the spatio-temporal space into more regions would increase the quantity of storage used beyond that of the initialisation step. Therefore, *k*D-STR would not iterate beyond the first iteration.

While neither of these extreme settings are appropriate, values of α close to 0 allow the user to minimise the error incurred while values close to 1 allow the user to minimise the storage used. From the results presented in Section 6.1, the same value of α yields different results for each dataset as the storage used and error incurred are directly affected by the variability of the dataset in space and time. That is, to achieve the same NRMSE or storage ratio for each dataset would require a different value of α that is optimised for each dataset. Moreover, the modelling technique chosen also impacts the exact results achieved. Yet, the results presented in Section 6.5 suggest that reducing a subset of a dataset may provide a sufficient estimate of the reduction and storage overhead for a full dataset. For the traffic and rainfall datasets, increasing the size of the dataset while maintaining approximately the same distribution of instances in the spatial and temporal domains led to a decrease in storage used, while the NRMSE incurred remained approximately the same. This was also true in some cases for the air temperature dataset, though in others the NRMSE decreased and the storage used increased. However, it is important to note that the differences incurred in these cases were small, and that the results achieved were broadly the same.

Therefore, by reducing one or more representative samples of a dataset with different values of α, the results achieved for the full dataset can be inferred (discussions on choosing an appropriate sampling technique can be found in literature [10, 14]). Reducing a representative sample of the data is likely to be much faster than reducing the full dataset, allowing different values of α to be tested. To choose a value for α, the user may choose a sample of the dataset that maintains approximately the same variability in space and time. That is, the semivariograms of the full dataset and representative sample should be similar. Then, this sample should be reduced using *k*D-STR with different values of the parameter α, and the storage reduction achieved measured alongside the reconstruction error incurred. When a value of α that achieves a satisfactory balance between storage reduction and minimised error is chosen, this value of α can be used to reduce the full dataset.

8 CONCLUSION

In this paper, we introduced *k*D-STR, a method for reducing spatio-temporal datasets in a manner that permits multiple types of analysis on the resulting reduced dataset. *k*D-STR overcomes the issue of region forming present in 2D-STR when the number of spatial dimensions is greater than 1. Furthermore, it overcomes the issues with existing model-based techniques as discussed in Section 2.5. *k*D-STR uses hierarchical partitioning to decompose space and time into regions of similar instances. Then, each region is modelled using an appropriate technique, thus reducing the original dataset to a set of regions and their models. This process reduces the quantity of data that needs to be processed for analysis and answering questions.

We have demonstrated the effectiveness of *k*D-STR in achieving an average reduction of 99.7% in storage on the datasets used for evaluation when a reduction in dataset volume is preferred, whilst achieving an average NRMSE of 7.6%. Further, we have demonstrated *k*D-STR achieves a reduction of 86.5% in storage for the same datasets, whilst achieving an average NRMSE of 3.5% when a minimal introduced error is preferred. The number of regions output is significantly smaller than the number of instances in the original datasets, reducing the quantity of data to be processed for analysis.

In comparison to other techniques, *k*D-STR is found to reduce a dataset to sizes similar to DEFLATE [16] whilst achieving NRMSE error rates similar to IDEALEM [25, 52] and PCA [15, 37]. Unlike IDEALEM and other reduction techniques, *k*D-STR stores information about all instances and features and, unlike ISABELA and other reduction by modelling techniques, permits single instance retrieval without requiring multiple other instances also be retrieved. Unlike data sketching techniques, *k*D-STR does not require prior knowledge about how the reduced dataset will be used and, unlike compression techniques such as DEFLATE, *k*D-STR permits analysis using the reduced form of the data. Therefore, *k*D-STR is shown to perform comparably to state of the art techniques whilst overcoming the issues they present.

