Artificial Intelligence for Engineering Design, Analysis and Manufacturing

cambridge.org/aie

Margin-based approach for outlier detection of industrial design data using a modified general regression neural network

Jayaram Sivaramakrishnan, Gareth Lee, David Parlevliet and Kok Wai Wong

College of Science, Health, Engineering and Education, Murdoch University, 90 South Street, Murdoch, WA 6150, Australia

Research Article

Cite this article: Sivaramakrishnan J, Lee G, Parlevliet D, Wong KW (2022). Margin-based approach for outlier detection of industrial design data using a modified general regression neural network. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **36**, e7, 1–19. https://doi.org/10.1017/S0890060421000329

Received: 24 March 2021
Revised: 15 September 2021
Accepted: 6 October 2021

Keywords:
Decision support; design margin; GRNN; industrial design; machine learning; outlier detection; Parzen window

Author for correspondence:
Jayaram Sivaramakrishnan,
E-mail: j.sivaramakrishnan@murdoch.edu.au, jaya.sivaraman@hotmail.com

Abstract

The choice of components in industrial design involves setting design parameters that typically must reside inside permissible ranges called "design margins". This paper proposes a novel automated method called the Margin-Based General Regression Neural Network (MB-GRNN) that classifies design errors for design parameters that are outside of permissible ranges as outliers, directly from industrial design data, using an unsupervised machine learning approach. The method is based on a modified GRNN that estimates extremal margin boundaries of design parameters by self-learning the features from datasets. These extremal permissible margin boundaries are determined by "stretching out" the upper and lower GRNN surfaces using an iterative application of stretch factors (a second kernel weighting factor). The method creates a variable insensitive band surrounding the data cloud, interlinked with the normal regression function, providing upper and lower margin boundaries. These boundaries can then be used to determine outliers and to predict a range of permissible values of design parameters during design. Pushing out extremal margin boundaries reduce the false identification of outliers. This classification technique could be used by industrial engineers to detect likely outliers and to predict a range of permissible output limits for chosen design parameters. The efficacy of this method has been validated against the widespread Parzen window method by comparing experimental results from three multivariate datasets. It was found that the two methods have different but complementary capabilities. The MB-GRNN also uses a modified algorithm for estimating the smoothing parameter using a combination of clustering, k-nearest neighbor, and localized covariance matrix.

Introduction and literature review

Design margins

In industrial design, design parameters are rarely set to the exact theoretically required values; rather, they often have a margin that might be chosen deliberately or as a consequence of other design decisions. Experienced engineers include such margins in a product design process to serve several purposes. At the outset of the design process of any complex system, requirements, constraints, and capabilities are often uncertain. These uncertainties result from engineering change or for accommodating product flexibility and future growth (Eckert and Isaksson, 2017). To cover these uncertainties, designers tend to add margins to the design parameter values based on their past experience to cover realistic requirements and later firm up these values with analysis and testing (Eckert *et al.*, 2019).

The design of large projects involves collaboration from multiple teams working in parallel to tight schedules and budget constraints. The design process typically involves multiple preplanned iterative revisions to gather inputs and integrate results from the parallel teams (Wynn and Eckert, 2017). During these iteration rounds, key requirements and key parameters can change significantly, forcing other teams to accommodate these changes, and hence, a margin is added to protect against design uncertainties (Eckert and Isaksson, 2017). In collaborative complex system designs, these margins are estimated in negotiation between design teams (Austin-Breneman *et al.*, 2015). These margins provide a certain level of isolation between design parameter choices so that a minor change to one design parameter does not force an immediate review of every other possible independent design parameters. Examples of such design parameters with lower and upper margins are: power, pressure, level, temperature, pH, composition, and flow rate.

Design margins also play a critical role in safely managing engineering change and iteration. Designers and engineers define upper and/or lower margins to form an envelope called the "Safe Design Envelope (SDE)" for the chosen equipment (Eckert *et al.*, 2019; Stauffer and Chastain-Knight, 2019). These envelopes establish Safe upper and lower Operating Limits (SOLs) for critical operating parameters, beyond which there is a risk of catastrophic equipment failure. SOLs aim is to prevent process safety incidents and must be identified and documented for effective management of safety (Dowell, 2001; Richardson, 2012; Forest, 2018).

The Center for Chemical Process Safety (Guidelines for safe automation of chemical processes 1993) advise multiple layers of protection are used to protect people, property, and the environment from the hazards of chemical processing. This leads to a formal design approach entitled Layers Of Protection Analysis (LOPA; Willey, 2014) that ensures safety systems are multiply redundant. The LOPA design philosophy requires that critical parameters have a margin defined during design, referred to as normal operating limits, safe operating limits, instrumentation limits, and equipment containment limits (Stauffer and Chastain-Knight, 2019).

Buchan and Bolton pointed out that system reliability and safety depend on an inherently safe design (ISD; Pandian *et al.*, 2015), and this requires that a high degree of rigor is applied to design and fault analysis (Buchan and Bolton, 2009). Active systems rely on their ability to detect a system moving out of its safe envelope, whereas passive systems provide more safety and reliability by applying inherently safer design at the earliest stages of the project design process. The passive features generally involve less components that are easier to characterize and whose design parameters form a safe operating envelope. One challenge is to ensure that these design limits are fully understood and complied with during the system's lifecycle. Design margins are not only applied in chemical processing but also in many other areas such as power generation (Sutherland, 2011), jet engine design (Eckert *et al.*, 2019), civil engineering (Buchan and Bolton, 2009), and nuclear power plants (Modarres, 2009). Setting insufficient margins for critical parameters can result in catastrophic accidents such as the collapse of [in]famous Tacoma Narrows Bridge (Buchan and Bolton, 2009), whereas overestimating margins results in overdesign and higher build costs. Eckert also argues that awareness of margins during the design process can lead to more efficient management of change processes (Eckert and Isaksson, 2017).

Multiple factors contribute to these margins. The operation of process plants or their components results in hidden variables not captured in the design dataset. For instance:

- A fluid may have frictional energy loss as it moves along a pipeline changing its pressure and/or temperature properties in accordance with some (hidden) distance variable.
- Pumping of fluids may change its pressure property in accordance with some (hidden) pump capacity variable.
- The fuel efficiency of engines may vary in accordance with some (hidden) cylinder diameter variables.

General regression neural network

Industrial design outcomes are influenced by a combination of design parameters of single or multiple dimensions. Determining the correlation between dependent and independent variables for a system or process can be difficult. Traditional approaches have limitations when modeling complex nonlinear systems. Artificial Neural Networks (ANNs) have been found to solve such tasks better in many practical applications (Aglodiya, 2017).

The General Regression Neural Network (GRNN) is a term coined by Donald F. Specht (Specht, 1991), for a family of ANNs derived from the statistical technique of Nadaraya–Watson kernel regression (Nadaraya, 1964; Watson, 1964). GRNNs are single-pass associative memory feed-forward type ANNs and are generally used for function approximation. GRNNs have been widely used for approximation, fitting, prediction, and regression problems in the industry for fault detection and diagnosis (He and He, 2011; Baqqar *et al.*, 2012; Li *et al.*, 2018; Niu *et al.*, 2019), batch processing and system identification (Kulkarni *et al.*, 2004; Ou and Hong, 2014; Al-Mahasneh *et al.*, 2018), building systems monitoring (Kim and Katipamula, 2018; Mohandes *et al.*, 2019), and in other areas of engineering (May *et al.*, 2004; Cigizoglu and Alp, 2006; Pal and Deswal, 2008; Kumar and Malik, 2016; Lee *et al.*, 2018; Jagan *et al.*, 2019). The GRNN algorithm is efficient, and it can learn high-dimensional mappings in a single pass and results in acceptable performance even with a tightly constrained training set (Specht, 1991). Furthermore, it has only a single tuning parameter to achieve function smoothing, unlike the Back Propagation Neural Network (BPNN) and Support Vector Regression (SVR) that require multiple parameters to be tuned, needing more computational time and training data (Celikoglu and Cigizoglu, 2007; Zhang *et al.*, 2012; Jiang and Chen, 2016). The estimation of the GRNN smoothing parameter also has the advantage that it is not prone to local minima, unlike other methods such as BPNNs (Pal, 2011; Niu*et al.*, 2017).

Outlier detection

Outlier detection (Hodge and Austin, 2004; Singh and Cantt, 2012; Zimek *et al.*, 2012; Patil and Chouksey, 2016; Ge *et al.*, 2017; Xu *et al.*, 2018) has been gaining prominence as an area of machine learning (Ge *et al.*, 2017; Xu *et al.*, 2018). In other published work, supervised, semi-supervised, and unsupervised methods have all been used for outlier detection (Hodge and Austin, 2004; Singh and Cantt, 2012; Xu *et al.*, 2018). Among the unsupervised methods, two commonly used approaches to detect outliers are: statistical models to estimate a probability density function (p.d.f) using methods such as Parzen window (Parzen, 1962; Mussa *et al.*, 2015; Wang *et al.*, 2019); and ANNs that establish the relationship between variables as a regression model such as the GRNN (Specht, 1991; Kartal *et al.*, 2018; Wang *et al.*, 2019). Unsupervised methods can learn directly from industrial design datasets, offering clear economic benefits, and use these models to validate data by detecting outliers. Outlier detection has become a field of interest for many practitioners of fault detection and diagnosis (Zimek *et al.*, 2012; Xu *et al.*, 2018; Blazquez-Garcia *et al.*, 2020). This paper investigates two different methods for outlier detection using a modified GRNN and the long-established Parzen method.

A new method for outlier detection

Most previous research work uses the GRNN algorithms to derive a single regression model either to be used as a prediction tool or to estimate the accuracy of data. An extension from this approach is presented in this paper using a novel method called the Margin-Based GRNN (MB-GRNN). It uses a modified GRNN for the estimation of upper and lower regression boundaries (also called *margin boundaries*) by deriving three interlinked regression models from the same dataset. These margin boundaries, derived directly from data, can be used as a classifier to detect outliers or to determine the maximum and minimum values of dependent variables for any given design parameters. To obtain the best estimate of these margin ranges, the model must achieve the best possible generalization of the data and stretch out the extremal margin boundaries to the appropriate extent. To achieve the best generalization, the leave-one-out cross-validation (LOOCV) method (Specht, 1991) is applied using a more systematic approach to derive features from data involving clustering, k-nearest neighbor, and individual covariance. The motivation for adding stretching factors to a GRNN comes from papers that have weighted the terms of a GRNN to achieve computational gains (Lee and Zaknich, 2015) and the weighted version of Nadaraya–Watson estimator to determine a conditional mean (Cai, 2001). The MB-GRNN method builds on these approaches by adding stretch factors as a "second kernel factor" to the GRNN to iteratively inflate the extremal margin boundaries. The GRNN has not been previously used in this way to determine the design margins by stretching the regression surfaces to extremes of the range of typical values, so that any atypical combinations located outside of the regression surfaces can be identified as potential outliers. These outliers may be indicative of a design fault or data entry error and warrant further investigation by the design engineers.

The investigation uses both single and multidimensional sample domains that display margin characteristics and use an unsupervised approach to learn from data without the need for specific knowledge of the function of components. Two methods to detect outliers are assessed here, contrasting their performance on this class of problems. The first probabilistic method applied uses a clustering (Berkhin, 2002) algorithm to find a set of neighboring vectors and uses Parzen window (Parzen, 1962) to estimate p.d.fs comprised of standard Gaussian kernels with optimized parameters. The second method is the MB-GRNN regression algorithm, developed by applying stretch factors as a second kernel term to the GRNN to detect outliers of data. In each case, optimum generalization is achieved by applying a "smoothing parameter" using the LOOCV method.

Computational methods for the proposed MB-GRNN

The process of learning a model by induction from a set of training samples involves developing an approximation to a multivariate function. The function learned can be a probability density, if a method such as Parzen window estimation (Parzen, 1962) is used, whereas when the range is not a probability, a regression approach such as the GRNN can learn the expected value.

The GRNN learns from a training dataset and generates a "best-fit" nonlinear regression surface. It exhibits excellent approximation to arbitrary functions having inputs and outputs taken from sparse and noisy sources and can achieve optimal performance by adjusting a single smoothing parameter (Specht, 1991; Islam *et al.*, 2017). The GRNN is a very long-established method, and it uses the summation of a large number of terms. Others have looked at the idea of whether these terms can be weighted (Cai, 2001; Lee and Zaknich, 2015), whereas the focus of this paper is to examine whether these weightings can be used as factors to establish the upper and lower margins enclosing a data cloud. The sections below explain the computational methods used. Details of the algebraic notations used in formulas are described in Table A.1.

Parzen window estimator

Parzen window density estimation (Parzen, 1962) is a nonparametric estimation method, which can estimate the joint probability density function $p(\boldsymbol{x}, y)$ of multivariate data evaluated at $\boldsymbol{x} \in \mathcal{R}^d$ and $y \in \mathcal{R}^1$, based on a set of samples S_{train}, comprised of N vectors $\{\boldsymbol{c}_i\}$, where each $\boldsymbol{c}_i \in \mathcal{R}^{d+1}$ and $\boldsymbol{c}_i = (\boldsymbol{x}_i, y_i)$. The p.d.f is an estimation of the average density in proximity to $(\boldsymbol{x}, y)$. The standard equation for a Parzen window estimator is defined as:

$$p(\boldsymbol{x}, y) = \frac{1}{N}\sum_{i=1}^{N} \phi((\boldsymbol{x}, y)|\boldsymbol{c}_{i,}\Sigma_i), \tag{1}$$

where $\phi((\boldsymbol{x}, y)|\boldsymbol{c}_i,\Sigma_i)$ is a kernel function of $\boldsymbol{x}$ parameterized by a central location $\boldsymbol{c}_i$ and covariance matrix Σ_i. A set of training examples S_{train} provides the central locations for the kernel functions in Eq. (1).

The model is continuous over the domain $\mathcal{R}^{d+1}$ for the given training sample S_{train} and a kernel function $\phi(\boldsymbol{x}, y)$. The implementation uses a Gaussian kernel function with $\boldsymbol{x}$ treated as a row vector:

$$\phi((\boldsymbol{x}, y)|\boldsymbol{c}_{i,}\Sigma_i) = \left[\frac{1}{\sqrt{(2\pi)^d|\Sigma_i|}}\right]\exp\left(\frac{-D^2((\boldsymbol{x}, y), \boldsymbol{c}_i|\Sigma_i)}{2}\right). \tag{2}$$

And the Mahalanobis distance D (De Maesschalck *et al.*, 2000) is defined as:

$$\mathrm{D}(\boldsymbol{A}, \boldsymbol{B}|\Sigma) = \sqrt{(\boldsymbol{A} - \mathbf{B})\Sigma^{-1}(\boldsymbol{A} - \mathbf{B})^T}. \tag{3}$$

The distance $D_i((\boldsymbol{x}, y),\boldsymbol{c}_i|\Sigma_i)$ is measured between a vector $(\boldsymbol{x}, y)$ and $\boldsymbol{c}_i$, the central location of a kernel, weighted by Σ_i. When $\boldsymbol{x}$ is multidimensional, the Mahalanobis metric provides elliptical equidistance boundaries with covariance Σ determining the major and minor axes. D can be used to determine the spread of points forming a cluster from its central point if the covariance is used as the normalizing measure (De Maesschalck *et al.*, 2000).

