Discretisation of Data in a Binary Neural k-Nearest Neighbour Algorithm

Article · June 2012

CITATIONS
7

READS
453

2 authors:

Victoria Hodge
The University of York
74 PUBLICATIONS **4,483** CITATIONS
SEE PROFILE

Jim Austin
The University of York
263 PUBLICATIONS **6,089** CITATIONS
SEE PROFILE

DISCRETISATION OF DATA IN A BINARY NEURAL K-NEAREST NEIGHBOUR ALGORITHM

VICTORIA J. HODGE AND JIM AUSTIN

DEPT OF COMPUTER SCIENCE, UNIVERSITY OF YORK, YO10 5GH, UK,

{victoria.hodge, jim.austin}@york.ac.uk

ABSTRACT

This paper evaluates several methods of discretisation (binning) within a k-Nearest Neighbour predictor. Our k-NN is constructed using binary neural networks which require continuous-valued data to be discretised to allow it to be mapped to the binary neural framework. Our approach uses discretisation coupled with robust encoding to map data sets onto the binary neural network. In this paper, we compare seven unsupervised discretisation methods for retrieval accuracy (prediction accuracy) across a range of well-known prediction data sets comprising time-series data. We analyse whether there is an optimal discretisation configuration for our k-NN. The analyses demonstrate that the configuration is data specific. Hence, we recommend running evaluations of a number of configurations, varying both the discretisation methods and the number of discretisation bins, using a test data set. This evaluation will pinpoint the optimum configuration for new data sets.

KEYWORDS

k-Nearest Neighbour, binary neural network, discretisation, binning, quantisation

I. INTRODUCTION

Standard k-Nearest Neighbour (k-NN) is a widely applicable data mining algorithm that demonstrates high recall accuracy; see Fix & Hodges (1951), Cover & Hart (1967), Dasarathy (1991), Wettscherek (1994), Györfi et al. (2002), Hodge & Austin (2004a), Bubeck & von Luxburg (2009) and Hodge (2011) for an overview of k-nearest neighbour techniques. For both classification and prediction, k-NN examines those points in a particular data space lying "nearest" to a query point. K-NN then uses the respective classifications or predictions of these nearest neighbours to determine the class of the query point or to predict the next value in a time-series.

The computational growth of standard k-NN is $O(N^2)$ (Dasarathy, 1991; Knorr & Ng, 1998) with respect to the number of records N in the data set. This is because the approach calculates the distance to each record for every record in the data set. The computational complexity is also directly proportional to the dimensionality of the data d. As a result, there is a practical upper limit to both the number of records and the data dimensionality that may be processed even on modern high speed computers depending on the processor time available.

We have previously introduced a binary neural k-NN (Weeks et al., 2003; Hodge & Austin 2004b; Hodge & Austin, 2005), based on the Advanced Uncertain Reasoning Architecture (AURA) (Austin, 1995) that speeds the identification of the k-nearest neighbours while maintaining the recall accuracy of a standard k-NN procedure. This allows the AURA k-NN to process larger data sets in the same time as a standard k-NN procedure. Our previous empirical evaluations showed that it is approximately four times faster (with respect to time) than conventional k-NN (Hodge & Austin, 2005) and the graph trend indicates that this speed gain will be maintained for larger data sets, computational resources permitting. AURA k-NN has recently been applied to the task of traffic classification (Austin et al., 2010, Hodge et al., 2010; Krishnan et al., 2010a; Krishnan et al., 2010b; Krishnan et al., 2010c) and traffic prediction (Hodge et al., 2011).

Our k-NN requires that real-valued (continuous) attributes are discretised (binned) to allow them to be mapped onto the binary neural network that underpins our method. Dougherty et al. (1995) provide a survey of discretisation techniques and evaluated various techniques in conjunction with a Naïve-Bayes and C4.5 classifier in their seminal paper. Skubacz and Hollman (2000) survey various discretisation methods for classification. Liu et al. (2002) and Kotsiantis & Kanellopoulos (2006) provide more contemporary surveys. In this paper, we evaluate discretisation techniques suited to k-NN prediction using our k-NN predictor.

Discretisation (also called quantisation or binning) allows discrete learning algorithms (such as discrete classifiers or predictors) to handle continuous attributes. It can also be used to speed and even improve the accuracy of other learning algorithms (Witten & Frank, 1999). The discretisation procedure generally consists of two steps: selecting the number of discrete partitions (bins) and selecting the partitions (bin boundaries). Some discretisation methods are able to autonomously determine the number of partitions but this number is frequently user-specified using either a heuristic or by evaluating a range of values using a suitable evaluation criterion. Selecting the discretisation partitions comprises four steps (Liu et al., 2002) (1) sorting the attribute values; (2) evaluating a partition boundary for splitting or evaluating adjacent partitions for merging; (3) using a suitable criterion to split or merge partitions; and finally, (4) terminating when a stopping criterion is met.

Dougherty et al. (1995) classify discretisation techniques by three criteria: global or local; static or dynamic; and, supervised or unsupervised.

Discretisation may be global or local. Global discretisation is performed independently of the algorithm that uses the discretised data (a filter approach) whereas local discretisation is performed in conjunction with the algorithm (a wrapper approach). Equi-width discretisation (Dougherty et al., 1995; Liu et al., 2002) which divides the attributes into b bins of equal width and equi-frequency discretisation (Dougherty et al.,1995; Liu et al., 2002) which divides the attributes into b bins each containing an equal number of data points are both global methods. The ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1992) decision tree learners perform local discretisation by determining the partition boundaries as the decision tree is formed. The AURA k-NN requires global discretisation as the bins must be induced before training.

Discretisation may be static or dynamic. Dynamic discretisation takes account of attribute inter-dependencies whereas static discretisation does not take them into account. Dynamic discretisation is performed using all attributes as a set, for example, using the k-means clustering algorithm on all attributes to induce the partition boundaries (Min, 2009; Joita,

2010). In contrast, static discretisation is performed on the attributes individually for example, using a clustering algorithm such as k-means on one attribute at a time to induce the partition boundaries (Joita, 2010), using equi-width discretisation or using equi-frequency discretisation. We analyse both static and dynamic techniques in our evaluations.

Discretisation may be supervised or unsupervised. Supervised discretisation uses class label information to drive the discretisation procedure whereas unsupervised discretisation derives the bins independently of any class labels. Holte's 1R discretiser (Holte, 1993) is an error-based global supervised technique; it induces one-level decision trees (decision stumps). Fayyad & Irani (1993) introduced a global supervised entropy-based algorithm which selects the bin boundaries by recursively partitioning the attribute value range using top-down partitioning. Equi-width, equi-frequency and clustering using k-means (described later) are all unsupervised discretisation techniques. The evaluation in this paper concerns predicting future values of time series variables. This necessitates unsupervised discretisation as no classification labels are available. Hence, we evaluate seven unsupervised techniques in this paper.

The aim of the paper is: to identify the optimal discretisation technique from the techniques evaluated with respect to recall accuracy and recall consistency across a wide range of well-known data sets. We note that we only compare a standard k-NN discretiser with no amendments. We have not weighted attributes nor pre-selected attributes. We have not weighted the classification. i.e., we use simple majority voting rather than weighted majority voting (Wettscherek, 1994) which takes account of the distances to the nearest neighbours when calculating the prediction. This is to ensure consistency across the evaluations and to allow us to produce a definitive recall figure.

In the remainder of this paper we provide: a detailed overview of binary neural networks in section II; AURA and our k-NN and discretisation method in section III; a description of the evaluation methodology and the results in section IV; an analysis of the results in section V; a detailed discussion and comparison of the methods evaluated in section VI; and the conclusions we have drawn from our analyses in section VII.

II. BINARY NEURAL NETWORKS

AURA belongs to a class of binary neural networks called Random Access Memory (RAM-based) networks; see Austin (1998) for a detailed compilation of RAM methods. The first RAM-based networks were developed by Bledsoe & Browning (1959) and Aleksander & Albrow (1968) for pattern recognition and led to the WISARD pattern recognition machine (Aleksander, Thomas & Bowden, 1984)

RAM-based neural networks based on the twin tenets of matrices (usually called Correlation Matrix Memories (CMMs)) and pre-processing that maps the data onto the CMMs. Thus, the matrices store associations between inputs I_j and outputs O_j as shown in figure 1. There are many methods for pre-processing. The process of discretisation (binning) underpins the majority of these. In training the CMM matrix elements are initialised to 0. Each matrix takes m inputs as a vector which addresses m rows and takes n outputs as a vector which addresses n columns of the matrix. The vectors are binary. I_{jl} is set (i.e. the ith bit in vector I_j is a 1) if row l is active and I_{jl} is clear (i.e. the lth bit in vector I_j is a 0) otherwise. Bit O_{jk} is a 1 if column k is active and 0 otherwise. During the training phase, the matrix weights M_{lk} are set if the bit representing input row I_{jl} and the bit representing output column O_{jk} are BOTH set (both 1). During recall, the presentation of vector I_j elicits the recall

of vector O_j as vector I_j contains all of the addressing information (set bits) required to index and retrieve vector O_j from the matrix.

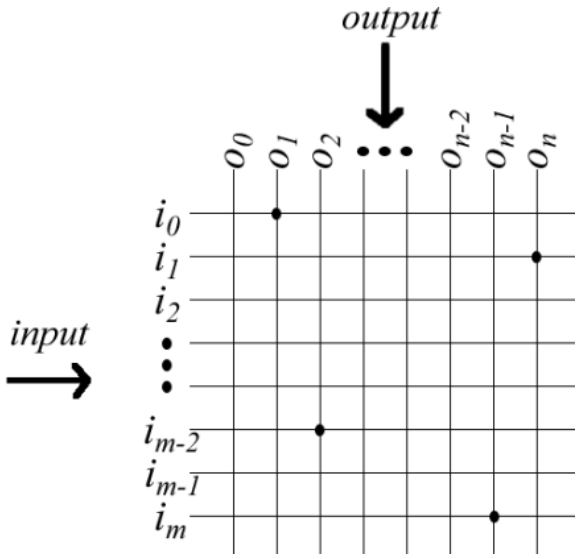


FIGURE 1 Diagram of a correlation memory matrix with input vector i_0, i_1, i_2, ..., i_{m-1}, i_m and output vector o_0, o_1, o_2, ..., o_{n-1}, o_n. The input vector addresses the rows and the output vector addresses the columns. The CMM is trained by associating input and output vectors which set the elements in the CMM to 1. All CMM elements are initialised as 0. In the diagram matrix elements i_0o_1, i_1o_n, $i_{m-2}o_2$ and i_mo_{n-1} are set.

In RAM-based networks, training is thus a single epoch process with one training step for each input-output vector association preserving the network's high speed and thus allowing large data sets to be processed. This simple association principle also makes RAM-based networks computationally simple and transparent with well understood properties. In contrast, most conventional neural networks used for classification such as MLP or RBF, (Bishop, 1995) require repeated training epochs and the resultant network is effectively a black box. RAM-based networks are also able to partially match records during retrieval. Therefore, they can rapidly match records that are close to the input but do not match exactly. This partial matching is a central concept for our binary k-NN described in the following paragraphs.

