UNIVERSITY OF
FORWARD
THINKING
WESTMINSTER

WestminsterResearch
http://www.westminster.ac.uk/westminsterresearch

Optimised meta-clustering approach for clustering Time Series Matrices

Movahdisavehmotlagh, A.

This is an electronic version of a PhD thesis awarded by the University of Westminster.
© Mr Amir Movahdisavehmotlagh, 2018.

The WestminsterResearch online digital archive at the University of Westminster aims to make the research output of the University available to a wider audience. Copyright and Moral Rights remain with the authors and/or copyright owners.

Whilst further distribution of specific materials from within this archive is forbidden, you may freely distribute the URL of WestminsterResearch: ((http://westminsterresearch.wmin.ac.uk/).

In case of abuse or copyright appearing without permission e-mail repository@westminster.ac.uk

Optimised meta-clustering approach for clustering Time Series Matrices

by

Amir Movahdi Saveh Motlagh

A thesis submitted to the University of Westminster in fulfilment of the thesis requirement for the degree of Master of Philosophy

In

Computer Science

2018

Acknowledgement

I would like to give my deepest thanks to my supervisor Dr. Epaminondas Kapetanios for believing in me and supporting me throughout my research journey in every way he could.

I would like to also thank my parents for their love and moral support which was the strongest influence for me to continue my research work. I want to thank Dr. Mohammad Samie for providing me the data used in my project and giving me valuable suggestions

Finally, I like to thank all my friends and family for their help and encouragement during difficult times.

Abstract

The prognostics (health state) of multiple components represented as time series data stored in vectors and matrices were processed and clustered more effectively and efficiently using the newly devised 'Meta-Clustering' approach. These time series data gathered from large applications and systems in diverse fields such as communication, medicine, data mining, audio, visual applications, and sensors. The reason time series data was used as the domain of this research is that meaningful information could be extracted regarding the characteristics of systems and components found in large applications. Also when it came to clustering, only time series data would allow us to group these data according to their life cycle, i.e. from the time which they were healthy until the time which they start to develop faults and ultimately fail. Therefore by proposing a technique that can better process extracted time series data would significantly cut down on space and time consumption which are both crucial factors in data mining. This approach will, as a result, improve the current state of the art pattern recognition algorithms such as K-NM as the clusters will be identified faster while consuming less space. The project also has application implications in the sense that by calculating the distance between the similar components faster while also consuming less space means that the prognostics of multiple components clustered can be realised and understood more efficiently. This was achieved by using the Meta-Clustering approach to process and cluster the time series data by first extracting and storing the time series data as a two-dimensional matrix. Then implementing an enhance K-NM clustering algorithm based on the notion of Meta-Clustering and using the Euclidean distance tool to measure the similarity between the different set of failure patterns in space. This approach would initially classify and organise each component within its own refined individual cluster. This would provide the most relevant set of failure patterns that show the highest level of similarity and would also get rid of any unnecessary data that adds no value towards better understating the failure/health state of the component. Then during the second stage, once these clusters were effectively obtained, the following inner clusters initially formed are thereby grouped into one general cluster that now represents the prognostics of all the processed components. The approach was tested on multivariate time series data extracted from IGBT components within Matlab and the results achieved from this experiment showed that the optimised Meta-Clustering approach proposed does indeed consume less time and space to cluster the prognostics of IGBT components as compared to existing data mining techniques.

Contents

List of Table and Figures

1. Introduction

1-1- Scope

1-1-1. The importance of this research

A significant amount of time series data are generated daily, in areas as diverse as astronomy, industry, sciences, and aerospace, to name just a few. By using my approach to process and analyse these extracted time series data, leads to useful information to be obtained more effectively and efficiently as opposed to current methods and techniques. As such, developing innovative pattern recognition algorithms will help with better clustering large sized time series data (stored as vectors and matrices) by consuming less space and time, which are both critical factors when processing time series data related to large and sensitive applications such as the reliability of Boeing's systems and components.