We believe the investigation of dataset reduction methods to be an important avenue of research. Methods like *k*D-STR, which permit multiple types of analysis to be performed on the reduced dataset, are important in allowing data scientists to work in a more efficient manner as the quantity of data in spatio-temporal datasets increases. *k*D-STR has been shown to be effective at reducing data with different spatial and temporal variances and trends (such as air temperature and traffic data), as well as data that exhibits frequent discontinuities in space and time (such as rainfall data). However, we may take further advantage of the characteristics of such datasets. For example, datasets which exhibit oscillations will be broken into multiple regions by the partitioning stage of *k*D-STR. Yet, by partitioning the spatio-temporal space into regions of instances that exhibit clear trends or patterns, we may be able to further reduce such types of data. Future extensions of this work will also investigate the reduction of data that exhibits autocorrelations in higher numbers of dimensions, such as high-dimensional simulation data, and investigate the process of reducing multiple linked datasets at the same time. Furthermore, future extensions may examine the effect of reduction on linking spatio-temporal datasets and prioritise the performance of predictive models built using reduced datasets, rather than prioritising the retention of nuances present in the raw data. Finally, in this work we focused on modelling data for which a maximum of one instance was recorded at any given location and time. Future work may look at reducing datasets where multiple instances may be recorded at the same time and location, perhaps using modelling techniques that are built for multiple dependent or response values for the same independent or predictor values.

ACKNOWLEDGMENTS

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, and funding from TRL.

REFERENCES

[1] David Aha, Dennis Kibler, and Marc Albert. 1991. Instance-Based Learning Algorithms. *Machine Learning* 6, 1 (Jan. 1991), 37–66. https://doi.org/10.1023/A:1022689900470

[2] Hussein Almuallim and Thomas G. Dietterich. 1991. Learning with Many Irrelevant Features. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (Anaheim, California). AAAI Press, 547–552. http://dl.acm.org/citation.cfm?id=1865756.1865761

[3] Franz Aurenhammer. 1991. Voronoi Diagrams - a Survey of a Fundamental Geometric Data Structure. *Comput. Surveys* 23, 3 (Sept. 1991), 345–405. https://doi.org/10.1145/116873.116880

[4] Dimitris Berberidis and Georgios B Giannakis. 2017. Data Sketching for Large-Scale Kalman Filtering. *IEEE Transactions on Signal Processing* 65, 14 (July 2017), 3688–3701. https://doi.org/10.1109/TSP.2017.2691662

[5] Darius Birvinskas, Vacius Jusas, Ignas Martisius, and Robertas Damasevicius. 2012. EEG Dataset Reduction and Feature Extraction Using Discrete Cosine Transform. In *2012 Sixth UKSim/AMSS European Symposium on Computer Modeling and Simulation.* 199–204. https://doi.org/10.1109/EMS.2012.88

[6] Burton H Bloom. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM* 13, 7 (July 1970), 422–426. https://doi.org/10.1145/362686.362692

[7] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and Regression Trees*. CRC press.

[8] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering* 40, 1 (2014), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024

[9] Chin-Liang Chang. 1974. Finding Prototypes For Nearest Neighbor Classifiers. *IEEE Trans. Comput.* C-23, 11 (Nov. 1974), 1179–1184. https://doi.org/10.1109/T-C.1974.223827

[10] Michael Chipeta, Dianne Terlouw, Kamija Phiri, and Peter Diggle. 2017. Inhibitory Geostatistical Designs for Spatial Prediction Taking Account of Uncertain Covariance Structure. *Environmetrics* 28, 1 (2017), e2425. https://doi.org/10.1002/env.2425 e2425 env.2425.

[11] AB Arockia Christopher and S Appavu alias Balamurugan. 2013. Feature selection techniques for prediction of warning level in aircraft accidents. In *Advanced Computing and Communication Systems (ICACCS), 2013 International Conference on*. IEEE, 1–6.

[12] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. 2012. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases* 4, 1–3 (2012), 1–294.

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

[14] Eric M. Delmelle. 2014. *Spatial Sampling*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1385–1399. https://doi.org/10.1007/978-3-642-23430-9_73

[15] Urška Demšar, Paul Harris, Chris Brunsdon, A Stewart Fotheringham, and Sean McLoone. 2013. Principal component analysis on spatial data: an overview. *Annals of the Association of American Geographers* 103, 1 (2013), 106–128.