III. AURA K-NN

The k-NN method requires the top k matching examples (those most similar to the input) to be selected. The AURA methods use a thresholding technique called L-Max (described in section III-B.1) that retrieves the top *k* matches thus allowing the k-nearest neighbours to be determined. Here we have coupled this with a discretisation technique to map numeric data on to the binary inputs needed by the CMM. This rapid training, computational simplicity, network transparency, partial match capability and thresholding coupled with our discretisation technique make AURA ideal to use as the basis of an efficient k-NN implementation. The AURA C++ library provides classes and methods for the rapid partial matching of large data sets (Austin, 1995). AURA techniques have demonstrated superior performance with respect to speed compared to conventional data indexing approaches (Hodge & Austin, 2001) such as hashing and inverted file lists. AURA also has a scalable

architecture that can be easily mapped onto high performance computing platforms including parallel (Weeks, Hodge & Austin, 2002a; Weeks, Hodge & Austin, 2002b; Hodge, Jackson & Austin, 2011) and distributed platforms (Austin et al., 2005; Hodge, Jackson & Austin, 2011). A more formal definition of AURA, its components and methods now follows.

CMMs, shown in figure 1, are the building blocks for AURA systems. AURA uses binary and integer-valued input I_j and output O_j vectors to train records in to the CMM and recall sets of matching records from the CMM. For the methodology described in this paper, we use CMMs in the following way:

• Train the training data set into the CMM which indexes all records in the training data set and allows them to be matched. AURA logically ORs the CMM with the product of the vector formed from the record to store and an indexing vector to produce the trained CMM using binary vectors. This procedure is described in detail in section III-A. The training records each have their classification label stored with them.

• Apply query records to the CMM in turn and retrieve a set of the best matching records, i.e., the nearest neighbours. AURA uses the dot product of an integer-based vector formed from the query record and the trained CMM. This recall process is described in detail in section III-B.1. We classify the input query by taking the majority classification label from the set of records returned.

A. Training

In our k-NN implementation, input vectors represent discretised records during CMM training and output vectors uniquely identify each record in the data set. The training process is given in equation 1 for CMM M and associating input vector I_j with output vector O_j.

$$M = \bigvee_{all\ j}\left(I_j O_j^T\right) \text{ where } \vee \text{ is logical OR} \tag{1}$$

1) Discretisation:

The CMMs in AURA require binary input vectors for training. Hence, we need to map the data onto binary vectors.

For categorical attributes, we simply map each distinct attribute value X_{fi} where i *in* $AttributeValue_f$ onto a specific row in the CMM as given in equation 2, where the rows are indexed by integers ($\mathbb{Z}_{fk}$) and $\rightarrowtail$is a one-to-one mapping.

$$X_{fi} \rightarrowtail \left(\mathbb{Z}_{fk} + \text{offset}_f\right) \tag{2}$$

The integer $\mathbb{Z}_{fk}$ identifies the bit to set within the CMM input vector as in equation 3 and thus corresponds to a row in the CMM which will be active.

$$I' = I \oplus \left(\mathbb{Z}_{fk} + \text{offset}_f\right) \tag{3}$$

$offset_f$ is a cumulative integer offset within the binary vector I_i for each attribute f where $offset_{f+1} = offset_f + numberOfBins(f)$ and $x = y \oplus z$ sets the bit at location z in the vector y to produce a new vector x.

For real-valued attributes, we need to discretise the range of values. Our approach is to map the attribute values for each attribute *f* onto bins. Each bin maps to a unique integer $\mathbb{Z}_{fk}$ which indexes a specific row in the CMM as in equation 4. We then set the appropriate bit in the input vector as in equation 3.

$$\mathbb{R}_{fi} \rightarrow bins_{fk} \rightarrowtail \left(\mathbb{Z}_{fk} + \text{offset}_f\right) \tag{4}$$

where *i in AttributeValue(f)* and $|\mathbb{Z}_{fk}| \equiv |bins_{fk}|$ as each integer maps to a single and unqiue bin and $\rightarrow$ is a many-to-one mapping.

2) DISCRETISATION METHODS

We evaluate various unsupervised discretisation methods in this paper which vary the procedure for selecting the subdivision of the attributes (i.e. vary the bin boundaries). The methods evaluated are: equi-width discretisation (or fixed-width discretisation), optimised equi-width discretisation, equi-frequency discretisation (also called histogram equalisation), k-means clustering discretisation on all attributes together and on the individual attributes and expectation maximisation clustering on all attributes together and on the individual attributes.

Equi-width (EW) discretisation (Dougherty et al., 1995; Liu et al., 2002) aims to subdivide an attribute in to bins whose widths are distributed uniformly across the range of the attribute. The range of values is divided into *b* bins such that each bin is of equal width as in equation 5.

$$Width\left(bins_{fk}\right) = \frac{max(AttribueValues(f)) - min\left(AttributeValues(f)\right)}{b} \tag{5}$$

Optimised equi-width **(OW) discretisation** (Schmidberger and Frank, 2005) optimises the number of bins specified for equi-width discretisation using a leave-one-out cross-validation estimate of the log-likelihood. For each iteration of the leave-one-out cross-validation, the log-likelihood scores the set of bin boundaries generated. This allows the best set of bin boundaries to be determined. Log-likelihood is given in equation 6

$$Log\text{-}likelihood = \sum_i n_{i-test} * log \frac{n_i + \frac{w_i}{W}}{w_i * (N+1)} \tag{6}$$

In contrast, **equi-frequency (EF) discretisation** (Dougherty et al., 1995; Liu et al., 2002) used previously in AURA (Zhou, Austin & Kennedy, 1999; Weeks et al., 2003; Hodge & Austin, 2004b) aligns the bin boundaries so each bin contains an approximately equal number of records. The range of values for each attribute is divided into *b* bins each containing an approximately equal number of records. Thus, each row in the CMM will have approximately the same number of bits set – one for each record whose attribute value maps to the particular bin that the CMM row represents. There will be a larger number of bins where the attribute values are clustered and relatively few bins representing the outlying values. With respect to each attribute,

1) Sort all *N* data points into ascending order,

2) Find the maximum number of identical points *Ni* and thus estimate the number of distinct data values in each bin N_p as $N_p = (N - N_i)/b$ where *b* is the number of bins.

3) Set the uppermost boundary of each bin as the next data value in the sorted order.

4) Count the number of data values either side of N_p which are equal, and either include these points in the current bin or promote them to the next bin.

5) If the number of distinct values in the final bin N_d is greater than (N_p +b) then increase N_p by (N_d - N_p)/b and rerun the partitioning process from step 2 for that attribute.

We note that re-running the discretisation process (after step 5 above) for equi-frequency discretisation may increase or reduce the number of bins relative to the initial number specified. We vary the number of bins (b) for the discretisation techniques in our evaluation in section IV between 10 and 100 in steps of 10. In our empirical analysis in this paper, we always state the initial number of bins specified for the **EF** method.

Other discretisation methods evaluated here are based on clustering (Joita, 2010). Clustering algorithms partition the data space by searching the data for similar examples and grouping them into clusters such that the intra-cluster distances are small whereas the inter-cluster distances are as large as possible. **k-means clustering** (**KM**) introduced by MacQueen (1967) is one of the most popular clustering algorithms and can be used for discretisation. K-means firstly randomly selects a set of points called "seeds" which represent k cluster centres. The algorithm traverses the entire data set assigning each data point to its nearest cluster centre. When all data points have been assigned to their nearest cluster centre, the k cluster centres are recalculated as the *mean* of all of the data points in the cluster. Assignment and recalculation are repeated until the termination criterion is met. The algorithm terminates when convergence is achieved i.e. no further changes occur in the clusters. The most common distance measure used in k-means algorithm is the Euclidean distance, a special case ($p=2$) of the Minkowski metric. We run k-means on all data to generate the cluster centres using dynamic discretisation (**KM**) (Min, 2009; Joita, 2010). We place the bin boundaries as half way between the cluster centres. We also run k-means on the individual attributes to generate the cluster centres for each attributes using static discretisation (**KMInd**). Again, the bin boundaries are the half way points between the cluster centres.

Expectation-Maximisation (EM) introduced by Dempster, Laird & Rubin (1977) finds clusters by determining a mixture of Gaussians that fit a given data set. EM assigns a probability distribution to each data record which indicates the probability of it belonging to each of the clusters. Expectation refers to computing the probability that each data record is a member of each class; maximization refers to altering the parameters of each class to maximize those probabilities. EM can determine the number of clusters to create by cross-validation or the number maybe specified a priori. We specify the number of clusters here. Similar to the k-means evaluation, we run **EM** on all data to generate the cluster centres using dynamic discretisation. We place the bin boundaries as half way between the cluster centres to ensure that bins do not overlap and there are no gaps between bins. We also run EM on the individual attributes to generate the cluster centres for each attribute using static discretisation (**EMInd**). Again, the bin boundaries are the half way points between the cluster centres.

We use the implementations of the binning algorithms available in the WEKA Java data mining library (Hall et al., 2009) to determine the bin boundaries. We pass the number of required bins as a parameter to WEKA. All other parameter settings for the WEKA algorithms are at their default values.

2) INPUT VECTORS:

Once the bins and integer mappings have been determined, we need to map each record X onto a binary input vector I_j for the CMM. In this paper, each record is a multivariate time-series X_t as given in equation 7,

$$Xt = \{x_{11},x_{12},\ldots,x_{1t-1},x_{1t},x_{21},x_{22},\ldots,x_{2t-1},x_{2t},\ldots,x_{f1},x_{f2},\ldots,x_{ft-1},x_{ft}\} \quad (7)$$

for time $t = 1 .. T$ and attribute $f = 1 .. F$

Each attribute X_f maps onto a consecutive section of bits in the binary vector as in equations 2 and 3 for categorical attributes and equations 4 and 3 for continuous attributes.

Each concatenated binary vector represents a record from the data set and forms an input I_j to the CMM. The CMM associates the input with a unique output vector O^T_j during training, see equation 1. Each output vector is orthogonal with a single bit set corresponding to the record's position in the data set, the first record has the first bit set in the output vector, the second and so on. In effect, each column of the CMM represents a data record.

B. RETRIEVING THE NEAREST NEIGHBOURS

To recall the nearest matches for a query record, we first produce an input vector for the CMM as in equations 2 and 3 for categorical attributes or equations 4 and 3 for continuous attributes to identify the input vector bits to set. This vector replicates distance-based nearest neighbour retrieval. The vector may then be input to the CMM. The dot product of this vector and the trained CMM will produce an output vector indexing the k nearest matching records as described in the following sections.