1-1-2. Challenges of processing time series data

The intrinsic structural characteristics of time series data, such as the high dimensionality and feature correlation, combined with the measurement-induced noises that beset real-world time series data, pose challenges that render standard data mining algorithms ineffective and inefficient for time series, especially in the field of power modules. Another challenge that must be considered refers to choosing the right similarity measurement to distinguish or match patterns that are perceptually similar but not mathematically identical. Also, the indexing method utilised to organise a set of time series data to enable fast clustering is also a challenge as these data consume too much space and are complex. The following issues mentioned above indicate why time series data mining has attracted an enormous amount of attention in the past two decades.

1-1-3. Case study: Industrial applications - 'IGBT Components'

The optimised pattern recognition algorithm tested on 'Insulated-gate bipolar transistor' (IGBT). IGBTs are important components found in many large systems which will be used as a case study for this PhD project. Therefore the algorithms and techniques proposed in this project will be tested on multivariate time series data extracted from multiple sensors monitoring the performance and life cycle of these IGBT components. The following time series data extracted were kindly provided by the University of Cranfield. The case study is described in much more depth within the results section on page 46.

1-2- Motivation

1-2-1. Impact of this research

The overall impact of this research is that it will enable the prognostics of several power modules to be clustered based on similarities shared. As a result, systems that are partially similar will be able to connect better and exchange patterns that closely resemble the health state of the IGBT component; hence avoiding the need to test the prognostics of such components individually which both consumes a lot of time and space. This will thus improve the overall reliability of IGBT components associated with large systems, as by efficiently processing the time series data related to the reliability of IGBT components will help avoid critical system failures, considering the sensitive nature of large applications such as Boeing.

1-2-2. Reasons and motivations directed toward hypothesis

What initially led me to pursue this research was that I wanted to propose a novel pattern recognition approach that can better identify and understand the prognostics of systems and components (in general) whilst consuming less time and resources as well as get rid of unnecessary data that adds no value to the overall implication of the results. This research carried out will immensely reduce the communication cost, hence saving us the trouble of having to deal with the failure profiles of a wide range of IGBT components. It does this by proposing an algorithm which clusters the time series data in two stages. The first stage creates a set of inner a cluster and the second stage groups the inner clusters into a general cluster. This procedure will thereby provide a set of refined clusters that get rid of meaningless data, hence saving space and time. Furthermore, using the proposed technique means that the health state of components does not need to be tested individually as multiple components can be clustered together if similar. Therefore, the implementation of an optimised pattern recognition solution will ultimately process and cluster the prognostics of IGBT components shared and matched across multi-variate time series data more effectively and efficiently, in contrast to standard data mining methods used before.

1-3- Hypothesis

The novel pattern recognition technique proposed will enable a large amount of time series data stored as vectors and matrices to be processed and clustered much more effectively and efficiently based on a 'Meta-Clustering' approach. As thus far, researchers have mainly dealt with processing the prognostics of such systems and components using standard data mining techniques such as k-means clustering, and Vector Quantization (VQ) techniques. K-means clustering basically partitions n objects into k clusters according to the nearest mean. VQ is a data compression type of technique which divides a broad set of objects in space into groups that have approximately the same amount of objects close to them. The meta-clustering approach differs from the two techniques mentioned above as it achieves the objective of these two methods more effectively and efficiently. This is because, contrary to k-means and VQ, the algorithm achieves faster clusters while also compressing the data as well. It does this via two stages; the first stage clusters the objects into a set of refined clusters which filters and gets rid of any data that is not meaningful. Then the second stage generalises all the refined clusters into a single optimised cluster. Therefore by developing such a novel Meta-Clustering method that can efficiently classify a set of failure profiles into similar groups at a faster rate will overall enhance the performance of various existing pattern recognition techniques.

1-4- Objectives and tasks

1-4-1. Obj1 Literature review

T1-1: Explore and review existing research that has been done so far in the field of time series data mining and pattern recognition.

T1-2: By doing so identify possible time series data mining and pattern search techniques that are most suitable for the case study of power modules.

1-4-2. Obj2 Understanding test data domain

T2-1: Achieving a good understanding of IGBT components.

T2-2: Once the application has been well understood, the meaning and information behind the extracted time series data and how it's presented as vectors and matrices will be explored in-depth for further processing

1-4-3. Obj3 Analysing and processing time series data

T3-1: Applying data pre-processing techniques on the time series data such as cleaning, normalisation, transformation and feature extraction and selection.