Covariance determination

Covariance, in the statistical sense, is a measure of the correlation between two variables (Dempster, 1972). In a multidimensional domain, the covariance matrix $\Sigma_i \in \mathcal{R}^{dXd}$, captures the correlation between each pair of the dimensions.

The Parzen estimator uses a covariance matrix Σ_i that just considers the set of neighbors S_i in proximity to each $\boldsymbol{c}_i$ and the method applied was to calculate the covariance matrix for the set of n nearest neighbors (Cover and Hart, 1967). The covariance matrix is calculated for vector c_{ij} using:

$$\Sigma_{ijk} = \frac{1}{n}\sum_{p\in S_i} (\boldsymbol{c}_{pj} - \mu_j)(\boldsymbol{c}_{pk} - \mu_k), \tag{4}$$

where $\mu_k = 1/n \sum_{p\in S_i,} \boldsymbol{c}_{pk}$ is the mean and n is the number of elements in the neighborhood set S_i and (j, k) are each in the range $[1,d]$ and select an element within the Σ_i matrix. The additional variable p is an index used to select the subset of $\boldsymbol{c}_i$ values that are members of the neighbors set S_i.

Equation (4) takes into account the local variation of density in a region around c_i. The choice of n specific to each training dataset is critical in determining the Mahalanobis distance metric, as covariance acts to scale the Mahalanobis distance. The covariance needs to be further scaled to provide a smooth generalization between neighbors, and this must be done as a separate optimization exercise for each training dataset using a scaling factor. The neighborhood set S_i is determined for each n chosen from a range of values and the optimum scaling factor is determined for each n as shown in Figure 3.

Scaling factor

The covariance calculation described in Eq. (4) represents the distribution of n neighboring points around a mean μ. However, it alone does not ensure smooth interpolation between neighbors. The covariance matrix can be further optimized by determining a scalar multiplier s (referred to as the scaling factor). The scaling factor is determined using the LOOCV method that involves omitting one sample at a time and constructing the network based on the remaining samples. This network is then used to predict the probability value of the omitted sample, and this is averaged (Xu *et al.*, 2013) by iterating over the entire set. The purpose of this method is to find the solution that maximizes the average probability of samples using the Parzen estimate as shown in Figure 2 (or to develop a regression model for a known variable by minimizing mean squared errors using the GRNN model as shown in Fig. 3). The scalar factor s is then multiplied by the individual covariance Σ_i to give the optimum model fit for both Parzen and GRNN models.

The scaling factor is obtained by multiplying the covariance by a range of scaling values and iteratively repeating the above process to find the best factor s that maximizes the average probability (or minimizes the mean squared error) of samples for the entire distribution. The optimization process is shown in Figure 1 and the section "Smoothing parameter determination for Parzen and GRNN methods (Refer Step 3 in Fig. 1)".

To avoid making an arbitrary choice regarding the number of neighbors n that influence the covariance, the scaling factor s can be determined for a range of n and the optimum solutions chosen as shown in Figures 2 and 3, to generate the model. It was found experimentally that, where vectors have very nonuniform distributions, choosing the n nearest neighbors may not give the desired width for the covariance and to avoid this, a vector quantization algorithm attributed to Linde, Buzo, and Gray (LBG; Linde *et al.*, 1980) was applied as shown in the section "Pseudocode for data preprocessing and clustering to remove near duplicates (Refer Steps 1 and 2 in Fig. 1)". This was used as a preprocessing step to find the optimum number of clusters z and these clusters were used in place of the neighboring samples n. This method aims to eliminate duplicates and minimize the influence of closely spread points (near duplicates) when determining covariance.

The LOOCV method can be applied again to determine the optimum number of neighbors n for a given constant s, resulting in minimizing the mean squared error (MSE) or maximizing the average probability (MAP) for the given sample set are shown in Figures 2 and 3, respectively.

GRNN model

The value of a dependent variable y is predicted by a GRNN using the joint p.d.f estimated from known training data. The resulting regression equation can be implemented in a parallel neural network structure to predict the expected value of y for a given $\boldsymbol{x}$, using (Specht, 1991):

$$E[y\,|\,\boldsymbol{x}] = \frac{\int_{-\infty}^{+\infty} y\phi(\boldsymbol{x}, y)dy}{\int_{-\infty}^{+\infty} \phi(\boldsymbol{x}, y)dy}, \tag{5}$$

where $\phi(\boldsymbol{x}, y)$ is the joint continuous p.d.f. In practice, the joint continuous p.d.f can be empirically estimated using Parzen window estimation from a finite set of training examples. As this approach is nonparametric, the model parameters are derived directly from the training set using kernel functions. Rectangular and Gaussian kernel functions are commonly used for multidimensional datasets (Cristianini and Shawe-Taylor, 2000). The GRNN determines the most probable value of y for a given value of $\boldsymbol{x}$ as a prediction, whereas Parzen determines a joint probability $p(\boldsymbol{x}, y)$.

Specht redefined the radial basis function model as an interpolator to give $y = f(\boldsymbol{x})$ that is related to Parzen window. He also observed that the GRNN could learn from training datasets of the form, $S_{\text{train}} \equiv \{(\boldsymbol{c}_i, y_i)\}$ *for* $1 \le i \le N$, a set of N training examples that are representative of the underlying probability distribution. Unlike in the case of the Parzen window (Section "Parzen window estimator"), the GRNN's $\boldsymbol{c}_i$ is a d dimensional variable $\boldsymbol{c}_i \in \mathcal{R}^d$ and $\boldsymbol{c}_i = \boldsymbol{x}_i$. The y_i values are used to weight the kernels on the numerator but not the denominator of Eq. (6). As discussed in the section "Scaling factor", the scaling factor s is applied to optimize the width of covariance matrices, to result in correct scaling for y_i "heights". The functional form of the equation thus becomes:

$$f(\boldsymbol{x}\mid \boldsymbol{c}_i, \Sigma_i, s) = \frac{\sum_{i=1}^{N} y_i\phi(\boldsymbol{x}\mid \boldsymbol{c}_i, s\Sigma_i)}{\sum_{i=1}^{N} \phi(\boldsymbol{x}\mid \boldsymbol{c}_i, s\Sigma_i)}. \tag{6}$$

Proposed MB-GRNN model

Apart from the use of the GRNN as a predictor to define a control function, this paper presents a new Margin-Based GRNN (MB-GRNN) approach for outlier detection by defining upper and lower margin boundaries for a given dataset. This is achieved by adapting the GRNN algorithm, by applying stretch factors as a second kernel weighting factor. As stated in the section "A new method for outlier detection", the idea of weighting each one of the terms as shown in Eq. (7) with an additional weighting w_i has previously been used by other researchers (Cai, 2001; Lee and Zaknich, 2015). The MB-GRNN takes the form:

$$f(\boldsymbol{x}\mid \boldsymbol{c}_i, \Sigma_i, s, w_i) = \frac{\sum_{i=1}^{N} y_i w_i\phi(\boldsymbol{x}\mid \boldsymbol{c}_i, s\Sigma_i)}{\sum_{i=1}^{N} w_i\phi(\boldsymbol{x}\mid \boldsymbol{c}_i, s\Sigma_i)}. \tag{7}$$

The computational complexity to evaluate $f(\boldsymbol{x})$ in Eq. (7) uses time $O(Nd^2)$ and the calculation is well suited to vector processing methods. By choosing the correct set of stretch factors w_i, the GRNN can be made to follow the extremal upper or lower margin boundaries of a data cloud. The choice of stretch factors pushes the margin boundaries outward based on the characteristics of data. The resulting boundaries not only classify the data outliers but also can be used as a range predictor of y for any given $\boldsymbol{x}$ as shown in Figure 6. An approach for choosing the stretch factors w_i is described in the section "New margin-based GRNN (MB-GRNN) model using stretch factors".

Methodology

This section explains the experimental method applied to estimate the upper and lower design margin boundaries from experimental

Table 1. Experimental dataset parameters

Dataset	Variable ($\boldsymbol{x}_1$)	Variable ($\boldsymbol{x}_2$)	Variable ($\boldsymbol{x}_3$)	Variable ($\boldsymbol{y}$)
Auto MPG	Cylinder Displacement	Horsepower	Weight	Miles per gallon
Energy Efficiency	Room Surface Area	Wall Area	Roof Area	Heating Load
Concrete	Concrete Qty	Water Qty	NA	Compressive Strength

design datasets using the MB-GRNN, enabling margin prediction and outlier detection. Section "Algorithm Implementation Details" provides the pseudocode for the methodology implemented.

Experimental data and its characteristics

Analysis of design datasets has highlighted the widespread use of design margins. When viewed as a regression problem, and for design parameters $\boldsymbol{x}$ (independent variables), when a regression model g is learned from data such that $y = g(\boldsymbol{x})$, it was found that the resulting scalar variable y (dependent variable) inhabits a range $[y_{\min}, y_{\max}]$, often with fairly uniform density, such that $y_{\min} \leq g(\boldsymbol{x}) \leq y_{\max}$. However, unlike the margin applied to support vector regression problems, the range m is not of fixed width but depends on $\boldsymbol{x}$ such that $y_{\max} = y_{\min} + m(\boldsymbol{x})$. These ranges can be observed directly from experimental datasets, such as those in Figures 7 and 9. This creates the possibility of adopting a classification approach to detect outliers outside these typical design margins $y_{\min}$ or $y_{\max}$.

Three publicly available engineering multivariate design datasets were chosen for the experiment that displays the characteristics of margins. The first dataset used for experimental testing is called the "Auto MPG Data Set" and is from the UCI machine learning repository (Repository, 1993). It describes automobile fuel consumption for a set of multivalued discrete and continuous design parameters. The second set is an "Energy Efficiency Data Set" also from the UCI machine learning repository (Repository, 2012) that documents the heating and cooling loads of buildings based on their design parameters. The final dataset is a "Concrete Design Data Set" from the same source (Repository, 2007), describing the compressive strength of concrete based on the composition of ingredients. These datasets were selected as they relate directly to the design parameters of engineered artifacts and also have specific characteristics with multiple values of the dependent variables y for each value of the independent variable $\boldsymbol{x}$ (e.g., see Figs. 7 and 9). Use of these experimental datasets from multiple domains also suggests the versatility of the approach for other datasets having similar characteristics.

Data description

For the Parzen window approach, data were processed in the format $(\boldsymbol{x}, y)$ for $\boldsymbol{N}$ samples where each vector $\boldsymbol{x} = (x_1, x_2, x_3, x_4, \ldots, x_d)$, and for the MB-GRNN models, $\boldsymbol{x}$ was used as the independent variable and scalar y as the dependent variable. Parameters used from the experimental datasets are listed in Table 1.

The Python language and the Spyder IDE were used to process data, develop algorithms, and present results in graphical and tabulated formats. The implemented algorithms deal with both single and multiple dimensional data, and these datasets are used to explain the experimental results and analysis.

Schematic diagram of the method proposed

The schematic diagram, as shown in Figure 1, describes the overview of the processing steps applied for outlier detection and margin prediction.

Data preprocessing and clustering (Steps 1 and 2 in Fig. 1)

Section "Data description" describes the selection of variables for each data and array format. Each datum was first transformed onto a logarithmic scale to lower the skewness of its data distribution, improving the interpretation of data. The transformed data were normalized by linearly mapping each data to the unit interval [0,1] to improve training efficiency and to minimize the risk of certain variables being given more significance, especially when they represent different orders of magnitude (Patton, 1995).

The LBG vector quantization algorithm was used to segregate input data into an optimum number of clusters z to eliminate duplicates and minimize the influence of near duplicates. The optimum number of clusters z and the cluster mean values μ are further used to determine optimum covariance. The pseudocode for the LBG algorithm is explained in the section "Pseudocode for data preprocessing and clustering to remove near duplicates (Refer Steps 1 and 2 in Fig. 1)".

Determination of scaling factor and optimum covariance (Step 3 in Fig. 1)

The covariance matrix Σ_i is calculated by choosing a suitable set of neighbors with n members in the range $1 \leq n \leq z$, and a multiplicative factor called the scaling factor s that provide appropriate smoothing for Σ_i. The scaling factor s is determined using the LOOCV method, as described in the section "Smoothing parameter determination for Parzen and GRNN methods (Refer Step 3 in Fig. 1)". Figure 2 shows the graph of average probability versus scaling factors in the range [0.1,3] for neighbors n in the range [4,19]. The optimum scaling factor $s_{\text{opt}} = 0.53939$ that maximizes the global average probability was obtained[1] when $n = 4$ and is used to determine optimum covariance ($s_{\text{opt}}\Sigma_i$) when calculating the Parzen p.d.f.

Figure 3 shows the graph of GRNN mean squared error versus scaling factors in the range [0.1,3] for neighbors n in the range [3,19]. The optimum scaling factor $s_{\text{opt}} = 0.25757$ that maximizes the global average probability was obtained[1] when $n = 16$ and is used to determine optimum covariance ($s_{\text{opt}}\Sigma_i$) when calculating the smoothing parameter. The computational complexity to find the optimal value of s requires time $O(N^2d^2)$ and memory $O(Nd^2)$ and is largely determined by the calculation of the leave-one-out metrics. The method is well suited to a range of established bounded optimization techniques since the error

[1]For clarity and legibility, Figures 2 and 3 show only selected values of n neighbors to explain the methodology.