For continuous attributes, we apply an integer-based parabolic kernel (Hodge & Austin, 2005) (as in figures 2 and 3) to the input vector which is analogous to quantised Squared Euclidean distance (see equation 8). It uses integer-valued vectors to input to the CMM and thus score the columns (records). Records with a high total column score are more similar to the input record than records with a low column score.

$$SquaredEuclidDist = \sum_{all\ f}\left(x_f - y_f\right)^2 \quad (8)$$

The Parabolic kernel value for each bin ($bins_{fk}$) in attribute f is given in equation 9 where $max(b)$ is the maximum number of bins across all attributes, $\left|\mathbb{Z}_{fk} - \mathbb{Z}_{ft}\right|$ calculates the offset (that is, the number of CMM rows between the index for the bin mapped to by the target attribute value $bins_{ft}$ ($\mathbb{Z}_{ft}$) and the index for the bin mapped to by $bins_{fk}$ ($\mathbb{Z}_{fk}$) as in equations 2, 3 and 4, $|bins_f|$ is the number of bins for attribute f. All kernels have the same maximum value $\left(\frac{\max(b)}{2}\right)^2$ to ensure no bias across the attributes. Note: this is also the maximum score for matching categorical attribute values to ensure no bias across all of the attributes, regardless of attribute type. We scale the kernel using α_f to spread the kernel across the range of each attribute in turn within the CMM input vector as attributes may have differing number if bins so we need to ensure that the parabolas representing the attributes are not biased, that is, the parabolas for all attributes have the same maximum value. The parabolic kernel is then superimposed onto the input vector as in equation 10 and shown in figure 4.

$$Parabolic_{bins_{fk}} = \left[\left(\frac{\max(b)}{2}\right)^2 - \left(\left(\left|\mathbb{Z}_{fk} - \mathbb{Z}_{ft}\right|\right)^2 \times \alpha_f\right)\right] \text{ where } \alpha_f = \frac{(\max(b))^2}{\left(|bins_f|\right)^2} \quad (9)$$

$$I'_j = I_j \oplus (Parabolic_{bins_{fk}} + \text{offset}_f) \text{ for all bins } (bins_{fk}) \text{ in all attributes } f \quad (10)$$

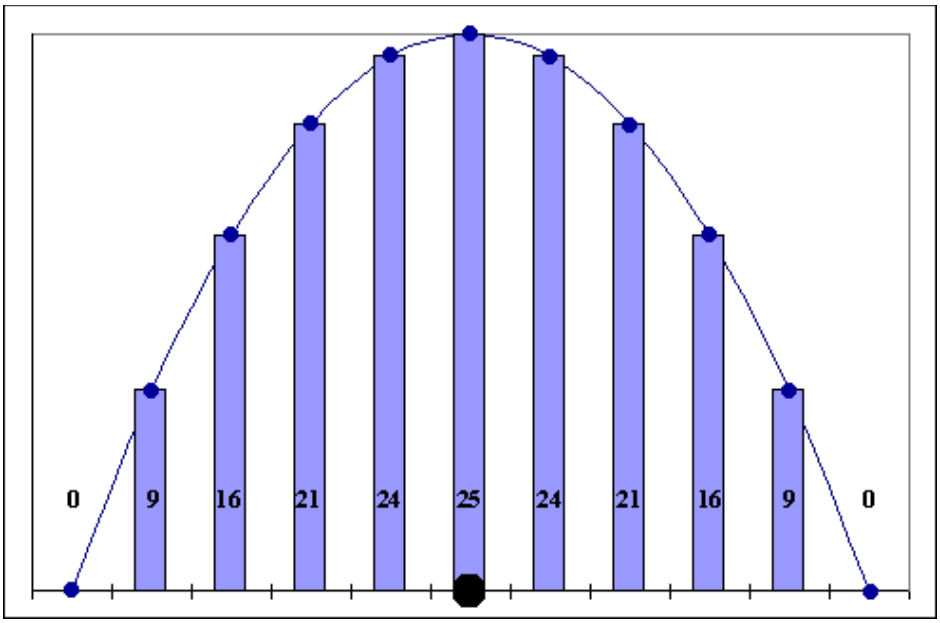
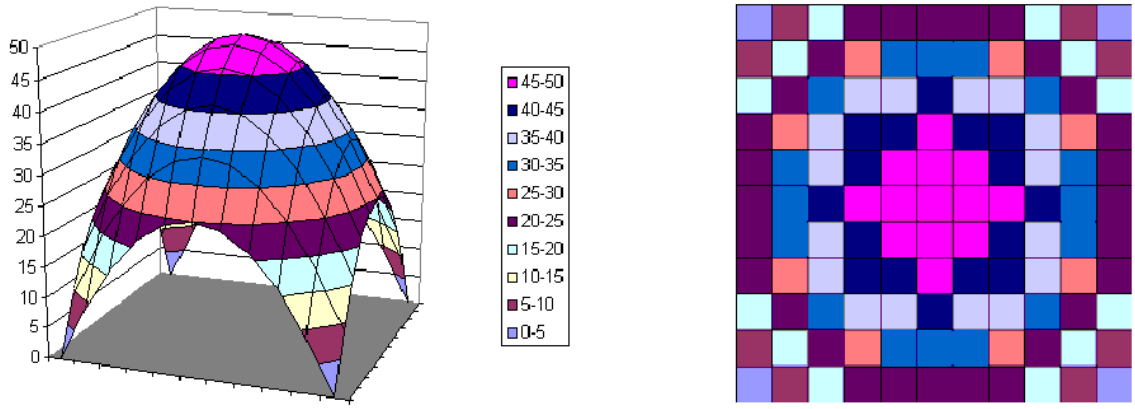


FIGURE 2 The input values (shown as bars) of the CMM rows are set to emulate the parabola (line) which represents the Euclidean distance from the central value (shown as large dot on central bar). The row input values (bar graph) are thus a discrete approximation of squared Euclidean distance.



(a) **(b)**

FIGURE 3 Figure a) shows the smoothed parabolic kernel intersection for a two attribute data with scores divided into ten discrete concentric regions. Figure b) shows the cumulative CMM column scores (representing the summed kernel intersections) for the AURA k-NN for the same two attribute data set with 11 bins per attribute and identical parabolas to figure 2 on both input attributes. The colours (scores) in the squares in b) match the banded colours on a) and represent the discrete concentric regions of equivalent score.

We move the kernels to match the input values unlike RBF (Bishop, 1995) where the kernels are fixed. The bin containing the query (target) value effectively receives the highest score with the score decreasing monotonically as the distance between the query value and a

bin increases. If the bin of the target value is offset, i.e. not the median bin, then the Parabola is offset and truncated at one end as in attribute2 of figure 4 where the Parabola is centred near the top and truncated at the top. If all attributes have an equivalent number of bins then the superimposed Parabolas will be identical. However, if the number of bins varies across the attributes, then the width of the Parabolas varies accordingly due to α_f in equation 9 spreading the kernel across the attribute width as shown in figure 4.

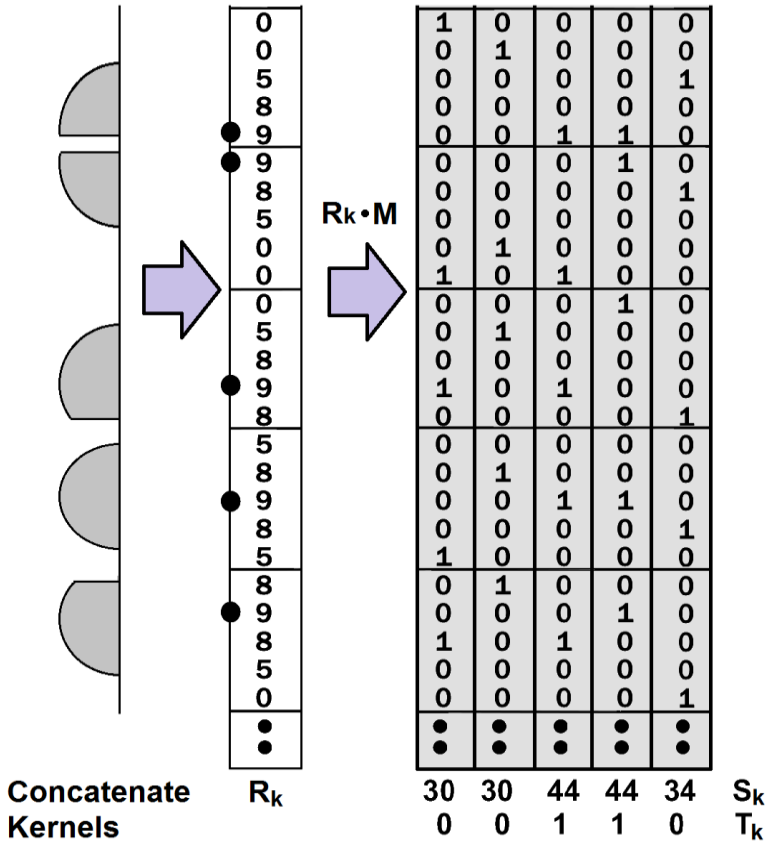


FIGURE 4 Diagram showing the application of kernels to a CMM to find the nearest neighbours. The left hand side illustrates the generation of the retrieval input vector *Rk* by applying kernels. The dot is the bin representing the query value for each attribute. AURA multiplies *Rk*M*, using the dot product, sums each column to produce the summed output vector *Sk* and thresholds the summed output vector to produce the thresholded output *Tk*.

For categorical attributes, we activate the single row matching the query attribute value and any records stored in the CMM will be activated and scored as shown in equation 11 where I_{jq} sets the *qth* element of the integer-valued input vector I_j. Note: all other elements in vector I_j will be 0.

$$I_{jq} = \left(\frac{\max(b)}{2}\right)^2 \tag{11}$$

1) *CMM RECALL:*

To retrieve the best matching records for a particular query record (represented by integer-valued input I_k) using Parabolic kernels, the AURA k-NN effectively calculates the dot product of the input vector I_k and the *CMM*, computing a positive integer-valued output vector O_k (the summed output vector) as in equation 12 and figure 4.

$$O^T{}_k = I_k \bullet M \quad (12)$$

The summed output O_k is thresholded to produce a binary output vector as in figure 4. We use the L-max threshold (Austin, 1995). L-Max thresholding essentially retrieves at least L top matches, i.e., at least L nearest neighbours. L-max thresholding sets a bit in the thresholded output vector for every location in the summed output vector that meets the criterion. For k-NN, L is set to the value of k, where k is the number of nearest neighbours required.

The method can identify the k-nearest matching records by inspecting the bits set in the thresholded output vector. In the work here, bit_0 in the output vector corresponds to the first record in the data, bit_1 to the second record and so on. Therefore, if bit_0 is set in the thresholded output vector then the first record is a match.

2) PREDICTION

To undertake prediction of the future values in the time series, we maintain a lookup table of values for the prediction attribute $t+n$ time steps ahead for each historical record. The values are indexed by the column indexes (integers) of the CMM. After AURA recall, the set of best matching columns are stored in the thresholded output vector T_k. For all columns set to 1 in T_k, we obtain the column index. The AURA k-NN cross-references the historical records from this set of column indices, sums the $t+n$ attribute values for all matching columns and calculates the mean value for the prediction attribute n time steps ahead.

IV. EVALUATION

Across all evaluations, we generate the discretisation cut point sets (binning partition) using the discretisation and clustering methods available in WEKA (Hall et al., 2009). These partition boundaries then form the boundaries to discretise the data for the AURA k-NN.

The AURA k-NN for prediction uses the AURA C++ class library (AURA, 2012) which provides classes and methods for CMMs and thresholding. The AURA k-NN is absolutely identical for all techniques evaluated to ensure consistency. We retrieve the top ten nearest neighbours for all evaluations. The number of neighbours to retrieve is a trade-off. Too few neighbours will cause the prediction to be affected by any erroneous values in the nearest neighbour set. Too many neighbours may mask any errors retrieved by averaging them out. To allow a thorough evaluation of the discretisation methods, we have chosen a value that will mask the odd discrepancy but will not mask repeated errors. This will allow a valid comparison of the discretisation methods. There is only one variation across data sets: we vary the time-series length according to the specifications of each dataset – the time series lengths used are given in Table 2 and are fixed across all evaluations of each data set.