T3-2: Selecting the right set of features and patterns is crucial, as it will be used to accurately describe and cluster reliability information concerning the IGBTs while also getting rid of unnecessary data that are meaningless (precursor parameter).

1-4-4. Obj4 Development of the Meta-Clustering (pattern recognition) approach

T4-1: Implementing a modified K-Nearest Means (K-NM) clustering algorithm based on a meta-clustering approach which is based on first classifying and organising each power module within their own refined individual clusters in order select the most relevant set of failure patterns that shows the highest level of similarity and to get rid of any unnecessary data that adds no value towards better understanding the failure state of the power module.

T4-2: Then once these clusters have been effectively obtained, based on the proposed meta-clustering approach group the following clusters into one optimised cluster that represents the failure health state of all the processed components that were initially clustered separately.

T4-3: Identifying and understanding the failure profile of multiple IGBTS just by looking at one optimised cluster as a result of the novel meta-clustering solution proposed. The knowledge obtained from this optimised cluster is that the health state of all 22 components can be understood to determine whether they are faulty or not.

1-4-5. Obj5 Test and evaluation

T5-1: Applying the developed meta-clustering algorithms on the extracted time series data (see also Obj 2)

T5-2: Running simulations to test the proposed techniques and algorithms

T5-3: Using f-measure and other evaluation tools to compare the results obtained from the simulations against other pattern search techniques that also dealt with time series data.

T5-4: Assessment of results towards justification and validation of hypothesis

1-5- Deliverables

1-5-1. Gap of knowledge

Reviewing existing research will, therefore, familiarise us with the latest algorithms and techniques available and as a result, help us with identifying the gap of knowledge. The areas that will be mainly researched are unsupervised classification techniques. This refers to exploiting hidden structures related to an unlabelled dataset. Examples of common clustering techniques include hierarchical clustering, partitioning clustering (k-means), and density-based, grid-based and model-based clustering techniques. Once the latest research relating to the mentioned clustering techniques has been conducted, the areas of interest which have not yet been uncovered and require further research and experimentation will be identified.

Additionally, different distance measuring tools will also be researched and their strength and weaknesses outlined. This will help us choose the right distance measure for a given dataset as this is one of the most significant challenges. Therefore choosing the right distance measure will help in better measuring the similarity between a collection of data points in a cluster and how it is dissimilar to those points in other clusters. The majority of distance measures proposed in literature mostly consist of metric functions; Euclidean distance, Manhattan distance, Minkowski distance and Hamming distance. For non-numeric datasets, special distance functions are proposed. For example, edit distance is a well-known distance measure for text attributes. As a result, both the clustering technique and distance measuring techniques researched will better support with identifying the gap of knowledge.

1-5-2. Features and meaning of data

The completion of the second objective will enable us to devise an algorithm that can more effectively and efficiently cluster the failure profiles of these power modules based on the features and meaning obtained from the data. This involves thoroughly understanding the time series data extracted and realising which clustering or distance measuring techniques perform best on the dataset when tests and simulations are carried out. To be able to process these data successfully, we need to gain a basic knowledge of where this data came from, what this data represents and how can it be processed and clustered using enhanced pattern recognition techniques.

1-5-3. Enhanced and effective clustering of time series data

Pre-processing and filtering the time series data prepares it for effective clustering, as now the data is transformed to an understandable format and thus can be more precisely clustered based on the relevant patterns selected.

Clustering time series data to gather valuable information through forecasting and prediction is becoming extremely difficult to process in many applications, for example forecasting stock prices and electricity prices. What Time series forecasting involves is predicting the values of a variable changing continuously with a forecasting model based on historical data. Time series forecasting has gained a lot of attention and emphasis in recent times, considering the useful information it provides for many applications. Popular time series forecasting models proposed are quite diverse including Neural networks, Garch Model and Support Vector Machines (SVM). Neural Networks are commonly used to approximate time series data. Tests carried out on Garch model has shown to perform well, especially when dealing with the volatility in time series data related to stock prices.

Furthermore, SVM has also proven to provide excellent results when applied to time series forecasting.