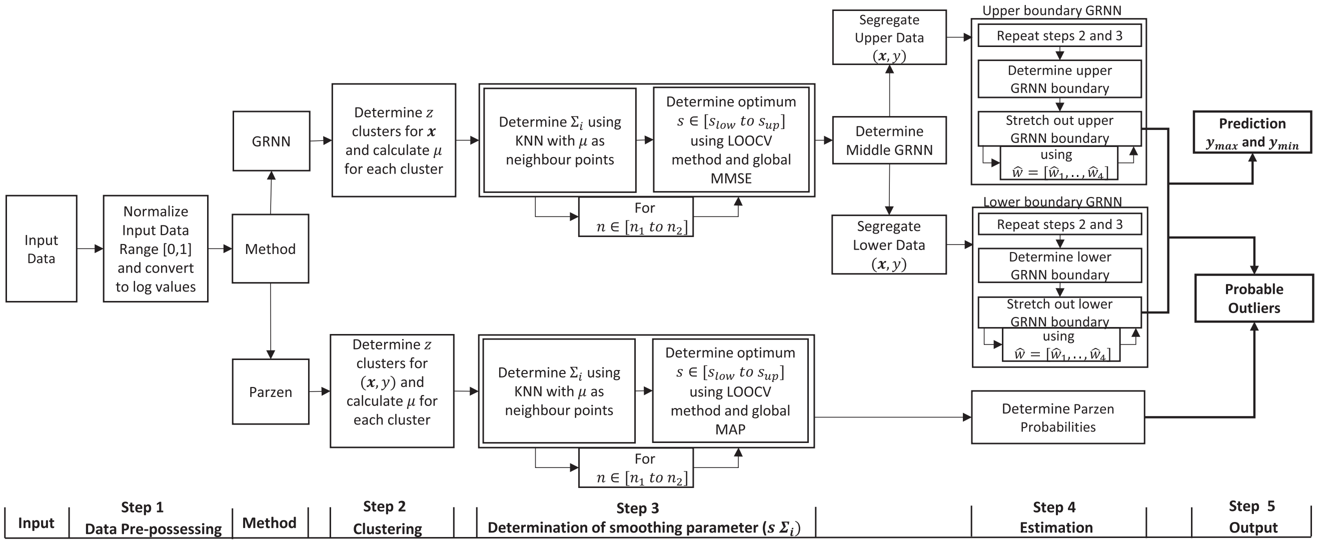

Input Data
Normalize Input Data Range [0,1] and convert to log values
Method
GRNN
Parzen
Determine z clusters for x and calculate μ for each cluster
Determine z clusters for (x,y) and calculate μ for each cluster
Determine Σ_i using KNN with μ as neighbour points
Determine optimum $s \in [s_{low}\ to\ s_{up}]$ using LOOCV method and global MMSE
Determine optimum $s \in [s_{low}\ to\ s_{up}]$ using LOOCV method and global MAP
For $n \in [n_1\ to\ n_2]$
Segregate Upper Data (x,y)
Determine Middle GRNN
Segregate Lower Data (x,y)
Upper boundary GRNN
Repeat steps 2 and 3
Determine upper GRNN boundary
Stretch out upper GRNN boundary using $\hat{w} = [\hat{w}_1,..,\hat{w}_4]$
Lower boundary GRNN
Determine lower GRNN boundary
Stretch out lower GRNN boundary using $\hat{w} = [\hat{w}_1,..,\hat{w}_4]$
Determine Parzen Probabilities
Prediction y_{max} and y_{min}
Probable Outliers
Input
Step 1 Data Pre-possessing
Method
Step 2 Clustering
Step 3 Determination of smoothing parameter ($s\ \Sigma_i$)
Step 4 Estimation
Step 5 Output

Fig. 1. Schematic diagram of Parzen and MB-GRNN application for outlier detection and prediction (for variable notations, refer Table A.1).

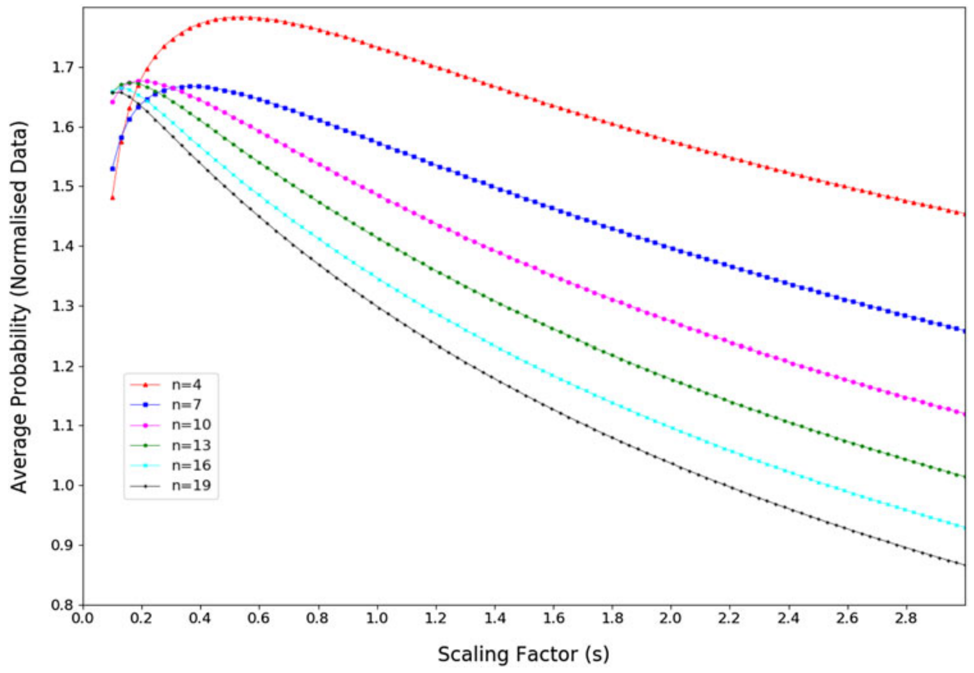


Fig. 2. Parzen: selection of scaling factor (*s*) for (*n*) neighbors.

surfaces are scalar, smooth, and typically have a global minimum/maximum. Experiments were conducted using various bounded, and root-finding optimization functions for the three datasets presented in this paper. The results suggest significant savings in computational time to reach the convergence. A novel derivative function was used for root-finding functions, and these methods and experimental results will be presented in a future paper.

Estimation (Step 4 in Fig. 1)

Parzen window p.d.f estimate

The implementation uses a Gaussian window function to estimate the density of samples using the optimized covariance obtained in Eq. (4). Figure 1 shows the steps involved, and the overall implementation uses Eqs (1)–(3).

New MB-GRNN model using stretch factors

The MB-GRNN implementation involves determining three inter-related GRNN models: for the middle, upper, and lower boundaries. These boundaries provide a way of ordering outliers to measure the distances beyond the typical limits using an error margin Error >0 as outlined in S23a and S23b of Table A.3. Firstly, a conventional GRNN model (providing the middle boundary) is found to segregate the data into the upper and lower subsets, and subsequently, the upper and lower boundaries are determined using two stretched GRNNs.

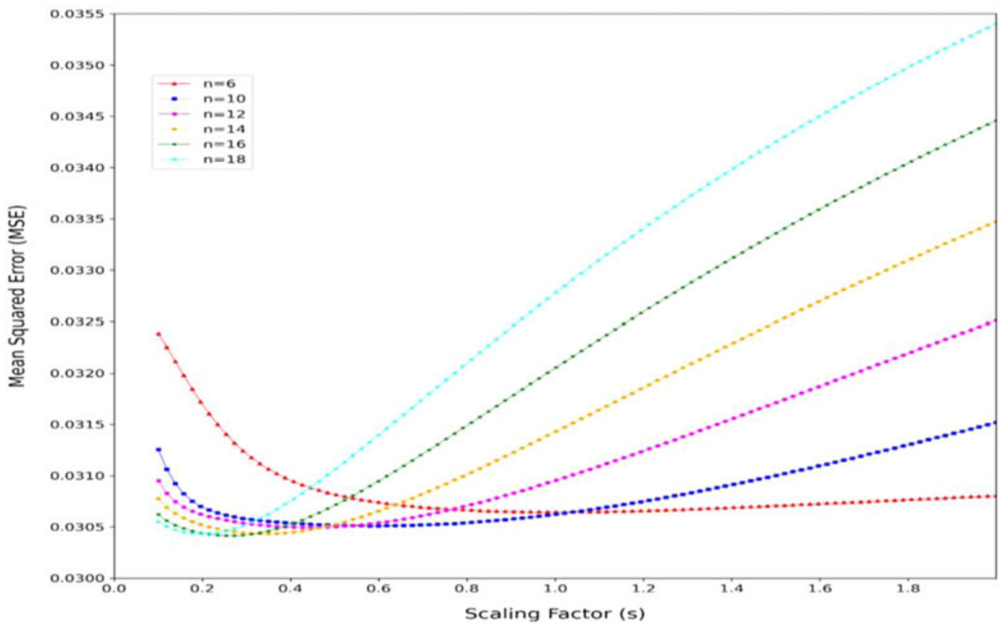


Fig. 3. GRNN: selection of scaling factor (s) for (n) neighbors.

Middle GRNN and segregation of datasets for the upper and lower GRNN

The scaling factor s_{opt} and optimum covariance matrix $s_{\mathrm{opt}}\Sigma_i$ were determined as per the principles described in the section "Determination of scaling factor and optimum covariance (Step 3 in Fig. 1)". Once the middle GRNN model is found using Eq. (6), it can be used to generate a predicted value $y_p = f(\boldsymbol{x})$ for each $\boldsymbol{x}_i$. The green dotted line in Figure 6 shows the result for the AutoMPG middle GRNN model estimated from the entire set of data and denoted as y_p^m. The distance $\delta_i^m = y_i - y_p^m$, where y_i represent the actual value and y_p^m is the corresponding predicted value, was used to segregate the dataset into disjoint upper and lower subsets, which were used to determine the upper and lower GRNN model, respectively. All y_i values with $\delta_i^m > 0$ were used for the upper MB-GRNN model versus those for which $\delta_i^m \leq 0$ that from the lower model.

Determination of the upper and lower GRNN boundaries

For each of the two segregated upper and lower datasets, the optimum number of clusters z, scaling factor s_{opt}, and optimum covariance matrix $s_{\mathrm{opt}}\Sigma_i$ were estimated. A set of four stretch factor thresholds were chosen $\hat{w} = (0.03,\ 0.1,\ 0.3,\ 0.5)$ for the experiment. These specific values were not found to be critical for success as they appear both in the numerator and denominator of the GRNN, and so they self-normalize. However, it is important that the values must span a wide range of at least two orders of magnitude to ensure that some points have more prominent weightings than others. The MB-GRNN was used to generate upper (or lower) GRNN models using Eq. (7), by applying the weight thresholds iteratively for datasets that lie within the upper and lower boundaries. This iterative process pushes out the upper (or lower) boundaries by ensuring that points closer to the upper and lower boundaries have a higher weighting than the interior points. The process shown in Table A.3 shows the greedy algorithm that was developed for determining the stretched upper and lower GRNN margin boundaries.

For this algorithm, a stretch factor threshold with $M = 4$ iterations was chosen as a trade-off between computational efficiency and performance. However, a convergence criterion could be applied for a range of stretch factor thresholds until no new data points get shifted from $\delta_i^u > 0$ to $\delta_i^u \leq 0$ in any subsequent iteration for upper boundary and vice versa for lower boundary. The computational time of the stretch algorithm grows approximately linearly with M resulting in $O(MN + N^2d^2)$ since the optimal scaling factor s need to be re-estimated once each for the upper and lower datasets, whereas the memory requirement is $O(Nd^2)$. Figure 4 shows the outcome of an experiment utilizing artificial data and using a known function $0.1\sin 9x + 0.75$ (only the upper GRNN curves are shown for clarity). The orange curve shows the middle GRNN, the blue curve shows the surfaces for each stretch step iteration, the green curve shows the stretch boundary after eight stretch steps, and the red curve shows the artificially generated upper surface for the known function. Eight stretch factor thresholds were used in a uniform logarithmic scale between 10^{-2} and 1. Figure 5 shows the progression of the stretch steps and MSE calculated for each stretch step against the known function. The experimental results suggest that the optimum performance is obtained after the first four stretch steps, after which the difference in MSE is nominal.

Once the GRNN upper and lower boundaries are determined using the stretch iterations, either points that lie outside of the extremal boundaries can be nominated as outliers as explained in the section "Experimental results and analysis", or a reliability threshold can be applied using a distance metric to identify outliers and near outliers as explained under the section "Experimental results and outlier detection".

Experimental results and analysis (Step 5 as per Fig. 1)

The following sections explain the experimental results and analysis obtained from the application of the MB-GRNN and Parzen methods to the univariate and multivariate dataset parameters in Table 1. Results from the MB-GRNN and Parzen methods use

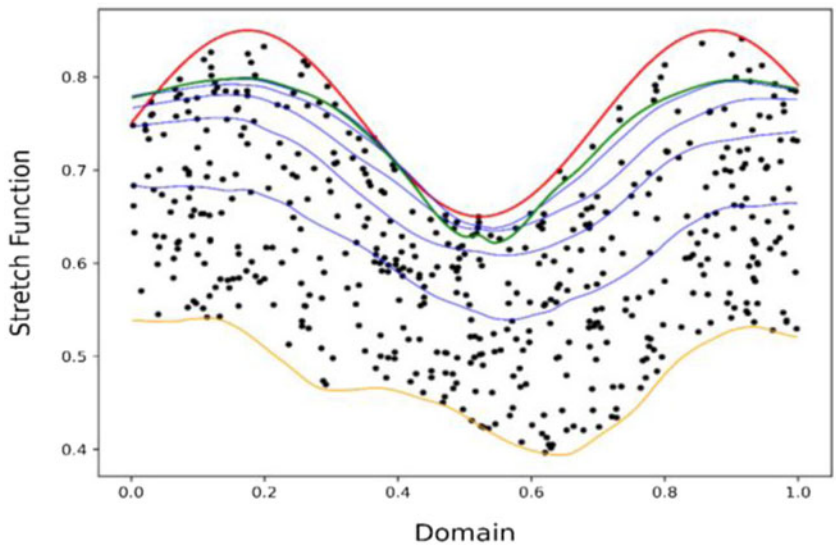


Fig. 4. Progression of stretch algorithm.

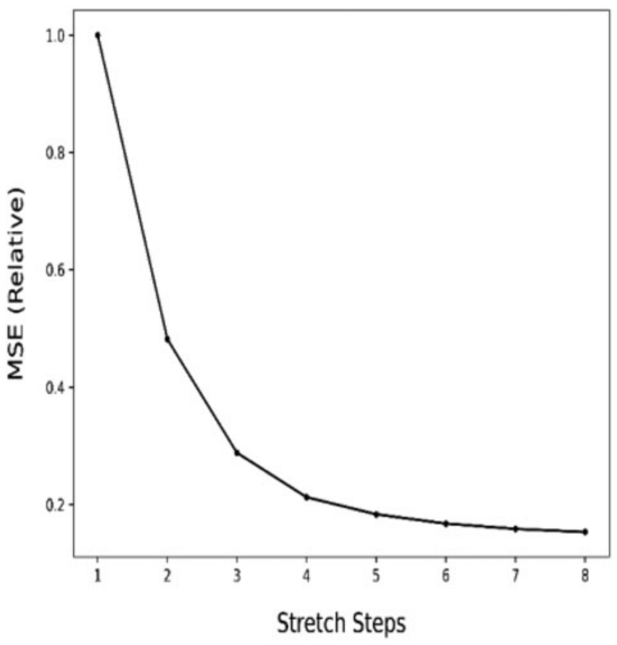


Fig. 5. Stretch steps and MSE.

similar optimization techniques as analyzed in the sections "AutoMPG dataset" and "Energy efficiency and concrete datasets".