Therefore, the only variation that we are evaluating is the discretisation of real-valued attributes.

A. DATA SETS

We evaluate the discretisation algorithms within our AURA k-NN framework using 10 well-known data sets from the UCR time-series data repository (UCR, 2012) and the UCI machine learning repositories (Frank & Asuncion, 2010). We detail the data sets below, indicating which attributes we used from the original data (we refer to the column number

indexing from 1) and the size of the training and test splits. We split the data sets chronologically by using the first *X* records as the training set and the final *Y* records as the test set to mimic the real-world scenario. We use all original data records where possible except for the Motes data where some of the later records contain large number of sensor errors which we omitted. For the Motes Q8 Temperature data set (Deshpande et al., 2004) we omit the final three columns as they contain large numbers of sensor errors. We only select the attributes recommended in the paper (Costa et al., 1999) for the Power Demand data set.

1. **CalIt2**: Observations from two data streams (people flow in and out of the building), over 15 weeks, 48 time slices per day (half hour count aggregates) (Frank & Asuncion, 2010).

2. **Inline Skating**: Activation of three muscles, foot contact signal, three angles and three angular velocities of professional inline speed skater during the last thirty seconds of a three minute exercise on a treadmill at 3.72 m/s sampled at 1kHz (Mörchen, Ultsch & Hoos, 2005).

3. **pH**: Simulation data of a pH neutralization process in a stirring tank (de Moor, 2012; McAvoy, Hsu & Lowenthal, 1972).

4. **Power Demand**: short-term load forecasting of power system data provided by AEM-Turin covering the three years (1995-1997) (Costa et al., 1999). The data attributes we use are transformed power load, minimum temperature (for the day) and maximum temperature (for the day) as recommended in the paper (Costa et al., 1999).

5. **Q8 Humidity**: Sensor motes: humidity sensor measurements collecting from wireless sensors (Deshpande et al., 2004).

6. **Q8 Temperature**: Sensor motes: temperature sensor measurements collecting from wireless sensors (Deshpande et al., 2004).

7. **Q8 Voltage**: Sensor motes: voltage sensor measurements collecting from wireless sensors (Deshpande et al., 2004).

8. **Stanford**: multivariate data set recorded from a patient in the sleep laboratory of the Beth Israel Hospital in Boston, Massachusetts. The first column is the heart rate, the second is the chest volume (respiration force), and the third is the blood oxygen concentration (measured by ear oximetry). The patient shows sleep apnoea (Weigend, and Gershenfeld, 1994).

9. **Steam Generation**: Model of a steam generator at Abbott Power Plant in Champaign IL. (Pellegrinetti & Benstman, 1996).

10. **Winding**: Data from a test setup of an industrial winding process (de Moor, 2012; Bastogne et al., 1997).

Further details of the datasets are given in Table 1 which lists the column indices of the attributes from the original data set that we used (indexed 1 .. *F*), the column index of the attribute we used for *t+1* prediction, how many records formed the training set and how many records formed the test set.

TABLE 1 Table listing the attributes selected (column indexes), the attribute used for prediction (column index), the number of records in the training data set and the number of records in the test data set for each of the ten data sets evaluated.

Dataset	Attributes	Predict	Train	Test
CalIt2	4,8	8	3360	1680
Inline Skating	2-11	2	19932	9968
pH	1-3	3	1265	736
Power Demand	5,7,8	5	19954	9976
Q8 Humidity	1-48	1	4332	2168
Q8 Temperature	1-53	1	3999	2001
Q8 Voltage	1-46	1	4867	2433
Stanford	1-3	1	22666	11334
Steam Generation	1-4	4	6400	3200
Winding	1-7	7	1666	834

B. EVALUATION

For each of the ten data sets, we ran each of the seven discretisation algorithms using all data with each of the bin counts (10-100 inclusive in steps of 10). We felt that a range of bins between 10 and 100 would provide a thorough analysis of the discretisation methods. This produces 10x7x10 (700) binning partition sets. Note: we use all of the data (train+test) for setting up the binning to ensure that no binning method is favoured by the data split. There may be bias if we only use the training data for bin selection, particularly for the smaller data sets and particularly as the data is time-series which may contain trends. We then ran each of the 70 binning partition sets for each of the ten data sets through the AURA k-NN in turn to predict the *t+1* attribute value for the selected prediction attribute for that data set. We used the set of attributes listed in table 1 for both the training and test data. We selected the 10 nearest neighbours to generate the *t+1* prediction for each of the records in the test set by averaging the *t+1* value for these 10 nearest neighbours.

The prediction accuracy is calculated as the Root Mean Square Error (RMSE) for all predicted *t+1* values against the actual value for that time slot across all records in the test set.

C. RESULTS

Table 3 to Table 12 in the appendix list the RMSE prediction accuracies for each of the seven binning algorithms with between 10 and 100 bins (inclusive in steps of 10) for each of the ten data sets. The data are also shown graphically in Figure 5 to Figure 14 in the appendix. The best discretisation technique versus number of bins combination is marked in grey shading and bold italic in Table 3 to Table 12. We also mark the second and third best combinations for each data set in bold font. This indicates whether one method is consistently best for each data set or whether the best method for the top three RMSE scores varies. We note that the optimised equi-width produced no partition boundary sets for the three Motes data sets (Q8 Humidity, Q8 Temperature and Q8 Voltage). These data sets have missing sensor values indicated by the value 0. We passed these data sets to **ALL** of the WEKA discretisation methods in their raw form so the 0-values were not marked as missing. We suspect that the 0-values prevented the optimiser working.

Table 2 in section V provides an overview of these results.

V. ANALYSIS

Table 2 details the characteristics of the various data sets coupled with the best discretisation technique and the optimum number of bins for each data set. The characteristics are:

- Atts – the number of attributes in total.
- Ints – the number of integer-valued attributes.
- Real – the number of real-valued attributes.
- µRange – the mean range of the attributes (sum of ranges of all attributes divided by the number of attributes).
- RangeP – the range of the attribute to be predicted.
- TSLen – the length of the time-series used for prediction
- DM – the best discretisation method for the data set of the seven evaluated (EW, OW, EF, KM, EM KMInd, EMInd).
- NumBins – the optimum number of bins for the data set.

These characteristics will allow us to analyse whether there is a correlation between particular data set characteristics such as the number of attributes or the type of attributes and the best discretisation technique or optimum number of bins.

TABLE 2 Table providing the statistics of each data set, the best discretisation technique for that data set and the best number of bins to use for that discretisation technique.

	Atts	Ints	Real	µRange	RangeP	TSLen	DM	NumBins
CalIt2	2	2	0	58	62	24	EMInd	50
Inline Skating	10	0	10	7.87	0.16	30	EW	100
pH	3	0	2	4.70	8.11	30	KMInd	10
Power Demand	3	1	2	94.6	214	12	OW	100
Q8 Humidity	48	0	48	62.4	50.6	30	EMInd	70
Q8 Temperature	53	0	53	49.7	28.9	30	KMInd	10
Q8 Voltage	46	0	46	2.80	2.75	30	EW	100
Stanford	3	2	1	33816.7	108.1	60	KMInd	10
Steam Generation	4	0	4	114.0	32.6	30	EW	40
Winding	7	0	7	6.84	8.19	30	EW	30

From Table 2, we can see that there is no single best discretisation technique nor is there a single best number of bins. Skubacz and Hollman (2000) concluded that there was no single best discretisation method for the classification task that they evaluated. We have studied the characteristics of the data sets listed in columns 2-7 of Table 2 and there does not appear to be a correlation between the data characteristics and the best discretisation technique or the best number of bins to use. The only obvious correlation is that, for the three data sets where KMind is the best discretisation method then the best number of bins is 10. However, we note

that on the Q8 Humidity data set, the optimum number of bins for KMInd is 100 so we would not expect the optimum number of bins to always be 10. This indicates that the best approach is to evaluate a number of discretisation techniques coupled with varying numbers of bins for each data set to be predicted.

We have also visualised the values of the individual attributes across the data sets to investigate whether there is a correlation between the distribution of the data and the best discretisation configuration. We examined the distributions of the individual attributes rather than the distribution of all data together as the discretisation using individual attributes has lower error than discretisation using all attributes. Again, there does not appear to be a correlation between the discretisation configuration and the data distribution. Three data sets have KMInd as the best discretisation method with 10 bins as the best number of bins. There is very little correlation between the data distributions of these with the two data sets (Q8 Temperature and Stanford) having a majority of attributes with Gaussian-shaped distributions although the Gaussian-shapes are offset and not centred on the centre value. The other data set (pH) has very different attribute distributions. Also, there are other data sets with similar attribute distributions to Q8 Temperature and Stanford and these other data sets had different discretisation configurations as the top performer.

Equi-width discretisation is the best discretisation technique for four data sets. KMInd is best for three and EMInd is best for two data sets. Optimised equi-width is the best discretisation technique for one data set. Neither equi-frequency nor either of the two dynamic clustering discretisation techniques (KM or EM) is the best for any data set. For the data sets where equi-width is best it also tends to have the top three scores (for three of the four winning data sets). For the data set where optimised equi-width is best it also has the top three scores. For the data sets where EMInd and KMInd are best respectively, they do not have the top three scores.

The optimum number of bins varies across the range of values with 10 and 100 both best for three data sets each. 30, 40, 50 and 70 are the other best bin counts for one data set each.

VI. DISCUSSION

The results indicate that there is no single best discretisation method or number of bins. This necessitates evaluating various discretisation configurations to determine the best method. From the ten data sets evaluated here, there is no obvious correlation between the data characteristics and the best configuration neither is there an obvious correlation between the data distribution and the best configuration. As the number of data sets evaluated builds as the method is used, it may be possible to infer heuristics for determining the best discretisation configuration from the data characteristics. These heuristics may not indicate a single best configuration but it may be possible to at least limit the configurations required for evaluation for a particular data set using that data set's characteristics.

Equi-width discretisation does not distort the bin widths so the hyper-grid formed by the bins in the *d*-dimensional data space will have hyper-cubic cells. This means that the Euclidean distances are preserved which may explain the highest recall consistency achieved by this technique. The approach may be considered distance-based and k-NN is a distance-based predictor; it generates predictions using the *k* nearest stored records. However, Catlett (1991) noted that equi-width discretisation may be distorted by outlying attribute values which skew the attribute range and thus the bin widths so this may explain the lower recall accuracy achieved on some data sets, most notably Q8 Humidity and Q8 Temperature. These

data sets have 0-valued entries for missing sensor values, Q8 Humidity has erroneous entries of -276 and other smaller negative values, Q8 Temperature also has erroneous sensor readings of 122.153 degrees and these will have distorted the ranges. Excluding the 0-values and erroneous high values, Q8 Humidity ranges between approximately 18 and 60; and, Q8 Temperature ranges between approximately 16 and 30. Hence, the range for Q8 Humidity rises from 18-60=42 to -276-60=336 when the sensors errors are included in the latter figures. The range for Q8 Temperature increases from 16-30=14 to 0-122=122 when the sensors errors are included in the latter figures. Q8 Voltage ranges between approximately 2.1 and 2.7 so the 0-values will have much less effect on the ranges for this data set and equi-width is the top performer for Q8 Voltage data set. Also, the attributes may not be equally relevant so the input space might not be isotropic, and distances may not vary with equal strength in all directions. Thus a discretisation technique that skews the distances may perform better than equi-width discretisation on such data sets where no attribute weighting is used for k-NN retrieval as in this evaluation.