When choosing a specific forecasting model, it is essential also to adopt a feature pre-processing technique to make forecasts more accurate. Pre-processing is the process of choosing the most relevant factors of a response variable while eliminating any variables that adds no value to the final forecast. This could as well mean constructing new features based on these factors. This is why feature pre-processing is very important for time series forecasting as these factors lead to accurate results since only relevant response variables will be selected and more space saved as it gets rid of a lot of the data that is removed. The consequence of not considering pre-processing is; running tests and simulations on a forecasting model that contain irrelevant features can indeed reduce the forecasting accuracy, and it will be challenging to determine relevant features that may have significant implications. Therefore, effective pre-processing techniques are needed to determine the most relevant feature sets or patterns. The following process is referred to as feature selection. Secondly, by not applying any pre-processing technique to the data could have an adverse effect on the performance of the forecasting model. Lastly, feature extraction techniques that are employed can be used for constructing new features that are independent of other features, and have less noise in the data.

There have been many studies revolving around feature selection/extraction especially in data mining, machine learning, and statistics which have been applied to many areas diverse areas including communication, medicine, data mining, audio, visual applications, and sensors. Many feature selection techniques are implemented as a pre-processing tool; hence selecting the relevant feature subset. Standard signal processing techniques that have been applied to time series data to reduce the noise in data are wavelet decomposition and reconstruction [1].

1-5-4. Accurate and timely prediction of the future health state of a component (prognostics)

By clustering the selected set of patterns into refined groups and further grouping all these selected clusters into one optimised cluster will provide meaningful and relevant information regarding the prognostics of the power modules much faster while using less space. Proposing new and innovative methods that can more effectively and efficiently achieve the health state of any electronic system or component is significant. This is because the safety and reliability of these large applications such as aircraft or driverless cars depend on the state of their internal systems and components. Therefore by collecting data from these systems and components via sensors and processing this information, will prevent any system/component faults that could potentially lead to the overall system application failing. The meta-clustering approach will be able to achieve the prognostics of such significant components related to large systems and applications more accurately and while consuming less time and space.

1-5-5. Validation of hypothesis

Performing these tests and evaluation measures will help towards validating the hypothesis. It will, as a result, show whether the optimised meta-clustering solution was capable of improving current pattern recognition techniques used for clustering time series data.

1-6- Research Methodology

1-6-1. Understanding the tested time series data

After having gathered the extracted time series data, the following data will be processed into vectors and matrices in order to prepare it for clustering. The following time series data were extracted and gathered from a set of sensors recording the performance from 22 IGBT components. The values inside the time series data represent the health state of the components recorded every 2 seconds from the point which they are healthy to the point which they eventually fail. Once the time series data have been collected, they are subsequently stored as vectors in a 2-dimensional matrix. This 2-dimensional matrix has 5130 rows and 22 columns. The 5130 rows reflect the data values (also referred to as failure patterns) obtained from the sensors recording the life cycle of the power modules every 2 seconds and the 22 columns refers to the total amount of components that were recorded for this simulation. This is thoroughly discussed in the later sections (section 3.3) whereby information regarding the meaning of these time series data and how they were further processed and clustered accordingly can be found in a published journal by me: IEEE transactions on power electronics journal, 2015 [38].

1-6-2. Applying various pre-processing and filtering techniques

Now that the data is understood, several pre-processing and filtering techniques are first applied to the extracted time series data in order to get the most predictively useful information from the data, reduce the dimensionality of the series and get rid of noise and outliers found in such data. Wavelet thresholding is used to filter out noise. Wavelet thresholding does this by calculating the discrete wavelet transform (DWT) of a signal passing the results of the wavelet coefficients for threshold testing. The coefficient results produced are subsequently used to reconstruct the signal, hence removing noise while losing little information [2]. Also, another pre-processing issue is the scaling differences between the time series, in which this can be resolved by applying linear transformation to the amplitudes as well as normalising a fixed range.

1-6-3. Devise Meta-Clustering approach

Devise a meta-clustering type of solution that can effectively and efficiently process and cluster the time series data. This is done by implementing an optimised k-nearest mean clustering algorithm based on this notion of meta-clustering, whereby a general cluster is formed based on inner clusters. The Euclidean distance function will be used to measure the similarity established between the patterns within the inner clusters as well as the optimised general cluster that is formed from the similarity shared between the 22 inner clusters.