[16] Peter Deutsch. 1996. *DEFLATE compressed data format specification version 1.3*. Technical Report.

[17] Highways England. 2018. Highways England network journey time and traffic flow data. http://webtris.highwaysengland.co.uk. [Online; accessed 3rd October 2018].

[18] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y Zomaya, Sebti Foufou, and Abdelaziz Bouras. 2014. A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Transactions on Emerging Topics in Computing* 2, 3 (Sept. 2014), 267–279. https://doi.org/10.1109/TETC.2014.2330519

[19] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*. 137–156.

[20] Shlomo Geva and Joaquin Sitte. 1991. Adaptive nearest neighbor pattern classification. *IEEE Transactions on neural networks* 2, 2 (March 1991), 318–322. https://doi.org/10.1109/72.80344

[21] Andreas Janecek, Wilfried Gansterer, Michael Demel, and Gerhard Ecker. 2008. On the Relationship Between Feature Selection and Classification Accuracy. In *Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008*, Yvan Saeys, Huan Liu, Iñaki Inza, Louis Wehenkel, and Yves Van de Pee (Eds.). PMLR, 90–105. http://proceedings.mlr.press/v4/janecek08a.html

[22] Kenji Kira and Larry A Rendell. 1992. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of tenth national conference on artificial intelligence*. 129–134.

[23] Teuvo Kohonen (Ed.). 2001. *Self-Organizing Maps* (3rd ed.). Springer-Verlag. http://www.worldcat.org/title/self-organizing-maps/oclc/807084376

[24] Sriram Lakshminarasimhan, John Jenkins, Isha Arkatkar, Zhenhuan Gong, Hemanth Kolla, Seung-Hoe Ku, Stephane Ethier, Jackie Chen, C S Chang, Scott Klasky, Robert Latham, Robert Ross, and Nagiza F Samatova. 2011. ISABELA-QA: Query-driven analytics with ISABELA-compressed extreme-scale scientific data. In *SC 2011*. 1–11. https://doi.org/10.1145/2063384.2063425

[25] Dongeun Lee, Alex Sim, Jaesik Choi, and Kesheng Wu. 2016. Novel Data Reduction Based on Statistical Similarity. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*. ACM, 21:1–21:12. https://doi.org/10.1145/2949689.2949708

[26] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2017. Feature Selection: A Data Perspective. *ACM Comput. Surv.* 50, 6 (Dec. 2017), 94:1–94:45. https://doi.org/10.1145/3136625

[27] Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao. 2010. Feature selection: An ever evolving frontier in data mining. In *Feature Selection in Data Mining*. 4–13.

[28] Qin Liu, Ran Chen, Hongming Zhu, and Hongfei Fan. 2017. Research and Comparison of Data Dimensionality Reduction Algorithms. In *ICBCI 2017*. ACM, 49–54. https://doi.org/10.1145/3135954.3135965

[29] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic Flow Prediction With Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (April 2015), 865–873. https://doi.org/10.1109/TITS.2014.2345663

[30] Aleix M Martinez and Avinash C Kak. 2001. PCA versus LDA. *IEEE Transactions on pattern analysis and machine intelligence* 23, 2 (Feb. 2001), 228–233. https://doi.org/10.1109/34.908974

[31] Syrine Ben Meskina. 2013. On the effect of data reduction on classification accuracy. In *2013 3rd International Conference on Information Technology and e-Services (ICITeS)*. IEEE, 1–7.

[32] Daniel Mullner. [n.d.]. Fastcluster. http://danifold.net/fastcluster.html. Accessed 4th May 2019.

[33] Fionn Murtagh and Pedro Contreras. 2017. Algorithms for Hierarchical Clustering: An Overview, II. *WIREs Data Mining and Knowledge Discovery* 7, 6 (2017), e1219. https://doi.org/10.1002/widm.1219 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1219

[34] Met Office. 2012. Met Office Integrated Data Archive System (MIDAS) Land and Marine Surface Stations Data (1853-current). http://catalogue.ceda.ac.uk/uuid/220a65615218d5c9cc9e4785a3234bd0. [Online; accessed 3rd March 2018].