AutoMPG dataset

Firstly, results and analysis for a univariate dataset, consisting of 199 examples, are explained to introduce the upper and lower GRNN margin boundaries derived using an MB-GRNN. This method is compared with the Parzen probabilistic approach. Later, results and analysis will focus on multivariate datasets.

The similarities and differences between the MB-GRNN and Parzen methods are indicated using color codes and markers. The colors and markers used in figures and legends have been kept consistent and are explained immediately below.

Each example was ranked in order of its Parzen probability with low probability, suggesting that a point might be an outlier. Data points that are outside the upper or lower GRNN boundaries as estimated by the MB-GRNN are treated as outliers with Error >0. Outliers estimated by the MB-GRNN are compared with the 25 lowest Parzen probabilities shown in Figures 6, 7, 9 and 10.

The small black dot "." denotes points that are not detected as outliers by the MB-GRNN (Error = 0 as per S23a and S23b of Table A.3) and have higher Parzen probability ranking ≥25, indicating agreement between the two methods.

The green colored "∧" data points also show cases with a very high degree of agreement between the two methods since the MB-GRNN identified them as outliers with the cost of error >0, and they also have Parzen least probability ranking ≤15. These denote most likely outlier points.

The magenta colored "■" symbol shows the outliers identified by MB-GRNN (Error > 0) and with least Parzen probabilities ranking <15 and ranking >25, nominating them as potential outliers. The thresholds[2] 15 and 25 were chosen as they correspond to 7.5% and 12.5% of points in the dataset, respectively.

The red color "*" data points are cases where the two methods disagree. They were not predicted as outliers by the MB-GRNN (Error = 0) but are within the lowest 15 Parzen probabilities. These data points are in regions with low point densities either at the extremities of the $\boldsymbol{x}$ domain or between higher density regions within the range. Due to MB-GRNN's interpolative[3] property, these points are judged by their y values being located between the upper and lower surfaces.

On the other hand, the blue data points "X" were identified as legitimate outliers by the MB-GRNN (Error > 0), and where least Parzen probabilities were ranked ≥25. These data points were

[2]These thresholds were chosen arbitrarily to compare the performance and to present the results.

[3]Subject to the specific selection of dependent and independent variables chosen for the experimental analysis. These data points could become legitimate outliers should a different axis system be chosen for the determination of MB-GRNN margin boundaries. A method for selection of dependent and independent variables will be presented in a future paper.

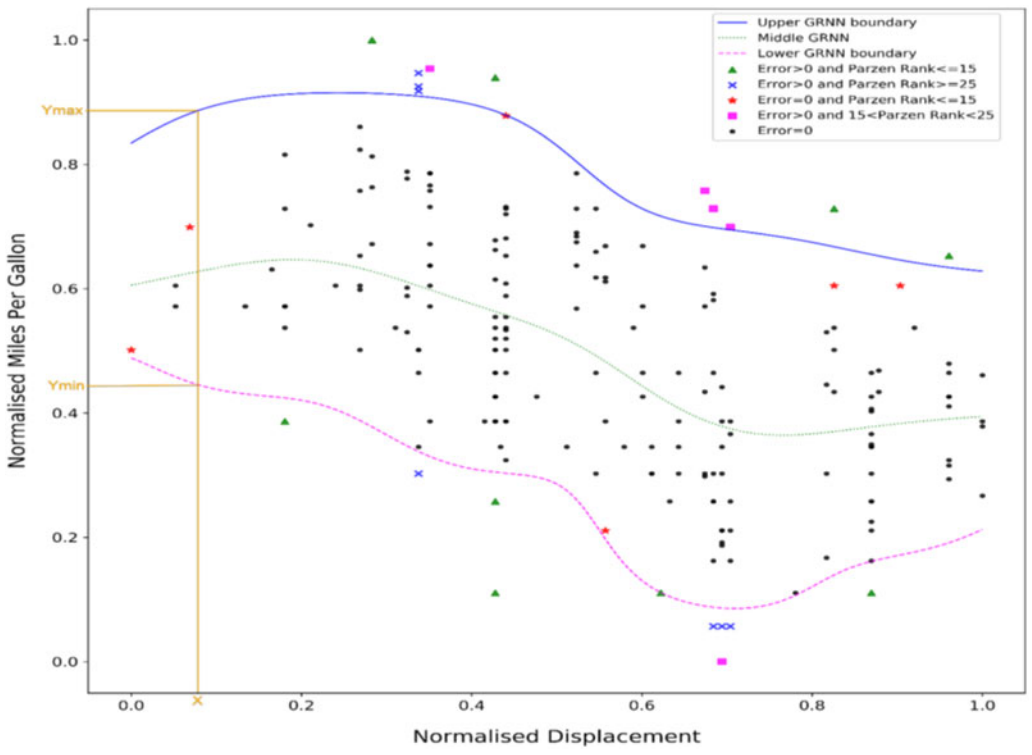


Fig. 6. AutoMPG univariate outlier detection: MB-GRNN and Parzen plots.

either above or below the extremal boundaries determined by the MB-GRNN.

The outlier classifications from both the MB-GRNN and Parzen methods for the selected univariate dataset are shown in Figure 6. Each case is color coded to show the degree of agreement or disagreement between the two methods. There is a high degree of correlation between the two methods (the black dots); however, some results contrast due to the different approaches.

The Parzen and MB-GRNN methods use similar optimization techniques to determine the scaling factor and covariance matrix as described previously in the section "Determination of scaling factor and optimum covariance (Step 3 in Fig. 1)". Unlike the conventional GRNN that only determines the line of best fit, the MB-GRNN can determine the upper and lower boundaries of the dataset to predict a range of legitimate y values for any given $\boldsymbol{x}$ and use these boundaries as the limiting "fence" to identify the possible outliers as shown in Figure 6. Unlike some other regression methods, such as those based on support vectors (Cristianini and Shawe-Taylor, 2000), that provide a constant width band of insensitivity, the MB-GRNN determines design margins that reflect the data distribution. These results suggest that the proposed MB-GRNN method can be a useful approach for the industry datasets that have a range of y values for any given $\boldsymbol{x}$.

Figure 7 compares the result of two different methods for a multivariate dataset (x_1, x_2, y). The results suggest that the MB-GRNN and Parzen methods identify different types of outliers. The MB-GRNN predicts outliers that are outside the upper and lower regression surfaces formed using a weighted conditional expectation on the chosen dependent variable y and independent variable $\boldsymbol{x}$, whereas the Parzen method uses a joint probability for the combination of ($\boldsymbol{x}$, y). Both methods have merits in identifying outliers as Parzen predicts the least probable data points by estimating lower probabilities, whereas the MB-GRNN predicts outliers based on a chosen dependent variable.

Compared to the univariate dataset, greater discrepancies were observed between the MB-GRNN and Parzen methods for datasets with multivariate independent variables. For the most isolated points, the green color "▲" and magenta colored "■" data points show a high degree of agreement between the two methods. From Figure 7, it can also be observed that the red "*" data points are outliers effectively detected by Parzen, and the blue colored "X" maximal/minimal data points are outliers best detected by the MB-GRNN. Just like Figure 6, the red "*" are at the extremities of the domain, and the blue "X" / magenta "■" are at the extremities of the range.

Table 2 shows the result of the two different methods applied to a four-dimensional multivariate dataset (x_1, x_2, x_3, y) and Figure 8 shows the parallel coordinates plot of outliers detected by both methods. The data points with MB-GRNN errors (Error > 0) shown in green-dashed, magenta-dashdotted, and blue-solid categories and Parzen outliers not detected by MB-GRNN (Error = 0) are shown in the red-dotted category. These results follow a similar pattern with previous experiments presented above.

The green-dashed, magenta-dashdotted, and blue-solid category data points in Figure 8 indicate the outliers identified by the MB-GRNN, due to a single or a combination of independent variables (x_1, x_2, x_3) having unexpectedly low or high values. The green-dashed and magenta-dashdotted points were also identified as outliers by the Parzen method as one or more variables (x_1, x_2, x_3, y) were extremal in the domain with ranking MB-GRNN Error > 0 and 0 < Parzen Rank <25. For example, the data point with minimum miles per gallon (value 18) was identified as a green-dashed category outlier, and this was due to high values of displacement, horsepower, and weight (values 121, 112, and

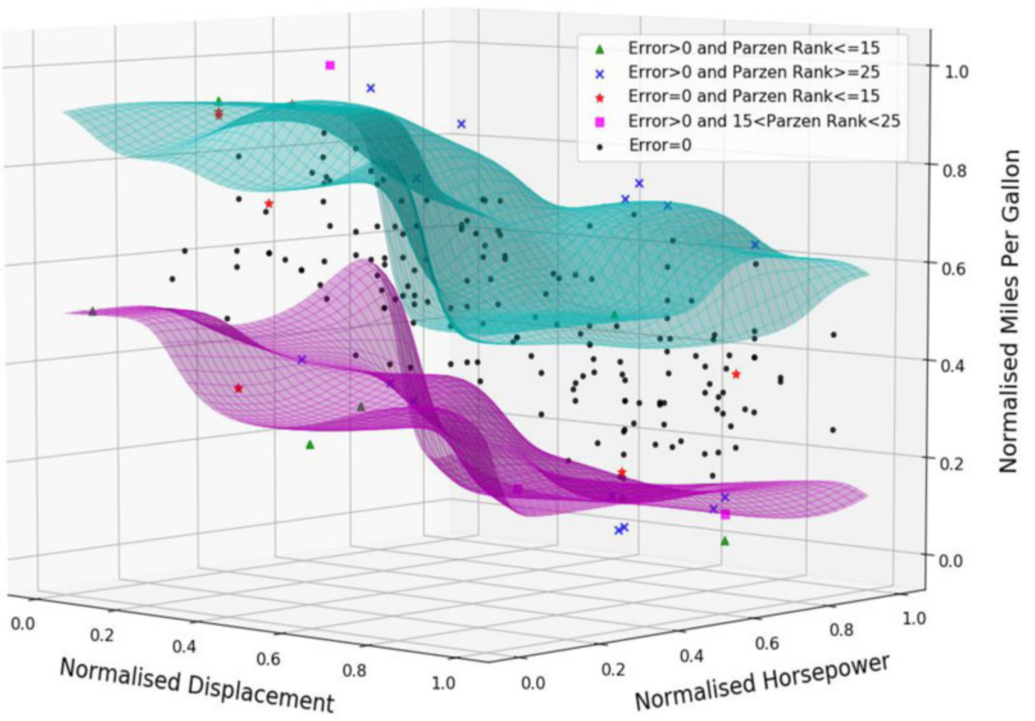


Fig. 7. AutoMPG multivariate outlier detection: MB-GRNN and Parzen.

Table 2. Analysis of results of AutoMPG multivariate dataset with (x_1, x_2, x_3, y) variables

Category (Color – Linestyle)	Ranking Criteria	Number of data points	% distribution of data points
Green – dashed	MB-GRNN Error > 0 and Parzen Rank ≤15	7	3.5
Magenta – dashdotted	MB-GRNN Error > 0 and 15 < Parzen Rank <25	3	1.5
Blue – solid	MB-GRNN Error > 0 and Parzen Rank ≥25	27	13.6
Red – dotted	MB-GRNN Error = 0 and Parzen Rank ≤15	8	4.0
Black – NA	MB-GRNN Error = 0	154	77.4

2933, respectively), resulting in lower fuel economy. Both Parzen and MB-GRNN methods identified this data point as an outlier, as it is positioned at the extremities of both range and domain. Whereas the maximum miles per gallon (value 46.6) is shown as a blue-solid category outlier due to relatively low values of displacement, horsepower, and weight (values 86, 65, and 2110), resulting in higher fuel economy. This data point was identified as an outlier by the MB-GRNN due to it being located in the extremities of the range. However, Parzen did not find this data point as an outlier as it was not located on the extremities of the domain. The red-dotted category data points were identified as outliers by the Parzen method due to a single or a combination of variables having significantly low or high values in the domain. An example of red-dotted data point is with displacement, horsepower, weight, and MPG values of 72, 69, 1613, and 35, respectively, with displacement and weight having low values in the domain. The results suggest that both Parzen and MB-GRNN can complement each other to detect outliers in the domain and range (for a given value of y), respectively.

Energy efficiency and concrete datasets

Figures 9 and 10 show the upper and lower margin boundaries for the energy efficiency and concrete multivariate design datasets. The algorithm and optimization techniques identical to that of AutoMPG were applied to these datasets. The results are consistent with the AutoMPG dataset presented in Figure 7.

The building energy efficiency data shown in Figure 9 can be seen to be heavily quantized in the domain, having a wide range of y values for a given $\boldsymbol{x}$. Even with such quantized data, the MB-GRNN was able to predict margin boundaries and pick the blue colored "X" points as outliers to identify values that are away from the distribution. The red "*" data points identified as outliers by Parzen being extremal points in the domain.

For the concrete dataset shown in Figure 10, the upper and lower boundaries are more complex. Despite this, the solution still only finds a small fraction (18%) of data points as outliers by combined Parzen and MB-GRNN methods.

Adaptation for incremental learning

Any online implementation of the algorithm described should deal efficiently with the inclusion of a new data point ($\boldsymbol{x}$,y) into S_{train}. The GRNN is a nonparametric model, so parameter re-estimation is not required with the advent of additional data points. However, the computational complexity of evaluating the model grows linearly with N, so any software implementation might need to limit N by, for instance, avoiding near duplicates to

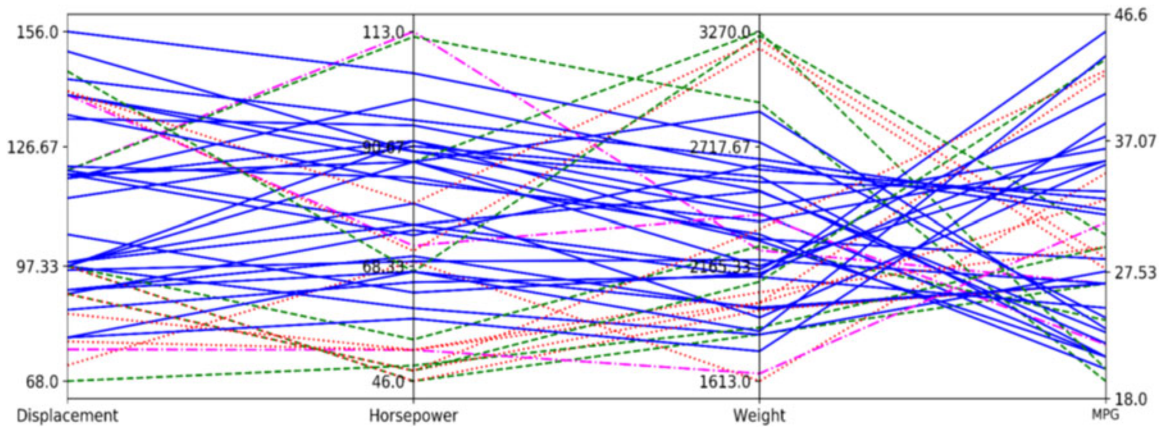


Fig. 8. Parallel coordinates plot of outliers for data points in Table 2 (black is hidden from the plot so as not to overwhelm the plot).