For the Power Demand data set the optimised version of equi-width binning (OW) clearly outperforms the other methods. Optimised equi-width binning(OW) is also the second best method on the CalIt2 data set. These data sets (CalIt2 and Power Demand) are somewhat similar representing 30 minute and hourly readings respectively, having two and three attributes with similar ranges and predicting an integer-valued attribute. This indicates that it is important to include optimised equi-width in any discretisation evaluation as it can outperform the other approaches on some data sets though overall performance is erratic across these ten data sets.

The static clustering discretisation methods (KMInd and EMInd) are not distorted by outlying attribute values unlike equi-width discretisation. The hyper-grid formed by the bins in the *d*-dimensional data space by both KMInd and EMInd will have cells with varying widths across each dimension. The cells will be narrower where the clusters are dense allowing finer-grained differentiation and the cells will be wider where the clusters are less dense. This allows records to be distinguished and separated. KMInd is the best method for three data sets and EMInd is the best for two data sets. However, they are not as consistent as equi-width discretisation as they do not hold the top three accuracies for any data set.

Dynamic clustering discretisation which uses all attributes to place the clusters and equi-frequency discretisation do not perform best on any data set. Using all of the attributes together to place the cluster centres as in KM and EM methods does not produce good quality partition boundaries. Optimising across all attributes can distort the bin boundaries. Better boundaries are generated using the attributes separately. Equi-frequency discretisation may be considered a density-based technique. The hyper-grid formed by the bins in the *d*-dimensional data space will have cells with an equal number of records mapping to each cell across each dimension. The cells will be narrower allowing finer-grained differentiation where the records are most dense and the cells will be wider where the records are less dense; equi-frequency is density-based. However, it distorts the distances represented, they are not the Euclidean distances but density-based distances and, we posit, that this has adversely affected prediction accuracy.

It may be worth including the dynamic clustering and equi-frequency discretisation methods in any evaluations if time limits permit. However, if time available for evaluation is limited, focusing on the static discretisation techniques (KMInd and EMInd), equi-width (EW) and optimised equi-width (OW) is likely to be most profitable.

We have shown empirically in our previous paper (Hodge & Austin, 2005) that the AURA k-NN is four times faster with respect to time than the standard k-NN on data sets up to 200,000 records. We note that the evaluation in Hodge & Austin (2005) used 149 bins whereas we only use 10 -100 bins in this paper (see section IV). Hence, the speed gain for the AURA k-NN here over the standard k-NN would be higher with up to fifteen times fewer bins to process in the AURA k-NN here than in Hodge & Austin (2005).

VII. CONCLUSION

The results indicate that there is no single best discretisation method or number of bins. Skubacz and Hollman (2000) concluded that there was no single best discretisation method for the classification task that they evaluated. This demonstrates that, when using AURA k-NN for prediction, we will need to evaluate various discretisation configurations with respect to both the binning method and the number of bins to determine the best discretisation configuration for the particular data set.

In Hodge, Jackson & Austin (2011), we proposed optimising the data and algorithm settings of the AURA k-NN using, for example, genetic algorithms (Holland, 1975; Goldberg, 1989) or particle swarm optimisation (Kennedy and Eberhart, 1995; Kennedy and Eberhart, 2001) which have been used widely in the literature for optimisation problems. Optimising the discretisation settings would form part of this process. Within this optimisation process, we also proposed a meta-learner similar to Brazdil, Soares and Da Costa (2003). This would use AURA k-NN to store the results of the optimisations run previously and learn the best settings. The k-NN distance function would be based on various features of the dataset to allow the selection of the most similar historical settings, that is, the best settings to use for the current dataset. These best settings may then be used to bootstrap future optimisations and short-circuit the optimisation process. From the evaluations here, there was no obvious correlation between data features and discretisation settings. However, over a larger number of data sets, correlations between data features and discretisation settings that may be used in the meta-learner may appear.

ACKNOWLEDGMENT

This work was supported by the CMAC (Condition Monitoring on a Cloud) project which is funded by the UK Technology Strategy Board.

REFERENCES

Aleksander, I. and Albrow, R.1968. Pattern recognition with Adaptive Logic Elements. In IEE Conference on Pattern Recognition, pp. 68–74.

Aleksander, I., Thomas, W. and Bowden, P. 1984. Wisard: A radical step forward in image recognition, Sensor Review, pp. 120–124, 1984.

AURA: Advanced Uncertain Reasoning Architecture web pages, 2012. http://www.cs.york.ac.uk/arch/neural-networks/technologies/aura (accessed 02 May 2012).

Austin, J., 1995. Distributed associative memories for high speed symbolic reasoning. In, IJCAI '95 Working Notes of Workshop on Connectionist-Symbolic Integration: From Unified to Hybrid Approaches, R. Sun and F. Alexandre, Eds., Montreal, Quebec, pp. 87–93.

Austin, J., 1998. RAM-Based Neural Networks, ser. Progress in Neural Processing: 9. World Scientific Pub. Co., Singapore.

Austin, J., Brewer, G., Jackson, T. and Hodge, V.J. AURA-Alert: The use of binary associative memories for condition monitoring applications. In, Procs 7th Int'l Conf on Condition

Monitoring and Machinery Failure Prevention Technologies: (CM 2010 and MFPT 2010), Stratford-upon-Avon, England, 22-24 June, 2010. Vol. 1: pp. 699-711.
Austin, J., Davis, R., Fletcher, M., Jackson, T., Jessop, M., Liang, B. and Pasley, A., 2005. DAME: Searching Large Data Sets within a Grid-Enabled Engineering Application. Proceedings IEEE - Special Issue on Grid Computing, 93(3): 496-509, ISBN 0018-9219
Bastogne, T., Noura, H., Richard, A. and Hittinger, J.M., 1997. Application of subspace methods to the identification of a winding process. In: Proc. of the 4th European Control Conference, Vol. 5, Brussels.
Bishop, C.M., 1995. Neural networks for pattern recognition. Oxford University Press, Oxford, UK.
Bledsoe, W. and Browning, I. 1959. Pattern recognition and Reading by Machine. In, Proceedings of Eastern Joint Computer Conference, pp. 225–231.
Brazdil, P.B., Soares, C. and Da Costa, J.P., 2003. Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. Machine Learning, 50(3): 251-277.
Bubeck, S. and von Luxburg, U., 2009. Nearest Neighbor Clustering: A Baseline Method for Consistent Clustering with Arbitrary Objective Functions. Journal of Machine Learning Research, 10(10): 657-698
Catlett, J. 1991. On changing continuous attributes into ordered discrete attributes. In, European Working Session on Learning -EWSL91. LNAI 482, Y. Kodratoff, Ed. Springer Verlag: Berlin, pp. 164–178.
Costa, M., Pasero, E., Piglione, F. and Radasanu, D.,1999. Short term load forecasting using a synchronously operated recurrent neural network, International Joint Conference on Neural Networks, IJCNN '99, vol.5, pp.3478-3482.
Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1): 21–27.
Dasarathy, B. (Ed.), 1991. Nearest Neighbor (NN) norms: NN pattern classification techniques. IEEE Computer Society.
De Moor, B.L.R. (ed.), 2012. DaISy: Database for the Identification of Systems, Department of Electrical Engineering, ESAT/SISTA, K.U.Leuven, Belgium, URL: http://www.esat.kuleuven.ac.be/sista/daisy/, (accessed 02 May 2012).
Dempster, A P., Laird, N.M. and Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, vol. 39, pp. 1-38.
Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J.M. and Hong, W., 2004. Model-Driven Data Acquisition in Sensor Networks. VLDB: 588-599.
Dougherty, J., Kohavi, R. and Sahami, M., 1995. Supervised and Unsupervised Discretization of Continuous Features. In, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA, pp. 194–202.
Fayyad, U.M. and Irani, K.B.,1993. Multi-interval discretization of continuous valued attributes for classification learning. In, Proceedings of the 13th International Joint Conference on Artificial Intelligence, R. Bajcsy, (Ed.) Morgan Kaufmann, pp. 1022–1027.
Fix, E. and Hodges, J., 1951. Discriminatory analysis, nonparametric discrimination: Consistency properties. Tech. Report 4, USAF School of Aviation Medicine, Randolph Field, Texas.
Frank, A. and Asuncion, A., 2010. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning Addison-Wesley Pub. Co. ISBN: 0201157675
Györfi, L., Kohler, M., Krzyzak, A. and Walk, H., 2002. A distribution free theory of nonparametric regression, Springer Verlag, New York.
Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten I.H., 2009. The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.
Hodge, V., 2011. Outlier and Anomaly Detection: A Survey of Outlier and Anomaly Detection Methods. LAMBERT Academic Publishing, ISBN: 978-3-8465-4822-6.
Hodge, V., Austin, J., 2001. An Evaluation of Standard Retrieval Algorithms and a Binary Neural Approach. Neural Networks, 14(3): 287-303, Elsevier Science.
Hodge, V. and Austin, J., 2004a. A Survey of Outlier Detection Methodologies, Artificial Intelligence Review, vol. 22, no. 3, pp. 85–126.

Hodge, V. and Austin, J., 2004b. A High Performance k-NN Approach using Binary Neural Networks, Neural Networks, vol. 17, no. 3, pp. 441–458.

Hodge, V. and Austin, J., 2005. A Binary Neural k-Nearest Neighbour Technique. Knowledge and Information Systems, vol. 8, no. 3, pp. 276–292.

Hodge, V., Jackson, T. and Austin, J., 2011. Intelligent Decision Support using Pattern Matching. In, Proceedings of the 1st International Workshop on Future Internet Applications for Traffic Surveillance and Management (FIATS-M 2011), Sofia, Bulgaria, Oct 2011, pp. 44-54. ISBN:978-989-8425-87-4

Hodge, V., Krishnan, R., Austin, J. and Polak, J., 2010. A computationally efficient method for online identification of traffic incidents and network equipment failures. Presented at, Transport Science and Technology Congress: TRANSTEC 2010, Delhi, Apr. 4-7, 2010.

Hodge, V., Krishnan, R., Jackson, T., Austin, J. and Polak, J., 2011. Short-Term Traffic Prediction Using a Binary Neural Network. Presented at, 43rd Annual UTSG Conference, Open University, Milton Keynes, UK, Jan. 5-7, 2011.

Holland, J.H., 1975. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, University of Michigan Press.

Holte, R., 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, Machine Learning, vol. 11, pp. 63–91.

Joita, D., 2010. Unsupervised Static Discretization Methods in Data Mining, Revista Mega Byte, vol. 9.

Kennedy, J. and Eberhart, R., 1995. Particle Swarm Optimization. In, Proceedings of IEEE International Conference on Neural Networks. IV. pp. 1942–1948.