[35] Stefanos Ougiaroglou and Georgios Evangelidis. 2012. Efficient Dataset Size Reduction by Finding Homogeneous Clusters. In *Proceedings of the Fifth Balkan Conference in Informatics*. ACM, 168–173. https://doi.org/10.1145/2371316.2371349

[36] Bei Pan, Ugur Demiryurek, Farnoush Banaei-Kashani, and Cyrus Shahabi. 2010. Spatiotemporal Summarization of Traffic Data Streams. In *Proceedings of the ACM SIGSPATIAL International Workshop on GeoStreaming*. ACM, 4–10. https://doi.org/10.1145/1878500.1878504

[37] Karl Pearson. 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 (1901), 559–572.

[38] K.R. Rao and P. Yip. 2014. *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Elsevier Science.

[39] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.

[40] Hanan Samet. 1984. The Quadtree and Related Hierarchical Data Structures. *Comput. Surveys* 16, 2 (June 1984), 187–260. https://doi.org/10.1145/356924.356930

[41] Priyanka Sinha. 2013. Multivariate Polynomial Regression in Data Mining: Methodology, Problems and Solutions. *International Journal of Scientific and Engineering Research* 4, 12 (2013), 962–965.

[42] Sabina Sisovic, Marija Brkic Bakaric, and Maja Matetic. 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)*. 221–228. https://doi.org/10.1109/BigDataService.2018.00040

[43] Liam Steadman, Nathan Griffiths, Stephen Jarvis, Stuart McRobbie, and Caroline Wallbank. 2019. 2D-STR: Reducing Spatio-Temporal Traffic Datasets by Partitioning and Modelling. *Proceedings of the 5th International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM)* (May 2019).

[44] Kai Sheng Tai, Vatsal Sharan, Peter Bailis, and Gregory Valiant. 2018. Sketching Linear Classifiers over Data Streams. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 757–772. https://doi.org/10.1145/3183713.3196930

[45] Waldo R Tobler. 1970. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography* 46 (1970), 234–240. http://www.jstor.org/stable/143141

[46] Isaac Triguero, Joaquín Derrac, Salvador Garcia, and Francisco Herrera. 2012. A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 1 (Jan. 2012), 86–100. https://doi.org/10.1109/TSMCC.2010.2103939

[47] Jorge R Vergara and Pablo A Estévez. 2014. A review of feature selection methods based on mutual information. *Neural computing and applications* 24, 1 (2014), 175–186.

[48] Mingliang Wang, Han-Xiong Li, and Wenjing Shen. 2016. Deep auto-encoder in model reduction of lage-scale spatiotemporal dynamics. In *2016 International Joint Conference on Neural Networks (IJCNN)*. 3180–3186. https://doi.org/10.1109/IJCNN.2016.7727605

[49] Michael Whelan, Nhien An Le Khac, and M-Tahar Kechadi. 2010. Data Reduction in Very Large Spatio-Temporal Datasets. In *2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*. 104–109. https://doi.org/10.1109/WETICE.2010.23

[50] Michael Whelan, Nhien-An A Le-Khac, and M-Tahar Kechadi. 2011. Comparing two density-based clustering methods for reducing very large spatio-temporal dataset. In *Proceedings 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*. 519–524. https://doi.org/10.1109/ICSDM.2011.5969100

[51] Liang Ze Wong, Huiling Chen, Shaowei Lin, and Daniel Chongli Chen. 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*. ACM, 227–230. https://doi.org/10.1145/2641798.2641816

[52] Kesheng Wu, Dongeun Lee, Alex Sim, and Jaesik Choi. 2017. Statistical data reduction for streaming data. In *2017 New York Scientific Data Summit (NYSDS)*. 1–6. https://doi.org/10.1109/NYSDS.2017.8085035

[53] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. 2016. A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation* 20, 4 (Aug. 2016), 606–626. https://doi.org/10.1109/TEVC.2015.2504420