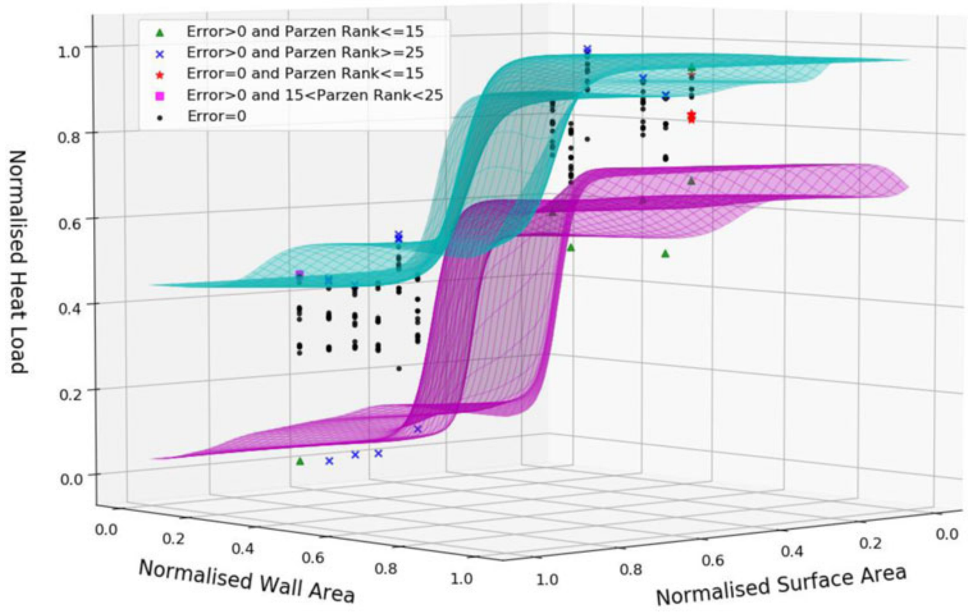


Fig. 9. Energy efficiency multivariate outlier detection: MB-GRNN and Parzen.

existing data or aging data measurements so that outdated measurements are excluded from consideration.

Although the GRNN itself is nonparametric, there are three parameters described elsewhere in this method. The first is derived from nearest neighborhood distance sets S_i, so the advent of a new data point would only affect one n-neighbor set and necessitate recalculation of one covariance matrix Σ_i at computational time complexity $O(Nd^2)$.

The second parameter is the scalar stretch scaling s. The advent of one new data point would have little impact on the margin estimates. However, once sufficient new data has been inducted into S_{train} a *batch* process would be needed to re-estimate s. As this is a well-behaved optimization problem, the previous value could be used as a seed for any optimization technique ensuring swift convergence to a new global optimum. This comes at a computational time cost $O(N^2)$.

Finally, the weighted GRNN is controlled by a parameter set $\hat{w}$. Since this is the focus of future research, it is likely that shortcuts can be found to further reduce this computational cost. For instance, a new data point can be temporarily assigned the same weighting as its nearest neighbor and then a *batch* process could periodically seek a global solution.

Conclusion

A novel MB-GRNN method has been presented in this paper that can learn the extremal upper and lower regression boundaries of a given data cloud by "stretching out" the GRNN surfaces. This method is free from any assumptions about the form of the data and is able to learn functions directly from training data to classify outliers and also to predict a range of valid parameters for any given new input parameter. The method has been tested experimentally on three different multivariate datasets, and the results were compared with the well-known Parzen window method. The results presented in the section "Experimental results and analysis" show a high level of agreement between the two methods, as might be expected, but with Parzen and MB-GRNN having their own distinct benefits and limitations.

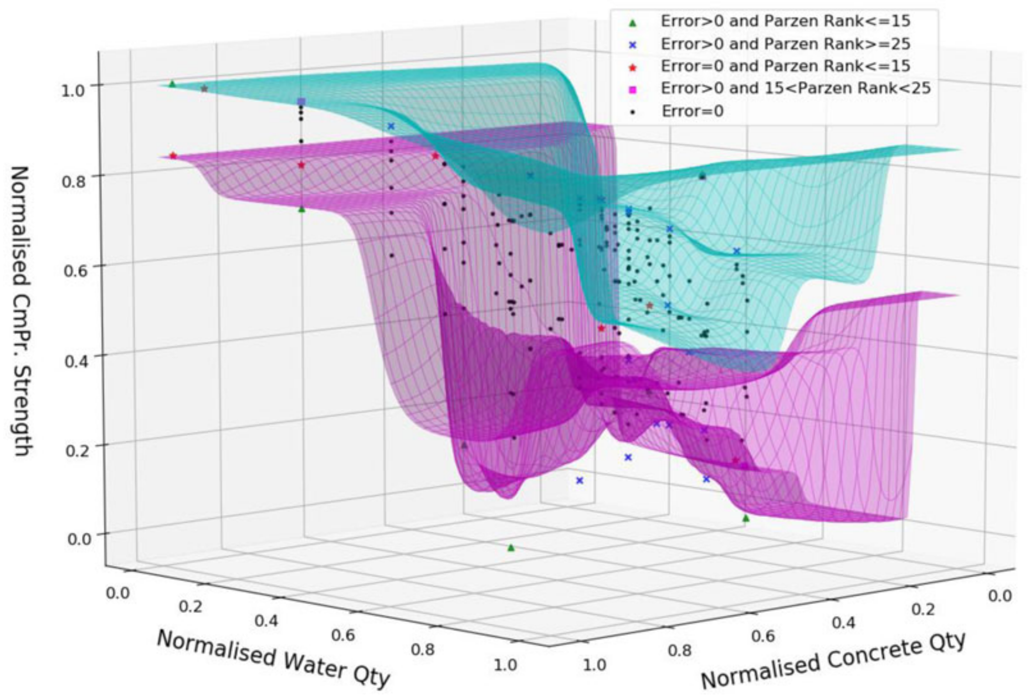


Fig. 10. Concrete multivariate outlier detection: MB-GRNN and Parzen.

The MB-GRNN was able to detect outliers not found by the Parzen probabilistic approach, but the regression method has the limitation that it nominates certain isolated and extreme data points within the regression boundaries as valid data. Such data points located in the extremities of the domain can be detected by Parzen as being outliers. However, considering the Safe Operating Limits (SOLs), it is debatable whether these data need to be treated as outliers as they are within the allowable margin. As the experiments show, for design datasets that show the characteristics of margins, the MB-GRNN provides better results than other methods, especially identifying outliers that lie near to the maximum or minimum margin boundary limits. The Parzen method provides a lower probability for isolated or extreme data due to the cluster centers being far away from the outskirts, whereas the GRNN, being a regression method, only detects extremities in the function's range rather than its domain. The MB-GRNN also provides better interpolation, making it much more versatile at establishing relationships between design variables.

Both the MB-GRNN and Parzen methods used similar optimization techniques in these experiments, creating the opportunity to use both methods in conjunction to form a decision support system that highlights potential design errors in industrial design datasets. Both methods are unsupervised and work very well for multivariate data. For industrial design data, these methods have the potential to learn from large volumes of available design data, identify data patterns from key parameters and rank the plausibility of data against the model. It is expected that this approach will benefit the design industry by detecting possible design errors that are otherwise extremely difficult and time-consuming to detect using manual checking.

With large quantities of multivariate data, considerable computational time would be needed to implement the methods described in this paper. To improve the computational time for the determination of smoothing parameter, two other methods, including a novel derivative-based root-finding method, have been experimented successfully using all three datasets presented in this paper. The results are consistent with the smoothing parameter method presented in this paper but with a significant reduction in computation time. It is envisaged that these new methods will be presented in a future publication.

The MB-GRNN algorithm can be applied to datasets having multiple values of the dependent variable for a given independent variable. The paper analyzed the application of algorithms for sample datasets relating to automobiles, building energy efficiency and concrete compressive strength. Machinery (e.g., rotating equipment) or systems (e.g., piping systems in a process plant) that have parameters to function within a specific range of design or operating limits have the prospect for application of this algorithms'. A cross-validation scheme is also needed in practice with existing design margin definitions to validate the algorithms" accuracy. It is expected that the algorithm has the potential for application with both online and offline datasets. The GRNN requires the determination of only one smoothing parameter, and with a significant reduction in computational time to determine this parameter using these as yet unpublished optimization methods, this creates the opportunity to extend this algorithm for online datasets.

Conflict of interest. The authors declare none.

References

Aglodiya A (2017) Application of artificial neural network (ANN) in chemical engineering: a review. *IJARIIE* **3**, 5322–5328. ISSN 2395-4396 (Online).

Al-Mahasneh AJ, Anavatti SG and Garratt MA (2018) Review of applications of generalized regression neural networks in identification and control of dynamic systems, pp. 1–5. Available at http://arxiv.org/abs/1805.11236.

Austin-Breneman J, Yu BY and Yang MC (2015) Biased information passing between subsystems over time in complex system design. *Journal of Mechanical Design* **138**, 1–9. doi:10.1115/1.4031745.

Baqqar M, Wang T, Ahmed M, Gu F, Lu J and Ball A (2012) A general regression neural network model for gearbox fault detection using motor operating parameters. *UKACC International Conference on Control 2012*, Cardiff, UK.

Berkhin P (2002) Clustering survey Bherkin (Accrue Software). Available at https://www.cc.gatech.edu/~isbell/reading/papers/berkhin02survey.pdf.

Blazquez-Garcia A, Conde A, Mori U and Lozano JA (2020) A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, 1–32. https://arxiv.org/abs/2002.04236v1.

Buchan AB and Bolton CJ (2009) Process and structural safety: do we need to have hazards? *IET Conference Publications* **2009**, 2–4. doi:10.1049/cp.2009.1539.

Cai Z (2001) Weighted Nadaraya-Watson regression estimation. *Statistics and Probability Letters* **51**, 307–318. doi:10.1016/S0167-7152(00)00172-3.

Celikoglu HB and Cigizoglu HK (2007) Public transportation trip flow modeling with generalized regression neural networks. *Advances in Engineering Software* **38**, 71–79. doi:10.1016/j.advengsoft.2006.08.003.

Cigizoglu HK and Alp M (2006) Generalized regression neural network in modelling river sediment yield. *Advances in Engineering Software* **37**, 63–68. doi:10.1016/j.advengsoft.2005.05.002.

Cover TM and Hart PE (1967) Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* **13**, 21–27.

Cristianini N and Shawe-Taylor J (2000) *An Introduction to Support Vector Machines and Other Kernel Based Learning Methods*. New York: Cambridge University Press.

De Maesschalck R, Jouan-Rimbaud D and Massart DL (2000) The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems* **50**, 1–18. doi:10.1016/S0169-7439(99)00047-7.

Dempster AP (1972) Covariance selection. *Biometrics* **28**, 157–175.

Dowell AM (2001) Critical safe operating parameters: "never exceed" limit and "never deviate" action. *Process Safety Progress* **20**, 208–214. doi:10.1002/prs.680200310.

Eckert C and Isaksson O (2017) Safety margins and design margins: a differentiation between interconnected concepts. *27th CIRP Design 2017*.

Eckert C, Isaksson O and Earl C (2019) Design margins: a hidden issue in industry. *Design Science* **5**, 1–24. doi:10.1017/dsj.2019.7.

Forest J (2018) Know your limits. *Process Safety Progress* **37**, 498–501. doi:10.1002/prs.12000.

Ge Z, Song Z, Ding SX and Huang B (2017) Data mining and analytics in the process industry: the role of machine learning. *IEEE Access* **5**, 20590–20616. doi:10.1109/ACCESS.2017.2756872.

He X and He S (2011) Fault detection of excavator's hydraulic system using dynamic general regression neural network. *Applied Mechanics and Materials* **48–49**, 511–514. doi:10.4028/www.scientific.net/AMM.48-49.511.

Hodge VJ and Austin J (2004) A survey of outlier detection methodologies. *Artificial Intelligence Review* **22**, 85–126. doi:10.4324/9781315744988-22.

Islam MM, Lee G and Hettiwatte SN (2017) Application of a general regression neural network for health index calculation of power transformers. *International Journal of Electrical Power and Energy Systems* **93**, 308–315. doi:10.1016/j.ijepes.2017.06.008.

Jagan J, Samui P and Kim D (2019) Reliability analysis of simply supported beam using GRNN, ELM and GPR. *Structural Engineering and Mechanics* **71**, 739–749. doi:10.12989/sem.2019.71.6.739.

Jiang P and Chen J (2016) Displacement prediction of landslide based on generalized regression neural networks with K-fold cross-validation. *Neurocomputing* **198**, 40–47. doi:10.1016/j.neucom.2015.08.118.

Kartal S, Oral M and Ozyildirim BM (2018) Pattern layer reduction for a generalized regression neural network by using a self-organizing map. *International Journal of Applied Mathematics and Computer Science* **28**, 411–424. doi:10.2478/amcs-2018-0031.

Kim W and Katipamula S (2018) A review of fault detection and diagnostics methods for building systems. *Science and Technology for the Built Environment* **24**, 3–21. doi:10.1080/23744731.2017.1318008.

Kulkarni SG, Chaudhary AK, Nandi S, Tambe SS and Kulkarni BD (2004) Modeling and monitoring of batch processes using principal component analysis (PCA) assisted generalized regression neural networks (GRNN). *Biochemical Engineering Journal* **18**, 193–210. doi:10.1016/j.bej.2003.08.009.

Kumar G and Malik H (2016) Generalized regression neural network based wind speed prediction model for western region of India. *International Conference on Advances in Computing & Communications*, Cochin, India.

Lee GE and Zaknich A (2015) A mixed-integer programming approach to GRNN parameter estimation. *Information Sciences* **320**, 1–11. doi:10.1016/j.ins.2015.04.052.

Lee H, Kim S and Jun K (2018) The study for storm surge prediction using generalized regression neural networks. *Journal of Coastal Research* **85**, 781–785. doi:10.2112/si85-157.1.