Kennedy, J. and Eberhart, R., 2001. Swarm Intelligence. Morgan Kaufmann. ISBN 1-55860-595-9.

Knorr, E.M. and Ng, R.T., 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. In, Proceedings of the VLDB Conference, New York, USA, pp. 392–403.

Kotsiantis, S. and Kanellopoulos, D., 2006. Discretization Techniques: A recent survey, GESTS International Transactions on Computer Science and Engineering, Vol.32(1):47-58

Krishnan, R., Hodge, V., Austin, J. and Polak, J. (2010a). A Computationally Efficient Method for Online Identification of Traffic Control Intervention Measures. Presented at, 42nd Annual UTSG Conference, University of Plymouth, UK: Jan. 5-7, 2010.

Krishnan, R., Hodge, V., Austin, J., Polak, J. and Lee, T-C. (2010b). On Identifying Spatial Traffic Patterns using Advanced Pattern Matching Techniques. In, Proceedings of Transportation Research Board (TRB) 89th Annual Meeting, Washington, D.C., Jan. 10-14, 2010. (DVD-ROM: Compendium of Papers).

Krishnan, R., Hodge, V., Austin, J., Polak, J., Jackson, T., Smith, M. and Lee, T-C. (2010c). Decision Support for Traffic Management. In, Proceedings of 17th ITS World Congress: (CD-ROM), Busan: Korea, Oct. 25-29, 2010.

Liu, H., Hussain, F., Tan, C.L. and Dash, M. 2002. Discretization: An Enabling Technique. Data Mining and Knowledge Discovery, 6:393-423

MacQueen, J, 1967. Some methods for classification and analysis of multivariate observations. In, Proceedings of the 5th Berkeley Symposium on Mathematics, Statistics and Probability 3 pp. 281–297.

McAvoy, T.J., Hsu E. and Lowenthal, S. 1972. Dynamics of pH in controlled stirred tank reactor, Ind. Eng. Chem. Process Des. Develop. 11, 71-78

Min, H., 2009. A Global Discretization and Attribute Reduction Algorithm Based on K-Means Clustering and Rough Sets Theory, *Second International Symposium on Knowledge Acquisition and Modeling, KAM '09.*, vol.2, pp.92-95, Nov. 30-Dec. 1.

Mörchen, F., Ultsch, A. and Hoos, O., 2005. Extracting interpretable muscle activation patterns with Time Series Knowledge Mining. International Journal of Knowledge-Based & Intelligent Engineering Systems.

Pellegrinetti G., and Benstman, J. 1996. Nonlinear Control Oriented Boiler Modeling - A Benchmark Problem for Controller Design, IEEE Tran. Control Systems Tech. Vol.4 No.1.

Quinlan, J.R., 1986. Induction of Decision Trees. Machine Learning, 1: 81-106.

Quinlan, J.R., 1992. C4.5 Programs for Machine Learning, San Mateo, CA: Morgan Kaufmann.

Schmidberger, G. and Frank, E., 2005. Unsupervised discretization using tree-based density estimation. In, Proceedings of the 9th European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05), Springer-Verlag, Berlin, pp. 240-251.

Skubacz, M. and Hollman, J., 2000. Quantization of Continuous Input Variables for Binary Classification. In Proceedings of the Second International Conference on Intelligent Data Engineering and Automated Learning, Data Mining, Financial Engineering, and Intelligent Agents (IDEAL '00), Springer-Verlag, London, UK, pp. 42-47

UCR Time-Series Data Archive, 2012. http://www.cs.ucr.edu/~eamonn/iSAX/iSAX.html (accessed 02 May 2012)

Weeks, M., Hodge, V. and Austin, J., 2002a. A Hardware Accelerated Novel IR System. In, Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (PDP-2002), Gran Canaria, Jan. 9–11. IEEE Computer Society, CA.

Weeks, M., Hodge, V. and Austin, J., 2002b. Scalability of a Distributed Neural Information Retrieval System. Presented at, HPDC-2002, 11th IEEE International Symposium on High Performance Distributed Computing. Edinburgh, Scotland, July 24–26.

Weeks, M., Hodge, V., O'Keefe, S., Austin, J. and Lees, K., 2003. Improved AURA k-Nearest Neighbour Approach. In Proceedings of IWANN-2003, International Work-conference on Artificial and Natural Neural Networks, Mahon, Menorca, Balearic Islands, Spain. June 3-6.

Weigend, A.S. and Gershenfeld, N.A., eds. 1994. Time Series Prediction: Forecasting the Future and Understanding the Past. Reading, MA: Addison-Wesley.

Wettschereck, D. 1994. A study of distance-based machine learning algorithms, Ph.D. dissertation, Department of Computer Science, Oregon State University, Corvallis.

Witten, I. and Frank, E., 1999. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann.

Zhou, P., Austin, J. and Kennedy, J. 1999. A High Performance k-NN Classifier Using a Binary Correlation Matrix Memory. In, Advances in Neural Information Processing Systems, vol. 11.

APPENDIX

Tables listing the prediction RMSE for each discretisation method (EF, EW, OW, KM, KMInd, EM and EMInd) for each number of bins (10-100) for the ten data sets. The best RMSE is shaded grey in bold-italic font. The second and third best are in bold font.

TABLE 3 Table listing the prediction RMSE for the various discretisation configurations on the CalIt2 data set.

CalIt2							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	4.924497	**4.890287**	4.95764	4.946287	4.922035	4.960346	4.994925
20	4.952212	4.920293	5.039857	4.960418	4.914853	4.936898	5.042166
30	4.996043	4.928377	5.021545	4.926668	4.985006	4.933097	4.956718
40	4.955113	4.934718	4.966828	4.921325	4.954037	4.923158	4.928758
50	4.966231	4.948845	4.900699	4.895965	4.937652	4.96301	***4.873144***
60	4.966231	4.917732	4.890981	4.906734	4.937652	4.981531	4.873144
70	4.966231	4.917935	**4.877423**	4.8975	4.937652	4.961059	4.873144
80	4.966231	4.938334	4.959215	4.941658	4.937652	4.907745	4.873144
90	4.966231	4.92796	4.96544	4.939677	4.937652	4.907626	4.873144
100	4.966231	4.929061	5.013031	4.948577	4.937652	4.934471	4.873144

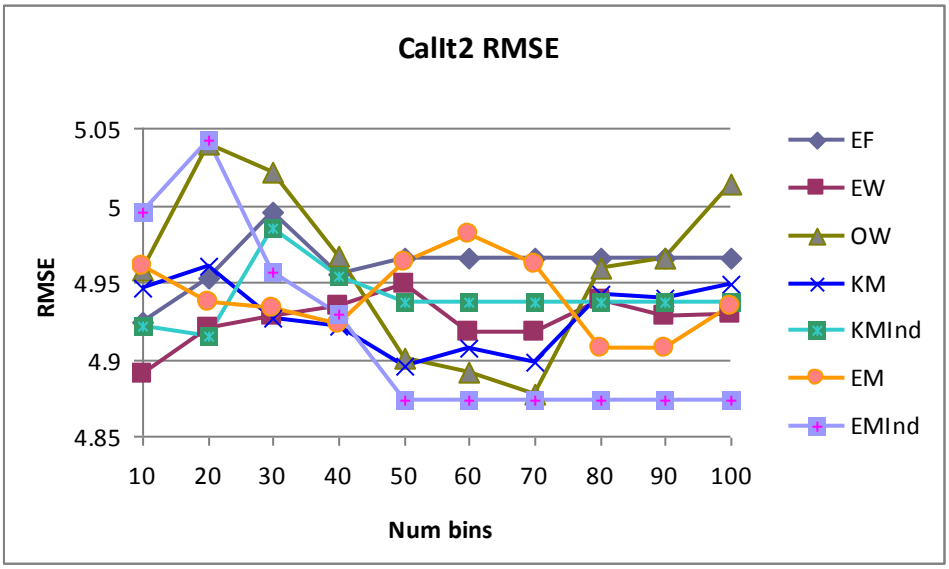


FIGURE 5 Graph of the RMSE data in TABLE 3.

TABLE 4 Table listing the prediction RMSE for the various discretisation configurations on the Inline Skating data set.

Inline Skating							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	0.005699	0.005264	0.005318	0.005848	0.005324	0.005688	0.005401
20	0.005525	0.004785	0.004939	0.005869	0.005036	0.005643	0.005043
30	0.005415	0.004715	0.004814	0.006096	0.005251	0.005851	0.004915
40	0.005421	0.004646	0.004682	0.006285	0.005146	0.006141	0.00497
50	0.005312	0.004633	0.004734	0.006456	0.005091	0.006188	0.004973
60	0.005363	0.004638	0.004857	0.006368	0.005034	0.006108	0.004869
70	0.005294	0.004644	0.004930	0.006420	0.005049	0.006227	0.004928
80	0.005317	**0.004606**	0.004911	0.006290	0.005034	0.006244	0.00494
90	0.005273	**0.004617**	0.004910	0.006262	0.005008	0.006182	0.004929
100	0.005276	***0.004595***	0.004788	0.006336	0.005059	0.005896	0.005046

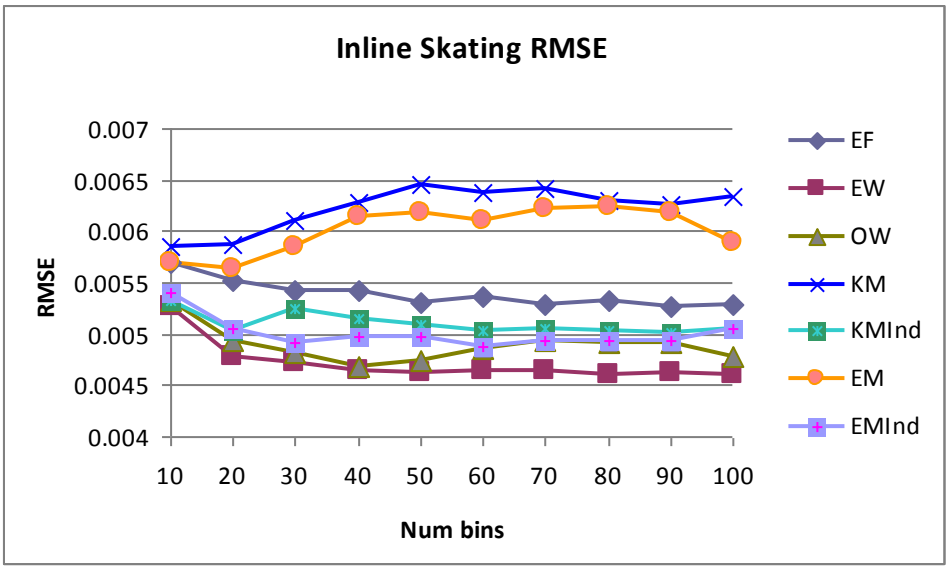


FIGURE 6 Graph of the RMSE data in TABLE 4.