1-6-4. Applying pattern recognition approach to much larger systems and applications

The developed techniques are mainly tested with failure data of IGBT components; therefore must test the scalability of the clustering algorithm against much larger and more complex systems using ‘Big O Notation’. Big O Notation will help describe the performance and complexity of the proposed meta-clustering algorithm to see whether the speed and accuracy of the algorithm are affected due to the scale or size of the system/component.

1-7- Risks and Mitigations

1-7-1. Risks

- Access to real data from experimental results, collecting real data requires expensive test rig and expensive tools
- Accuracy and consistency of pattern recognition results
- Experimental measurements are mainly conducted in labs which are far from real working conditions
- Difficulty in getting rid of noise at early stage of failure due to intermittency issues in the systems
- High dimensionality of data leads to the data becoming extremely sparse and thus difficult to find
- Failure data from real systems have random nature so that failure patterns may occur at different interval and different amplitudes

1-7-2. Mitigations

- Universities operating in Europe are working on test rig of power modules where I have already been able to collaborate with them to access such data. There are also experimental results available on the internet such as NASA which are open source
- Testing different pattern recognition techniques based on their strength and weaknesses and suitability regarding the extracted time series data.
- Work on various different data collected from different research centres i.e. Cranfield, Manchester, and NASA. Statistical analysis to better forecast variation of conditions, modules, and experiments
- Employing various filtering techniques that exist and trying to select the most applicable in regards to our data and analysing data based on different intervals, not specific sample/time
- Try to find the best set of features that best covers the systems health state
- Using different techniques such as time warping to identify the shift in pattern

1-7-3. Limitations and Conditions

- Cannot personally produce the data based on the conditions that I need for my pattern recognition solution.
- Another condition is we are only considering single power modules (IGBT)

2. Literature Review

2-1- Data Mining

2-1.1 Different types of data mining techniques

Data mining (or in some cases also referred to as machine learning) is divided into two distinct categories: supervised and unsupervised techniques. Supervised techniques focus on classifying a set of data from a labelled database whereas with unsupervised techniques; not much information is known about the data (i.e. unlabelled), as the data is clustered into different classes based on the patterns exploited. In regards to our research, since there is not much known about the time series data and given that the data are unlabelled, unsupervised techniques will be used to effectively process and cluster failure patterns that are similar.

- Supervised

To provide an insight into supervised techniques, classification seeks to assign labels to each series of a set. The main difference, when compared to the clustering task, is that classes are defined in advance and an example dataset is normally used for the algorithm to be trained. The ultimate objective is to observe and learn the distinct features of the dataset that distinguishes the classes from one another. Then, once an unlabelled dataset is entered into the system, the algorithm will thereby be able to automatically determine which class each series belongs to.

Classification techniques can be defined as given an unlabelled time series T, assign it to one class ci from a set C = {ci} of predefined classes. The tasks involved with supervised classification include training a classifier and labelling new time series. Keogh and Pazzani proposed a more advanced piecewise representation which effectively handled noise and weighting issues by applying a relevant feedback framework [3]. Singular Value Decomposition was introduced by Jeng and Huang to deal with the issue of high dimensionality by being able to only select the essential frequencies. This approach; however, lead to higher computational costs [4]. Three other types of classifiers were also compared by Rodriguez and Kuncheva including nearest neighbour, support vector machines and decision forests. The results obtained indicated that all three methods were valid, though it highly depended on the dataset at hand [5]. One of the most widely used classifiers is 1-NN classification algorithm with Dynamic Time Warping (DTW), as on a general basis it was highly accurate. However, due to the repeated DTW computations, speed does get affected.

- Unsupervised

Clustering is the process of finding natural groups among data where not much information is known about them in a dataset, called clusters. The main task of the clustering algorithm is to automatically find clusters that are distinct from other clusters. So the ultimate goal of clustering when applying it, for instance to each complete time series in a set is to regroup an entire time series into clusters, to the point where the time series are as similar to each other as possible within each group.