Li H, Zhao J, Ni X and Zhang X (2018) Fault diagnosis for machinery based on feature extraction and general regression neural network. *International Journal of Systems Assurance Engineering and Management* **9**, 1034–1046. doi:10.1007/s13198-018-0726-9.

Linde Y, Buzo A and Gray RM (1980) An algorithm for vector quantizer design. *IEEE Transactions on Communications* **28**, 84–95. doi:10.1109/TCOM.1980.1094577.

May RJ, Maier HR, Dandy GC and Nixon JB (2004) General regression neural networks for modeling disinfection residual in water distribution systems. *World Water Congress*, Marrakech, Morocco.

Modarres M (2009) Advanced nuclear power plant regulation using risk-informed and performance-based methods. *Reliability Engineering and System Safety* **94**, 211–217. doi:10.1016/j.ress.2008.02.019.

Mohandes SR, Zhang X and Mahdiyar A (2019) A comprehensive review on the application of artificial neural networks in building energy analysis. *Neurocomputing* **340**, 55–75. doi:10.1016/j.neucom.2019.02.040.

Mussa HY, Mitchell JBO and Afzal AM (2015) The Parzen window method: in terms of two vectors and one matrix. *Pattern Recognition Letters* **63**, 30–35. doi:10.1016/j.patrec.2015.06.002.

Nadaraya EA (1964) On estimating regression. *Theory of Probability & Its Applications* **9**, 141–142. doi:10.1137/1109020.

Niu D, Liang Y and Hong W-C (2017) Wind speed forecasting based on EMD and GRNN optimized by FOA. *Energies* **10**, 12. doi:10.3390/en10122001.

Niu Q, Tong Q, Cao J, Zhang Y and Liu F (2019) On-line prediction remaining useful life for ball bearings via grey NARX. *Journal of Vibroengineering*. doi:10.21595/jve.2018.20120.

Ou TC and Hong CM (2014) Dynamic operation and control of microgrid hybrid power systems. *Energy* **66**, 314–323. doi:10.1016/j.energy.2014.01.042.

Pal M (2011) Modelling pile capacity using generalised regression neural network. *Proceedings of Indian Geotechnical Conference*, pp. 811–814.

Pal M and Deswal S (2008) Modeling pile capacity using support vector machines and generalized regression neural network. *Journal of Geotechnical and Geoenvironmental Engineering* **134**, 7.

Pandian S, Hassim MH, Ng RTL and Hurme M (2015) Designing an inherently healthier process based on inherently safer design (ISD) concept: research and development stage. *Clean Technologies and Environmental Policy* **17**, 1247–1259. doi:10.1007/s10098-015-0951-8.

Parzen E (1962) On the estimation of probability density functions and mode. *Annals of Mathematical Statistics* **33**, 1065–1076.

Patil S and Chouksey PD (2016) A survey on: distance based outlier detection. *International Journal of Science and Research (IJSR)* **5**, 280–282. doi:10.21275/v5i1.nov152729.

Patton JB (1995) Brushless DC motor control using a general regression neural network. *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, Orlando, USA.

Repository, UCI Machine Learning (1993) AutoMPG Data Set. Available at http://archive.ics.uci.edu/ml/datasets/Auto+MPG.

Repository, UCI Machine Learning (2007) Concrete Compressive Strength Data Set. Available at http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength.

Repository, UCI Machine Learning (2012) Energy Efficiency Data Set. Available at https://archive.ics.uci.edu/ml/datasets/Energy+efficiency.

Richardson M (2012) Standardizing Safe Operating Limit Information. *15th Annual International Symposium*, Houston, Texas, October 23–25.

Singh K and Cantt M (2012) Outlier detection: applications and techniques. *International Journal of Computer Science Issues* **9**, 307–323.

Specht DF (1991) A general regression neural network. *IEEE Transactions on Neural Networks* **2**, 568–576. doi:10.1006/brcg.1996.0066.

Stauffer T and Chastain-Knight D (2019) Do not let your safe operating limits leave you S-O-L (out of luck). *15th Global Congress on Process Safety*, New Orleans, LA.

Sutherland PE (2011) Safe operating limits: generator capability study for offshore oil platforms. *IEEE Industry Applications Magazine* **17**, 14–19. doi:10.1109/MIAS.2010.939648.

Wang H, Bah MJ and Hammad M (2019) Progress in outlier detection techniques: a survey. *IEEE Access* **7**, 107964–108000. doi:10.1109/ACCESS.2019.2932769.

Watson G (1964) Smooth regression analysis. *Sankhyā: the Indian Journal of Statistics; Series A* **26**, 359–372.

Willey RJ (2014) Layer of protection analysis. *Procedia Engineering* **84**, 12–22. doi:10.1016/j.proeng.2014.10.405.

Wynn DC and Eckert CM (2017) Perspectives on iteration in design and development. *Research in Engineering Design* **28**, 153–184. doi:10.1007/s00163-016-0226-3.

Xu L-Y, Zhang M, Zhu W and He Y-L (2013) Comparison of geometric and arithmetic means for bandwidth selection in NWKE. *Proceedings of the 2013 International Conference on Machine Learning and Cybernetics*, Tianjin, China.

Xu X, Liu H, Li L and Yao M (2018) A comparison of outlier detection techniques for high-dimensional data. *International Journal of Computational Intelligence Systems* **11**, 652–662. doi:10.2991/ijcis.11.1.50.

Zhang XH, Wang QJ, Zhu JJ and Zhang H (2012) Application of general regression neural network to the prediction of LOD change. *Chinese Astronomy and Astrophysics* **36**, 86–96. doi:10.1016/j.chinastron.2011.12.010.

Zimek A, Schubert E and Kriegel HP (2012) A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining* **5**, 363–387. doi:10.1002/sam.11161.

Jayaram Sivaramakrishnan recently completed his PhD degree in machine learning with the Discipline of Engineering and Energy at Murdoch University in Western Australia. Jaya is currently working in the Oil & Gas industry, and he has more than 25 years of experience in engineering, project execution and digital technologies. His research interests focus on the digitalization of industrial plants, with emphasis on the application of machine learning related to industrial design and fault detection.

Gareth Lee is an Adjunct Senior Lecturer in Engineering at Murdoch University. His research is concerned with pattern recognition methods and their application to fault diagnosis, decision support and asset management of industrial process and plant equipment.

David Parlevliet is the Deputy Head of Engineering & Energy at Murdoch University in Western Australia. With over ten years tertiary education experience, he teaches across Renewable Energy Engineering and Industrial Computer Systems Engineering. David works with the International Energy Agency Photovoltaic Power Systems Task 13 group on performance and reliability of PV systems with upcoming projects focusing on digitalization of solar plants. He is also the Chief Remote Pilot at Murdoch University and conducts research in autonomous systems.

Kok Wai Wong is an Associate Professor with the Discipline of Information Technology at Murdoch University in Western Australia. He is the current Vice President (Conference) for The Asia Pacific Neural Network Society (APNNS). He is also the current joint chapter chair for IEEE Computer Intelligence Society and IEE Robotic & Automation Society (WA Chapter). He is a Senior Member of Institute of Electrical and Electronics Engineers (IEEE), a Senior member of Australia Computer Society (ACS), and Certified Professional of ACS. He has involved in the editorial boards for a number of international journals and in many international conference organising committees. His current research interests include Intelligent Data Analysis, Artificial Intelligence and Virtual Reality.

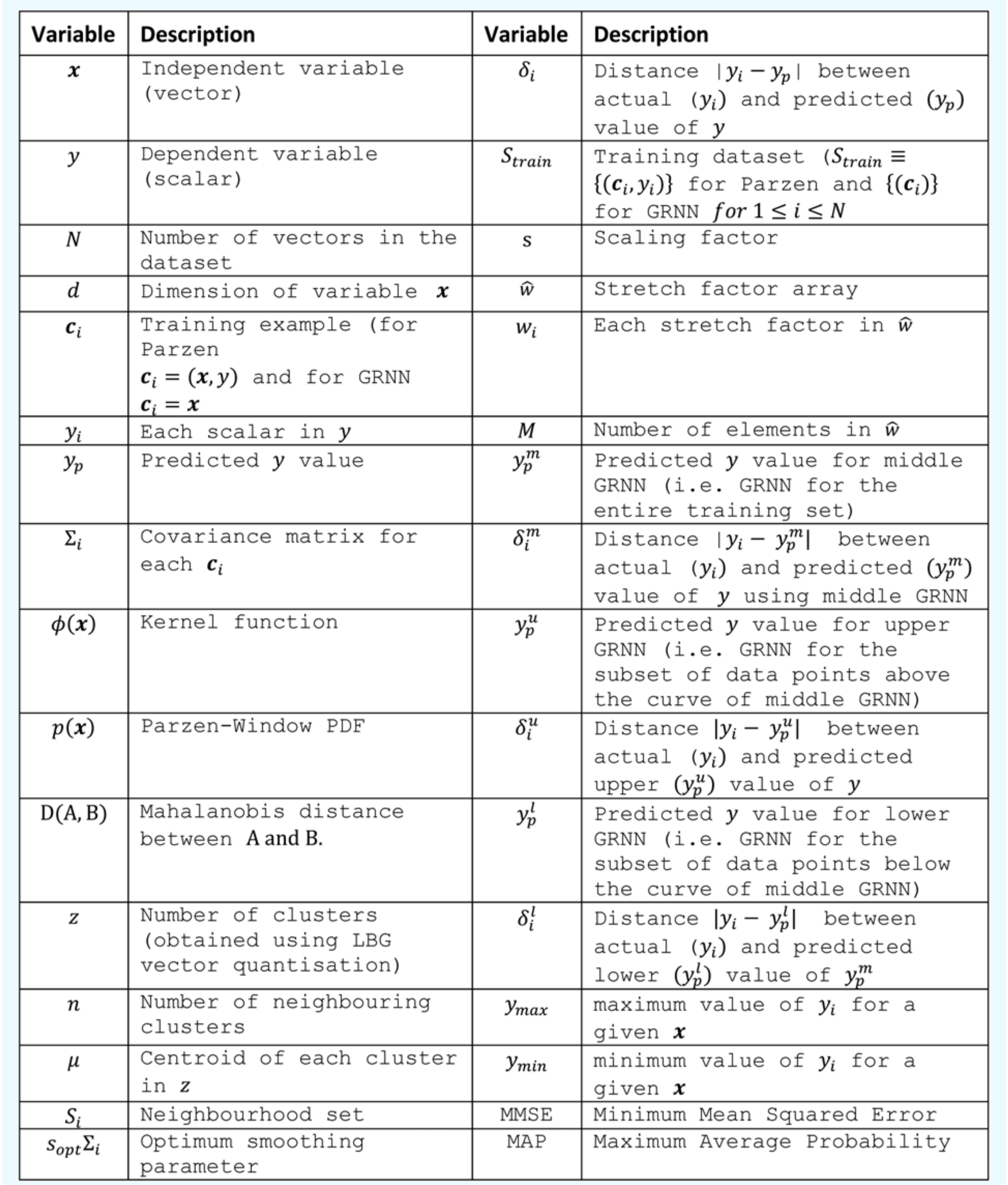
Appendix A: Algorithm Implementation Details

This section explains the algebraic notations and methodology applied to provide more clarity on the implementation steps to enable the reproduction of algorithms. Algorithm steps are explained using pseudocode along with initialization parameters and termination conditions used for the chosen example.

Variable notations and descriptions

Table A.1 provides details of algebraic notations used in the earlier equations.

Table A.1. Variable notations and descriptions



Variable	Description	Variable	Description
$\boldsymbol{x}$	Independent variable (vector)	δ_i	Distance $\lvert y_i - y_p \rvert$ between actual (y_i) and predicted (y_p) value of y
y	Dependent variable (scalar)	S_{train}	Training dataset $(S_{train} \equiv \{(\boldsymbol{c}_i, y_i)\}$ for Parzen and $\{(\boldsymbol{c}_i)\}$ for GRNN $for\ 1 \le i \le N$
N	Number of vectors in the dataset	s	Scaling factor
d	Dimension of variable $\boldsymbol{x}$	$\hat{w}$	Stretch factor array
$\boldsymbol{c}_i$	Training example (for Parzen $\boldsymbol{c}_i = (\boldsymbol{x}, y)$ and for GRNN $\boldsymbol{c}_i = \boldsymbol{x}$	w_i	Each stretch factor in $\hat{w}$
y_i	Each scalar in y	M	Number of elements in $\hat{w}$
y_p	Predicted y value	y_p^m	Predicted y value for middle GRNN (i.e. GRNN for the entire training set)
Σ_i	Covariance matrix for each $\boldsymbol{c}_i$	δ_i^m	Distance $\lvert y_i - y_p^m \rvert$ between actual (y_i) and predicted (y_p^m) value of y using middle GRNN
$\phi(\boldsymbol{x})$	Kernel function	y_p^u	Predicted y value for upper GRNN (i.e. GRNN for the subset of data points above the curve of middle GRNN)
$p(\boldsymbol{x})$	Parzen-Window PDF	δ_i^u	Distance $\lvert y_i - y_p^u \rvert$ between actual (y_i) and predicted upper (y_p^u) value of y
D(A, B)	Mahalanobis distance between A and B.	y_p^l	Predicted y value for lower GRNN (i.e. GRNN for the subset of data points below the curve of middle GRNN)
z	Number of clusters (obtained using LBG vector quantisation)	δ_i^l	Distance $\lvert y_i - y_p^l \rvert$ between actual (y_i) and predicted lower (y_p^l) value of y_p^m
n	Number of neighbouring clusters	y_{max}	maximum value of y_i for a given $\boldsymbol{x}$
μ	Centroid of each cluster in z	y_{min}	minimum value of y_i for a given $\boldsymbol{x}$
S_i	Neighbourhood set	MMSE	Minimum Mean Squared Error
$s_{opt}\Sigma_i$	Optimum smoothing parameter	MAP	Maximum Average Probability