TABLE 5 Table listing the prediction RMSE for the various discretisation configurations on the pH data set.

pH							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	1.172878	**0.651434**	1.364287	1.472774	***0.577938***	1.444895	0.911799
20	1.410466	**0.734924**	1.906532	1.537587	1.068633	1.333453	1.334506
30	1.65645	0.782993	1.896108	1.314685	1.378566	1.512411	1.527391
40	1.827855	0.796493	1.972796	1.535547	1.208107	1.317581	1.170691
50	1.885141	0.795941	2.014499	1.498622	1.590704	1.224661	2.013144
60	1.849427	0.816428	2.048418	1.461966	1.450132	1.469338	1.817294
70	1.851682	0.820291	2.035918	1.282719	1.433494	1.177819	1.650824
80	1.813855	0.823881	2.045153	1.399878	1.341892	1.220095	1.718529
90	1.888506	0.821751	2.079092	1.338068	1.540737	1.206869	1.676036
100	1.887283	0.828481	2.126529	1.329711	1.636054	1.30012	1.744316

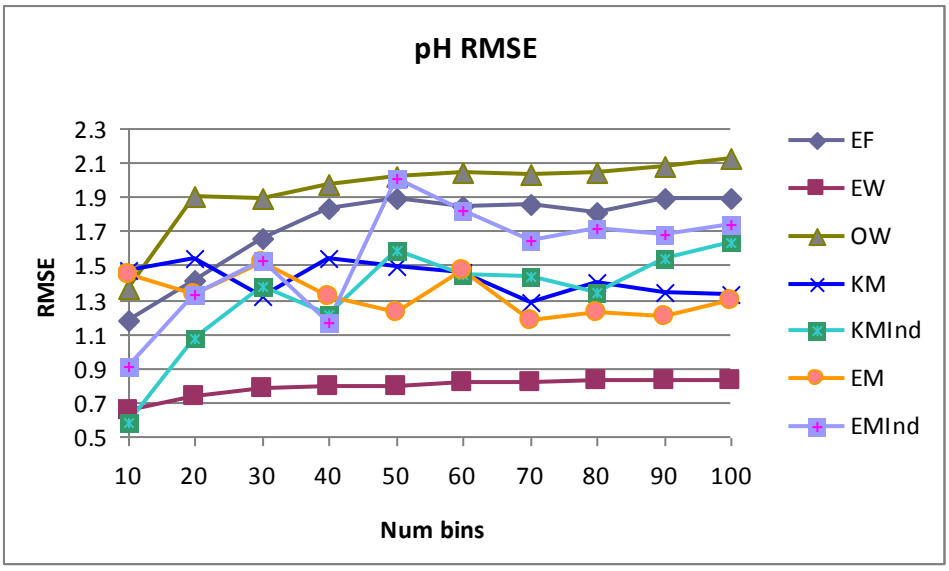


FIGURE 7 Graph of the RMSE data in TABLE 5.

TABLE 6 Table listing the prediction RMSE for the various discretisation configurations on the Power Demand data set.

Power Demand							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	11.91378	12.24336	12.63159	12.71695	11.19925	12.72386	11.47972
20	10.59119	10.93452	10.90303	10.80652	10.38338	10.82566	10.89315
30	10.52534	10.8376	10.99338	10.80449	10.55618	10.78955	10.20982
40	10.47906	10.79125	10.4197	10.71948	10.27625	10.7651	10.45479
50	10.52463	10.818	9.895694	10.53563	10.38274	10.54539	10.45545
60	10.44292	10.76346	9.597229	10.55794	10.46292	10.85642	10.64175
70	10.55007	10.7455	9.225207	10.65808	10.29918	10.70108	10.71359
80	10.47155	10.71229	**8.750793**	10.51056	10.33432	10.66725	10.59385
90	10.49693	10.75208	**8.585312**	10.59197	10.40026	10.5205	10.46287
100	10.42139	10.69797	***8.378631***	10.54727	10.37307	10.65517	10.53053

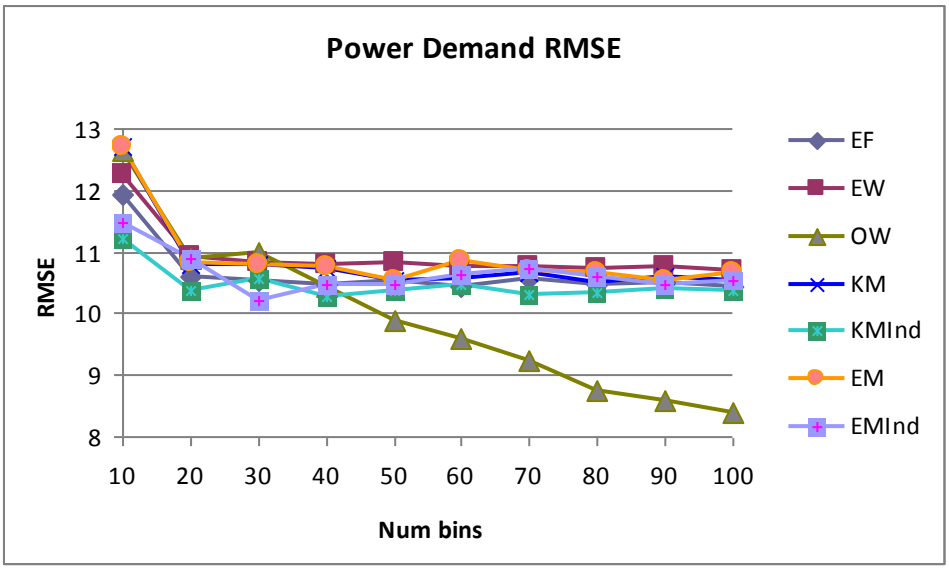


FIGURE 8 Graph of the RMSE data in TABLE 6.

TABLE 7 Table listing the prediction RMSE for the various discretisation configurations on the Q8 Humidity data set.

Q8 Humidity							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	5.539855	5.696871	-	5.574461	5.58411	5.654277	5.496418
20	5.485876	5.610919		5.504454	5.509741	5.482417	5.529952
30	5.478956	5.577556		5.506319	5.497571	5.457697	5.504945
40	5.473801	5.573197		5.49217	5.512809	5.463492	5.494317
50	5.474619	5.554312		5.467233	5.530864	5.46082	5.498137
60	5.47028	5.564743		5.500056	5.523913	5.476252	5.473664
70	5.472675	5.561528		5.49733	5.519527	5.47912	***5.449213***
80	5.474751	5.567025		5.48873	5.513559	5.481249	5.464963
90	5.476791	5.56056		5.507742	5.499874	**5.457313**	**5.456431**
100	5.47733	5.564607		5.488644	5.495294	5.465483	5.47779

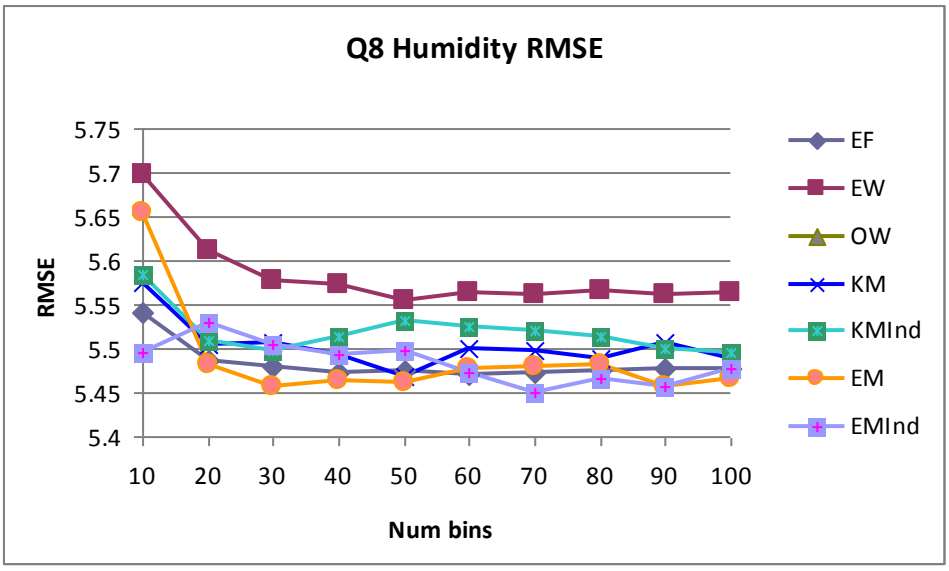


FIGURE 9 Graph of the RMSE data in TABLE 7.

TABLE 8 Table listing the prediction RMSE for the various discretisation configurations on the Q8 Temperature data set.

Q8 Temperature							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	2.859895	3.287342	-	2.842124	***2.76867***	2.825171	2.798231
20	2.818977	2.949413		2.810331	**2.772145**	2.822821	2.795002
30	2.812592	2.911362		2.804223	2.796941	2.79583	2.803491
40	2.812978	2.911264		2.809086	2.799589	2.821871	2.806346
50	2.811949	2.911923		2.81496	2.786565	2.818447	2.811924
60	2.8118	2.913309		2.811009	2.812517	2.784408	2.816255
70	2.812411	2.916016		2.813631	2.789157	**2.770904**	2.806368
80	2.812434	2.911724		2.809134	2.813404	2.787543	2.809671
90	2.812241	2.913893		2.807965	2.816527	2.775255	2.811235
100	2.812262	2.914636		2.81979	2.815964	2.803253	2.810602

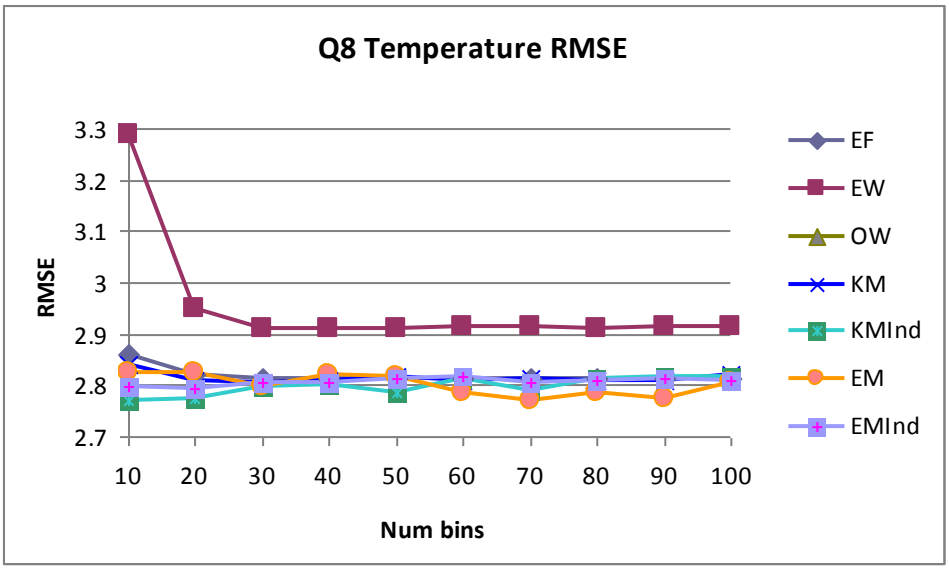


FIGURE 10 Graph of the RMSE data in TABLE 8.

TABLE 9 Table listing the prediction RMSE for the various discretisation configurations on the Q8 Voltage data set.

Q8 Voltage							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	0.450794	0.481753	-	0.509104	0.412714	0.459833	0.415563
20	0.455402	0.378643		0.51458	0.46287	0.519738	0.569271
30	0.582598	0.373808		0.521669	0.497748	0.581193	0.547806
40	0.559939	0.356845		0.572594	0.477481	0.620057	0.542968
50	0.55408	0.346293		0.547834	0.499776	0.634712	0.490699
60	0.583917	0.339062		0.535845	0.513607	0.616254	0.48559
70	0.578829	0.337866		0.570717	0.55089	0.630979	0.531651
80	0.588896	**0.335987**		0.551562	0.507068	0.587364	0.472054
90	0.557773	**0.336498**		0.551946	0.508434	0.615647	0.496174
100	0.558097	***0.335806***		0.591989	0.511198	0.624624	0.517107

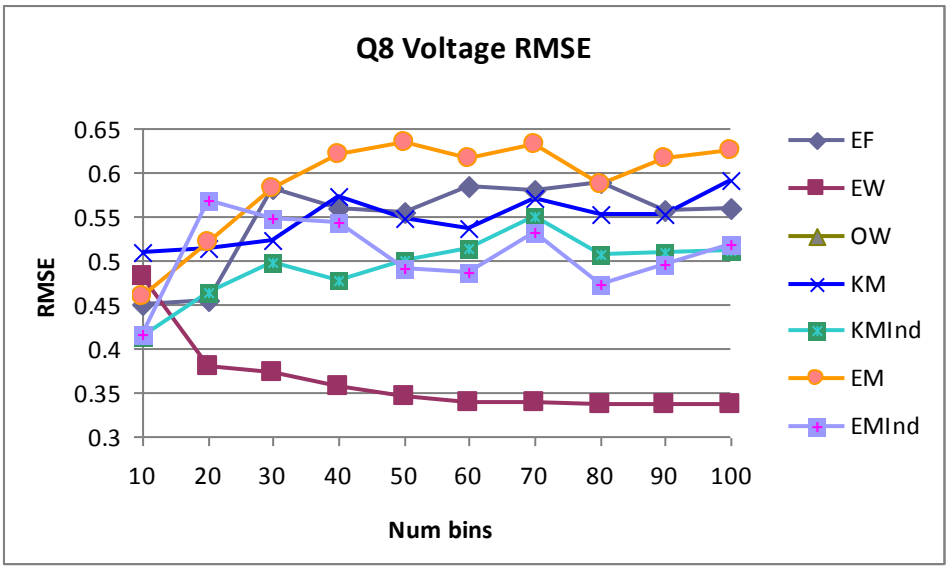


FIGURE 11 Graph of the RMSE data in TABLE 9.

TABLE 10 Table listing the prediction RMSE for the various discretisation configurations on the Stanford data set.

Stanford							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	16.09229	16.92937	16.71849	16.21269	***14.65761***	15.96104	**14.82125**
20	16.2657	16.7926	16.81843	15.76794	**14.74246**	16.0056	16.05704
30	16.30286	16.82134	16.85936	16.28071	15.01656	15.94739	15.9288
40	16.31204	16.74684	16.71774	16.20726	15.67695	16.74684	16.01587
50	16.3224	16.7538	16.75841	15.84172	16.25412	16.67528	16.30804
60	16.32028	16.78443	16.77114	15.71987	16.44666	15.71325	16.47258
70	16.32764	16.79053	16.73768	15.95333	16.62443	15.82765	16.58979
80	16.53062	16.74776	16.58009	16.06153	16.68895	16.48893	16.57137
90	16.68649	17.12461	16.69834	15.71757	16.67616	16.52897	16.59264
100	16.71821	17.12456	16.98707	15.56455	16.67605	16.60761	16.59462

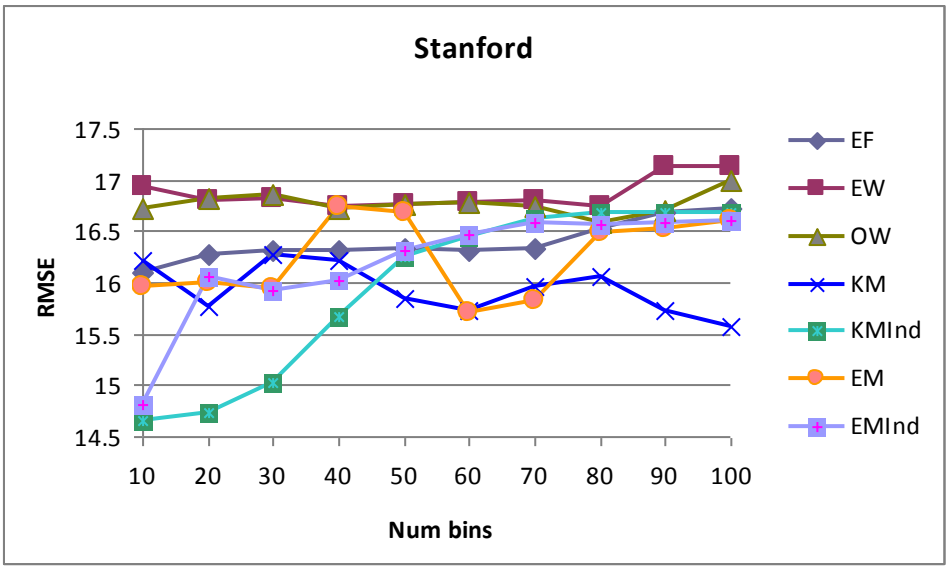


FIGURE 12 Graph of the RMSE data in TABLE 10.

TABLE 11 Table listing the prediction RMSE for the various discretisation configurations on the Steam Generator data set.

Steam Generator							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	2.820537	2.317172	2.363179	2.893302	2.560617	2.809536	2.713328
20	2.691419	2.312637	2.312637	2.936155	2.497025	2.750746	2.591133
30	2.671236	**2.291332**	**2.291698**	2.920642	2.529659	2.762968	2.578568
40	2.650103	***2.289425***	2.395625	2.801842	2.548329	2.757805	2.549705
50	2.658709	2.295461	2.345279	2.804712	2.574465	2.693802	2.623495
60	2.637555	2.293386	2.367243	2.775619	2.560677	2.745303	2.577679
70	2.623957	2.309085	2.360193	2.728743	2.601815	2.84196	2.640309
80	2.627001	2.301891	2.299493	2.75461	2.667130	2.886866	2.726628
90	2.635325	2.300801	2.422864	2.750975	2.710515	2.791833	2.697327
100	2.632024	2.305357	2.387626	2.684562	2.690951	2.844817	2.746271

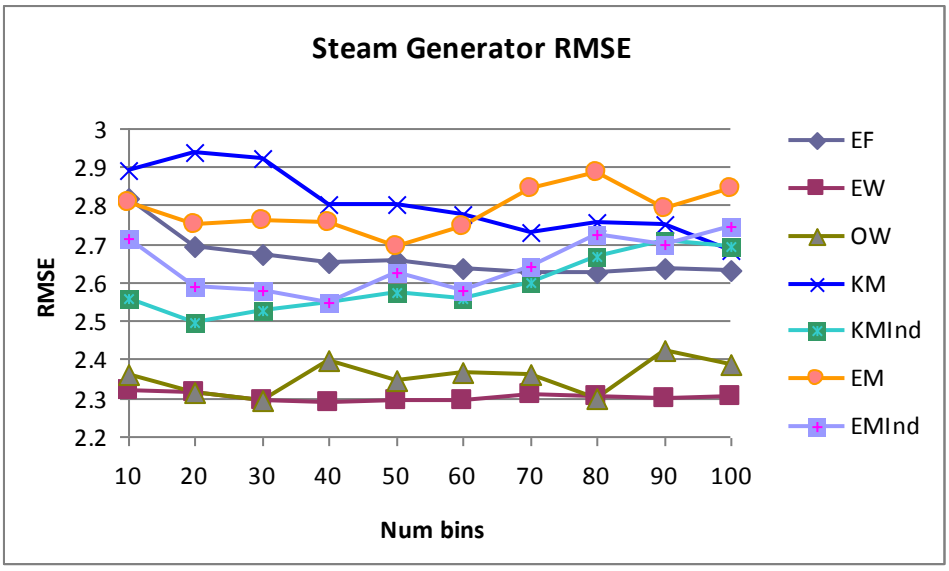


FIGURE 13 Graph of the RMSE data in TABLE 11.

TABLE 12 Table listing the prediction RMSE for the various discretisation configurations on the Winding data set.

Winding							
	EF	EW	OW	KM	KMInd	EM	EMInd
10	0.513782	**0.429671**	0.72545	0.545187	0.488177	0.527268	0.49026
20	0.506447	0.433126	0.69499	0.533119	0.47947	0.520153	0.472553
30	0.504079	***0.427338***	0.74488	0.529803	0.493573	0.50677	0.485893
40	0.502287	0.430681	0.781342	0.520284	0.50276	0.512498	0.489589
50	0.50024	**0.429751**	0.80811	0.518527	0.512097	0.505224	0.495685
60	0.501338	0.43078	0.837936	0.511035	0.512185	0.505768	0.492478
70	0.499693	0.430938	0.877855	0.507379	0.509961	0.510407	0.486573
80	0.501224	0.432308	0.886347	0.506368	0.513724	0.507807	0.487934
90	0.500129	0.430581	0.86298	0.50433	0.509512	0.505357	0.491668
100	0.501811	0.431722	0.886525	0.504464	0.510039	0.499687	0.488917

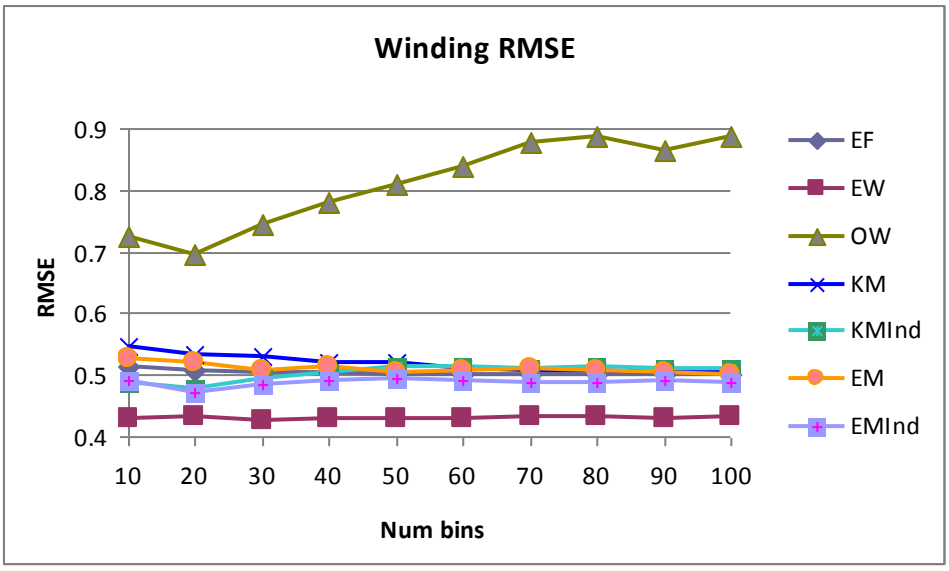


FIGURE 14 Graph of the RMSE data in TABLE 12.