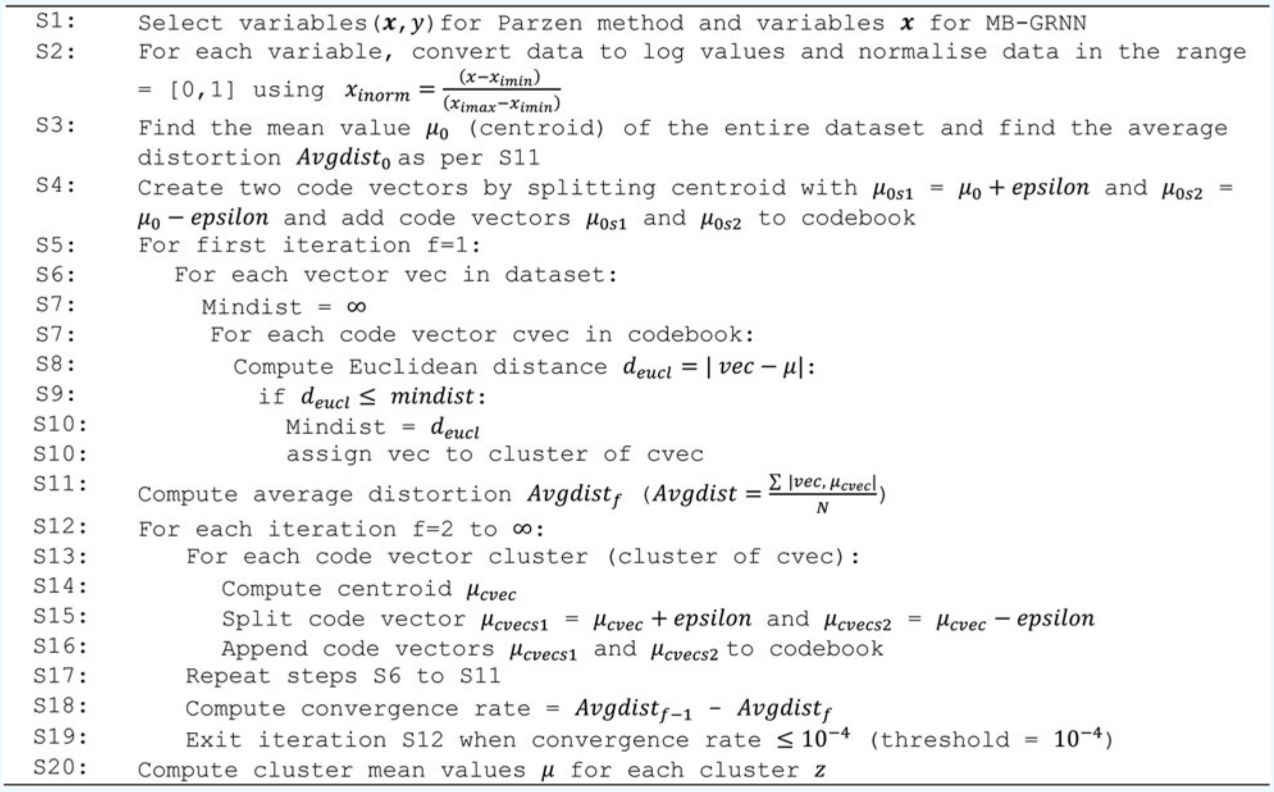
Input dataset used in this example

The dataset in Table A.2 has been used to explain the implementation of the algorithm in this section.

Table A.2. Example of dataset used in this section

Dataset	Variable (x_1)	Variable (x_2)	Variable (x_3)	Variable (y)
AutoMPG	Cylinder Displacement	Horsepower	Weight	Miles per gallon

Pseudocode for data preprocessing and clustering to remove near duplicates (Refer Steps 1 and 2 in Fig. 1)

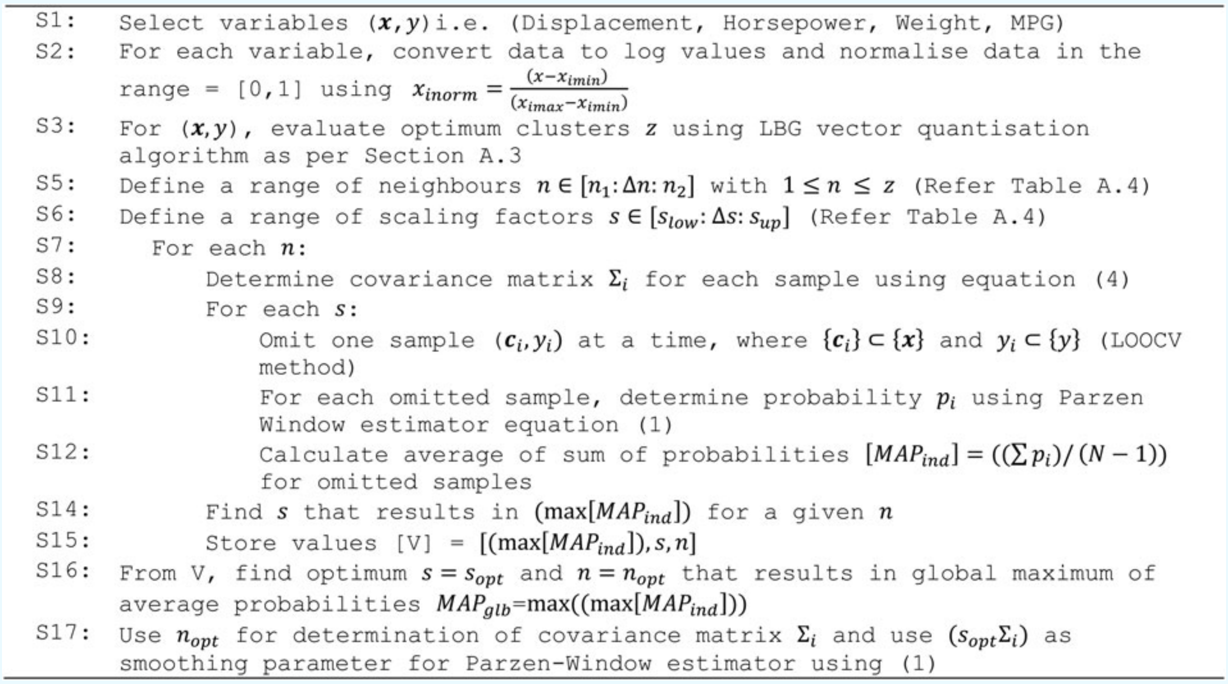


```
S1:   Select variables $(x, y)$ for Parzen method and variables $x$ for MB-GRNN
S2:   For each variable, convert data to log values and normalise data in the range
      = [0,1] using $x_{inorm} = \frac{(x - x_{imin})}{(x_{imax} - x_{imin})}$
S3:   Find the mean value $\mu_0$ (centroid) of the entire dataset and find the average
      distortion $Avgdist_0$ as per S11
S4:   Create two code vectors by splitting centroid with $\mu_{0s1} = \mu_0 + epsilon$ and $\mu_{0s2} =$
      $\mu_0 - epsilon$ and add code vectors $\mu_{0s1}$ and $\mu_{0s2}$ to codebook
S5:   For first iteration f=1:
S6:      For each vector vec in dataset:
S7:        Mindist = $\infty$
S7:         For each code vector cvec in codebook:
S8:           Compute Euclidean distance $d_{eucl} = |vec - \mu|$:
S9:             if $d_{eucl} \leq mindist$:
S10:              Mindist = $d_{eucl}$
S10:              assign vec to cluster of cvec
S11:  Compute average distortion $Avgdist_f$ ($Avgdist = \frac{\sum |vec, \mu_{cvec}|}{N}$)
S12:  For each iteration f=2 to $\infty$:
S13:      For each code vector cluster (cluster of cvec):
S14:         Compute centroid $\mu_{cvec}$
S15:         Split code vector $\mu_{cvecs1} = \mu_{cvec} + epsilon$ and $\mu_{cvecs2} = \mu_{cvec} - epsilon$
S16:         Append code vectors $\mu_{cvecs1}$ and $\mu_{cvecs2}$ to codebook
S17:      Repeat steps S6 to S11
S18:      Compute convergence rate = $Avgdist_{f-1}$ - $Avgdist_f$
S19:      Exit iteration S12 when convergence rate $\leq 10^{-4}$ (threshold = $10^{-4}$)
S20:  Compute cluster mean values $\mu$ for each cluster $z$
```

Smoothing parameter determination for Parzen and GRNN methods (Refer Step 3 in Fig. 1)

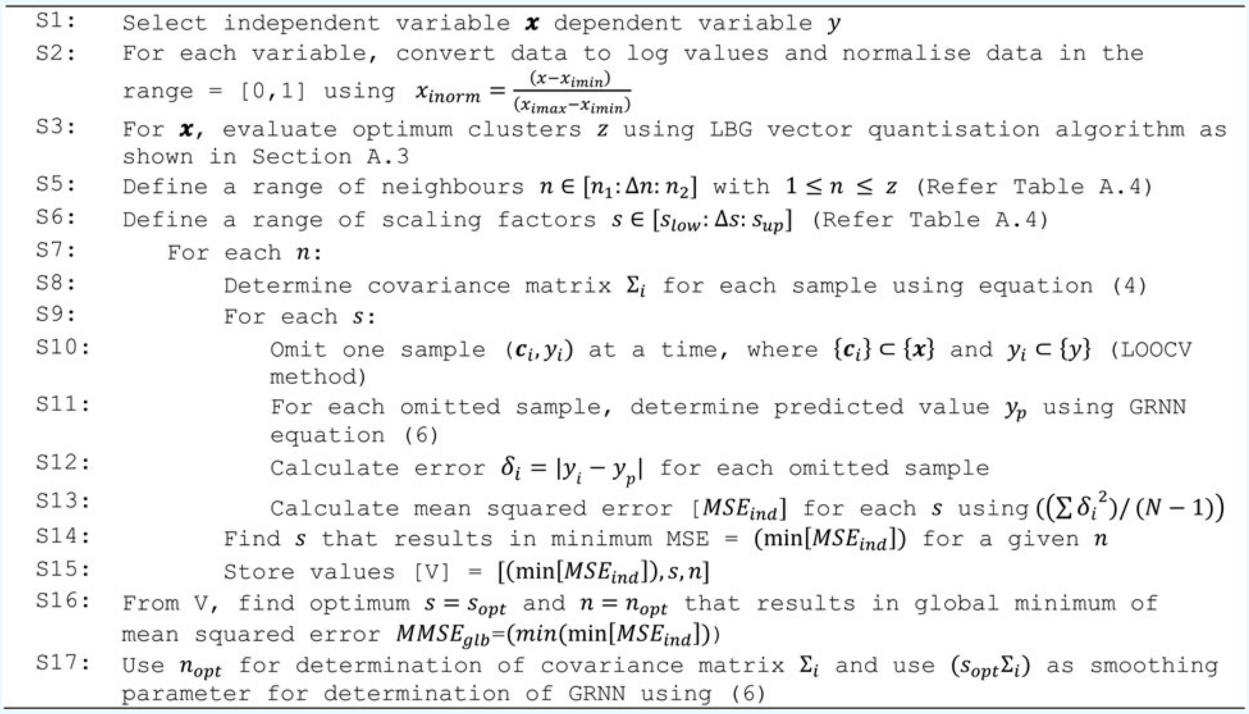
This section explains the determination of optimum smoothing parameter for both Parzen window and MB-GRNN methods.

Parzen window: Pseudocode for smoothing parameter determination



```
S1:  Select variables $(\boldsymbol{x}, y)$ i.e. (Displacement, Horsepower, Weight, MPG)
S2:  For each variable, convert data to log values and normalise data in the
     range = [0,1] using $x_{inorm} = \frac{(x - x_{imin})}{(x_{imax} - x_{imin})}$
S3:  For $(\boldsymbol{x}, y)$, evaluate optimum clusters $z$ using LBG vector quantisation
     algorithm as per Section A.3
S5:  Define a range of neighbours $n \in [n_1 : \Delta n : n_2]$ with $1 \le n \le z$ (Refer Table A.4)
S6:  Define a range of scaling factors $s \in [s_{low} : \Delta s : s_{up}]$ (Refer Table A.4)
S7:    For each $\boldsymbol{n}$:
S8:        Determine covariance matrix $\Sigma_i$ for each sample using equation (4)
S9:        For each $\boldsymbol{s}$:
S10:           Omit one sample $(\boldsymbol{c}_i, y_i)$ at a time, where $\{\boldsymbol{c}_i\} \subset \{\boldsymbol{x}\}$ and $y_i \subset \{y\}$ (LOOCV
               method)
S11:           For each omitted sample, determine probability $p_i$ using Parzen
               Window estimator equation (1)
S12:           Calculate average of sum of probabilities $[MAP_{ind}] = ((\sum p_i)/(N-1))$
               for omitted samples
S14:       Find $\boldsymbol{s}$ that results in $(\max[MAP_{ind}])$ for a given $\boldsymbol{n}$
S15:       Store values [V] = $[(\max[MAP_{ind}]), s, n]$
S16: From V, find optimum $s = s_{opt}$ and $n = n_{opt}$ that results in global maximum of
     average probabilities $MAP_{glb}$=max((max$[MAP_{ind}]$))
S17: Use $n_{opt}$ for determination of covariance matrix $\Sigma_i$ and use $(s_{opt}\Sigma_i)$ as
     smoothing parameter for Parzen-Window estimator using (1)
```

MB-GRNN: Pseudocode for smoothing parameter determination



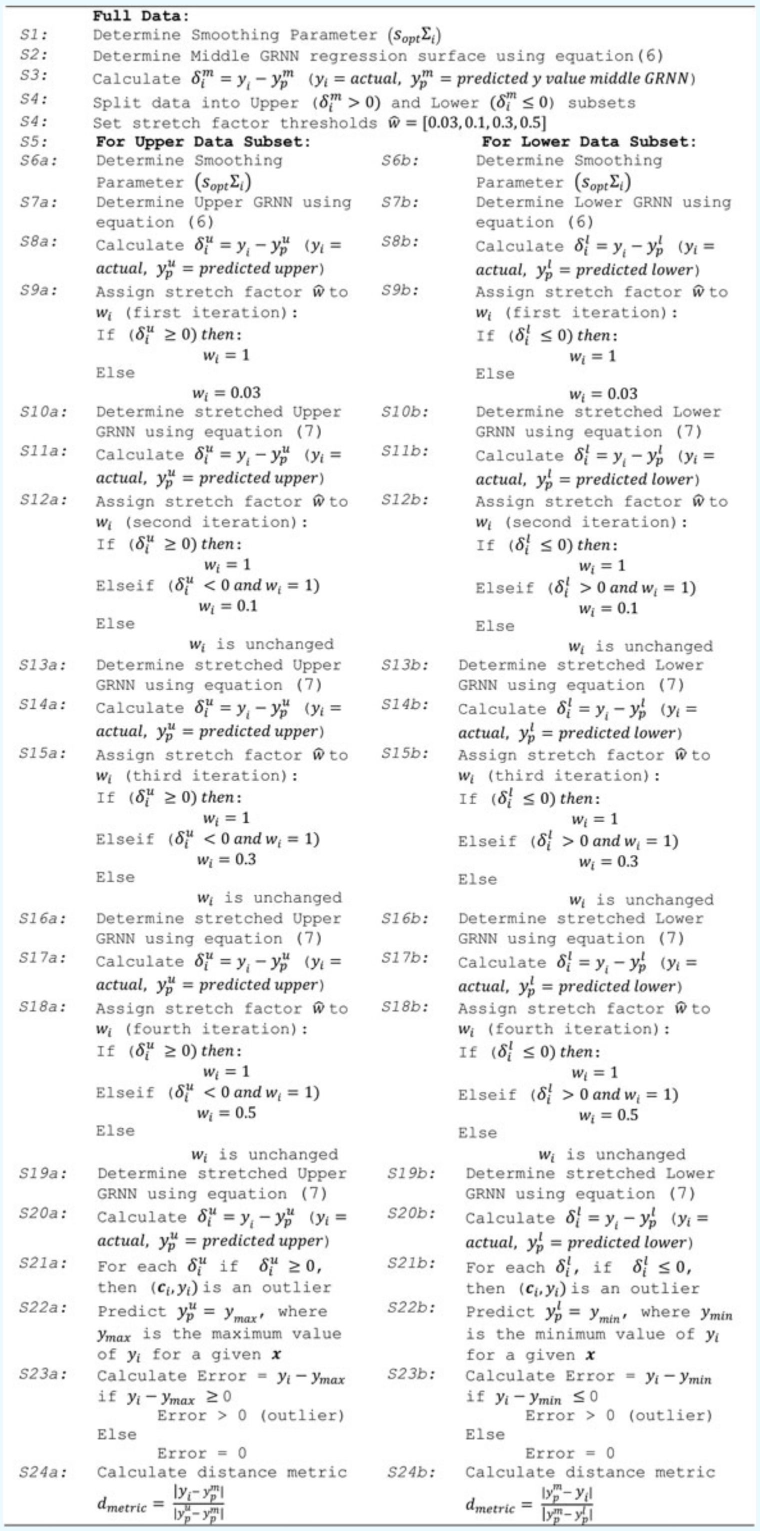
```
S1:  Select independent variable $\boldsymbol{x}$ dependent variable $y$
S2:  For each variable, convert data to log values and normalise data in the
     range = [0,1] using $x_{inorm} = \frac{(x - x_{imin})}{(x_{imax} - x_{imin})}$
S3:  For $\boldsymbol{x}$, evaluate optimum clusters $z$ using LBG vector quantisation algorithm as
     shown in Section A.3
S5:  Define a range of neighbours $n \in [n_1 : \Delta n : n_2]$ with $1 \le n \le z$ (Refer Table A.4)
S6:  Define a range of scaling factors $s \in [s_{low} : \Delta s : s_{up}]$ (Refer Table A.4)
S7:    For each $\boldsymbol{n}$:
S8:        Determine covariance matrix $\Sigma_i$ for each sample using equation (4)
S9:        For each $\boldsymbol{s}$:
S10:           Omit one sample $(\boldsymbol{c}_i, y_i)$ at a time, where $\{\boldsymbol{c}_i\} \subset \{\boldsymbol{x}\}$ and $y_i \subset \{y\}$ (LOOCV
               method)
S11:           For each omitted sample, determine predicted value $y_p$ using GRNN
               equation (6)
S12:           Calculate error $\delta_i = |y_i - y_p|$ for each omitted sample
S13:           Calculate mean squared error $[MSE_{ind}]$ for each $s$ using $((\sum \delta_i^2)/(N-1))$
S14:       Find $\boldsymbol{s}$ that results in minimum MSE = $(\min[MSE_{ind}])$ for a given $\boldsymbol{n}$
S15:       Store values [V] = $[(\min[MSE_{ind}]), s, n]$
S16: From V, find optimum $s = s_{opt}$ and $n = n_{opt}$ that results in global minimum of
     mean squared error $MMSE_{glb}$=$(min(\min[MSE_{ind}]))$
S17: Use $n_{opt}$ for determination of covariance matrix $\Sigma_i$ and use $(s_{opt}\Sigma_i)$ as smoothing
     parameter for determination of GRNN using (6)
```

Parzen window and MB-GRNN estimation

This section explains the determination of the upper and lower extremal GRNN boundaries by the MB-GRNN method and estimation of Parzen probabilities.

MB-GRNN: Pseudocode for the middle, upper, and lower GRNN boundaries estimation (Refer Step 4 in Fig. 1)

Table A.3. Pseudocode for the determination of stretched upper and lower GRNN boundaries



```
Full Data:
S1:   Determine Smoothing Parameter $(s_{opt}\Sigma_i)$
S2:   Determine Middle GRNN regression surface using equation(6)
S3:   Calculate $\delta_i^m = y_i - y_p^m$  $(y_i = actual,\ y_p^m = predicted\ y\ value\ middle\ GRNN)$
S4:   Split data into Upper $(\delta_i^m > 0)$ and Lower $(\delta_i^m \le 0)$ subsets
S4:   Set stretch factor thresholds $\hat{w} = [0.03, 0.1, 0.3, 0.5]$
S5:   For Upper Data Subset:
S6a:  Determine Smoothing Parameter $(s_{opt}\Sigma_i)$
S7a:  Determine Upper GRNN using equation (6)
S8a:  Calculate $\delta_i^u = y_i - y_p^u$  $(y_i = actual,\ y_p^u = predicted\ upper)$
S9a:  Assign stretch factor $\hat{w}$ to $w_i$ (first iteration):
      If $(\delta_i^u \ge 0)$ then:
              $w_i = 1$
      Else
              $w_i = 0.03$
S10a: Determine stretched Upper GRNN using equation (7)
S11a: Calculate $\delta_i^u = y_i - y_p^u$  $(y_i = actual,\ y_p^u = predicted\ upper)$
S12a: Assign stretch factor $\hat{w}$ to $w_i$ (second iteration):
      If $(\delta_i^u \ge 0)$ then:
              $w_i = 1$
      Elseif $(\delta_i^u < 0\ and\ w_i = 1)$
              $w_i = 0.1$
      Else
              $w_i$ is unchanged
S13a: Determine stretched Upper GRNN using equation (7)
S14a: Calculate $\delta_i^u = y_i - y_p^u$  $(y_i = actual,\ y_p^u = predicted\ upper)$
S15a: Assign stretch factor $\hat{w}$ to $w_i$ (third iteration):
      If $(\delta_i^u \ge 0)$ then:
              $w_i = 1$
      Elseif $(\delta_i^u < 0\ and\ w_i = 1)$
              $w_i = 0.3$
      Else
              $w_i$ is unchanged
S16a: Determine stretched Upper GRNN using equation (7)
S17a: Calculate $\delta_i^u = y_i - y_p^u$  $(y_i = actual,\ y_p^u = predicted\ upper)$
S18a: Assign stretch factor $\hat{w}$ to $w_i$ (fourth iteration):
      If $(\delta_i^u \ge 0)$ then:
              $w_i = 1$
      Elseif $(\delta_i^u < 0\ and\ w_i = 1)$
              $w_i = 0.5$
      Else
              $w_i$ is unchanged
S19a: Determine stretched Upper GRNN using equation (7)
S20a: Calculate $\delta_i^u = y_i - y_p^u$  $(y_i = actual,\ y_p^u = predicted\ upper)$
S21a: For each $\delta_i^u$ if $\delta_i^u \ge 0$,
      then $(c_i, y_i)$ is an outlier
S22a: Predict $y_p^u = y_{max}$, where $y_{max}$ is the maximum value of $y_i$ for a given $x$
S23a: Calculate Error = $y_i - y_{max}$
      if $y_i - y_{max} \ge 0$
              Error > 0 (outlier)
      Else
              Error = 0
S24a: Calculate distance metric
      $d_{metric} = \frac{|y_i - y_p^m|}{|y_p^u - y_p^m|}$

      For Lower Data Subset:
S6b:  Determine Smoothing Parameter $(s_{opt}\Sigma_i)$
S7b:  Determine Lower GRNN using equation (6)
S8b:  Calculate $\delta_i^l = y_i - y_p^l$  $(y_i = actual,\ y_p^l = predicted\ lower)$
S9b:  Assign stretch factor $\hat{w}$ to $w_i$ (first iteration):
      If $(\delta_i^l \le 0)$ then:
              $w_i = 1$
      Else
              $w_i = 0.03$
S10b: Determine stretched Lower GRNN using equation (7)
S11b: Calculate $\delta_i^l = y_i - y_p^l$  $(y_i = actual,\ y_p^l = predicted\ lower)$
S12b: Assign stretch factor $\hat{w}$ to $w_i$ (second iteration):
      If $(\delta_i^l \le 0)$ then:
              $w_i = 1$
      Elseif $(\delta_i^l > 0\ and\ w_i = 1)$
              $w_i = 0.1$
      Else
              $w_i$ is unchanged
S13b: Determine stretched Lower GRNN using equation (7)
S14b: Calculate $\delta_i^l = y_i - y_p^l$  $(y_i = actual,\ y_p^l = predicted\ lower)$
S15b: Assign stretch factor $\hat{w}$ to $w_i$ (third iteration):
      If $(\delta_i^l \le 0)$ then:
              $w_i = 1$
      Elseif $(\delta_i^l > 0\ and\ w_i = 1)$
              $w_i = 0.3$
      Else
              $w_i$ is unchanged
S16b: Determine stretched Lower GRNN using equation (7)
S17b: Calculate $\delta_i^l = y_i - y_p^l$  $(y_i = actual,\ y_p^l = predicted\ lower)$
S18b: Assign stretch factor $\hat{w}$ to $w_i$ (fourth iteration):
      If $(\delta_i^l \le 0)$ then:
              $w_i = 1$
      Elseif $(\delta_i^l > 0\ and\ w_i = 1)$
              $w_i = 0.5$
      Else
              $w_i$ is unchanged
S19b: Determine stretched Lower GRNN using equation (7)
S20b: Calculate $\delta_i^l = y_i - y_p^l$  $(y_i = actual,\ y_p^l = predicted\ lower)$
S21b: For each $\delta_i^l$, if $\delta_i^l \le 0$,
      then $(c_i, y_i)$ is an outlier
S22b: Predict $y_p^l = y_{min}$, where $y_{min}$ is the minimum value of $y_i$ for a given $x$
S23b: Calculate Error = $y_i - y_{min}$
      if $y_i - y_{min} \le 0$
              Error > 0 (outlier)
      Else
              Error = 0
S24b: Calculate distance metric
      $d_{metric} = \frac{|y_p^m - y_i|}{|y_p^m - y_p^l|}$
```

Parzen window p.d.f estimate (Refer Step 4 in Fig. 1)

Parzen window probabilities are estimated using optimum smoothing parameter $s_{\text{opt}}\Sigma_i$ explained under the section "Parzen window: Pseudocode for smoothing parameter determination" and Eq. (1). The results are ranked from lower to higher probabilities. The data points having lower probabilities (with higher ranking) indicate potential outliers in the domain.

Parameters and outlier detection (Refer Step 5 in Fig. 1)

This section describes the parameters used by the algorithm and how results are used for outlier detection.

The initialization parameters and optimization results

Table A.4. Parameters used by the algorithm for datasets in Table A.2

Method	Parzen	Middle GRNN	Upper GRNN	Lower GRNN
Clustering	epsilon = 10^{-3}, threshold = 10^{-3}	epsilon = 10^{-3}, threshold = 10^{-4}		
Neighbors n	$n_1 = 5$, $n_2 = 15$	$n_1 = 7$, $n_2 = 18$	$n_1 = 7$, $n_2 = 15$	$n_1 = 7$, $n_2 = 15$
Scaling factor range	$[s_{\text{low}} = 0.1{:}\ \Delta s = 0.05{:}\ s_{\text{up}} = 5]$			
Scaling factor s_{opt}	2.150505 for $n = 6$	0.495959 for $n = 13$	0.693939 for $n = 12$	0.545454 for $n = 13$
Initial stretch factor w_i	N/A	N/A	$w_i = 1$	$w_i = 1$
Stretch factors $\hat{w}$	N/A	N/A	[0.03, 0.1, 0.3, 0.5]	[0.03, 0.1, 0.3, 0.5]

Experimental results and outlier detection

The data points ($\boldsymbol{c}_i$, y_i) with Error >0 (as per S23a and S23b of Table A.3) identify points outside of the upper and lower GRNN extremal boundaries and hence classified as outliers in the range by MB-GRNN. The higher-ranking data points having lower Parzen probabilities indicate outliers in the domain. Table 2 and Figure 8 show how a combination of these methods can be used to identify outliers for both in the range and domain. Results and analysis obtained from this experiment are explained further in the section "Experimental results and analysis".

The MB-GRNN can also be used to identify outliers based on a reliability matrix by calculating a distance metric from middle GRNN (as per S24a and S24b of Table A.3). For any given $\boldsymbol{x}$ value, a distance metric for each y value was computed using a normalized distance spanning the Middle GRNN as 0 and the Upper or Lower GRNNs as 1. Hence, data points that lie outside the upper and lower MB-GRNN boundaries show $d_{\text{metric}} > 1$. Figure A.1 shows the outliers detected by MB-GRNN using a distance metric. Blue solid lines show the $d_{\text{metric}} > 1$ and the orange dashed lines show $0.9 > d_{\text{metric}} > 1$.

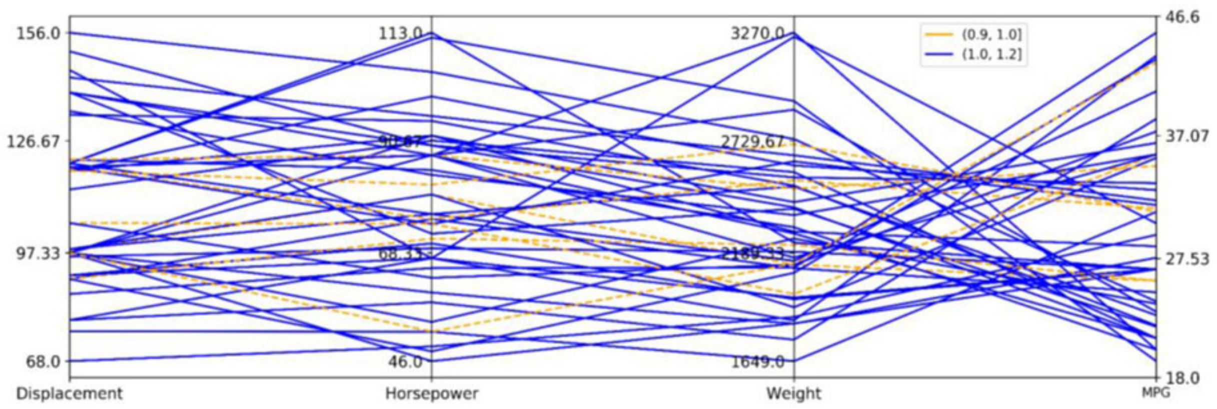


Fig. A.1. Parallel coordinates plot indicating outliers with distance metric >0.9.