The clustering task can be defined as given a time series database DB and a similarity measure D (Q, T), find the set of clusters C = {ci} where ci = {Tk | Tk ∈ DB} that maximizes inter-cluster distance and minimizes intra-cluster variance. Typically speaking, once an adequate distance function is devised, any algorithm can be amended to it, depending on the field of study. Common clustering techniques are often performed using Self Organizing Maps (SOM), Hidden Markov Models (HMM) or Support Vector Machines (SVM) [6].

The models mentioned above are vulnerable to some scalability problems. A thorough survey discussing generic clustering algorithms from a data mining perspective mainly focuses on different

clustering techniques that can be applied to time series data [7]. These clustering techniques can be divided into five categories: hierarchical, partitioning, density-based, grid-based and model-based. The best instance for clustering time series data is to use it for data streams. Examples of specific clustering algorithms commonly implemented for time series data include Auto-Regressive (AR) models [8], k−Means [9] and – with greater efficiency – k-center clustering [10].

2-1.2 Comparison between Unsupervised and Supervised learning

In this section, both types of methodologies used to organise a group of patterns into distinct groups will be analysed and compared and their strength and weaknesses will be discussed. Applying cognitive reasoning into a computer system can address many issues, for example using mapping techniques such as pattern recognition and classification. These different types of models are essentially presented through Artificial Neural Networks (ANN). ANN is basically mathematical models that describe a function. The learning algorithm devised is the secret behind the main functionality of these models which functions as a step by step method attempting to imitate human actions. The three parameters of ANN consist of; (A) a network that forwards feeds and is recurrent. This affects the interconnection property of the neural network. (B) A Classification model, Association model, Optimization model and Self-organizing model. This affects the application function. Lastly (C) choosing between supervised or unsupervised techniques. This reflects the learning rule adopted by your system.

Artificial Neural Networks have their own distinct advantages. Both practically and theoretically, ANN can be applied to various applications that have different implications. In recent research, the main focus of ANN has been mostly targeted towards pattern classification. This is because ANN can efficiently perform well structured and organised classification tasks due to its structural design and learning methods. An algorithm's design changes when training ANN models because what an algorithm learns and it's degree of inference different from each other. Therefore in this section, supervised and unsupervised learning will be evaluated and the performance of different types of classification methods will be compared in a specific example.

The items that will be discussed in this section are as follow, first of all, multiple learning algorithms that are used in ANN for solving pattern classification problems related to unsupervised and supervised learning will be introduced. Then classification will be discussed and its requirements will be pointed out. This will focus on discussing the distinction between supervised and unsupervised learning on the pattern-class information. Then a specific example will be used to explain the foundation for constructing a classification network. The final results of the two algorithms used in the study will be shown from two different perspectives. Lastly, the journal analysed concludes by providing their final insight on supervised and unsupervised learning algorithms when tested on an educational classification scenario.

In regards to utilising learning algorithm, this can either refer to gaining knowledge or enhancing the knowledge already acquired. A quote given by Herbert Simon describes Machine Learning as alterations happening in a system which in a sense allows the system to do a repeated task or multiple tasks gathered from the same number of users more efficiently at consecutive times.

The three main learning paradigms of ANN can be categorised as unsupervised, supervised and reinforcement learning. Unsupervised learning model involves identifying the pattern class information based on unlabelled data (i.e., not much information is known about the data). On the other hand, the supervised learning model already has some knowledge of the data and based on this information it classifies the training examples into different groups. Reinforcement learning is based on a trial and error concept which learns and adapts through the interactions it has with the environment. Even though these learning models take different approaches to address learning, learning is still highly dependent on the space of interconnection neurons. What this means is supervised learning learns by using the help of signal faults to adjust the weight of its interconnection

weight combinations. Unsupervised learning, however, takes advantage of the information associated with a group of neurons and reinforcement learning changes its local weight parameters by utilising a reinforcement function

Therefore, ANN learns by adjusting the parameters of the network that are altered to where the ANN is embedded. The following adjustments made to the parameters are vital when trying to differentiate between the learning algorithm such as supervised or unsupervised models and reinforcement learning. It is also important to note that such learning algorithms are facilitated by number different learning rules.

Supervised learning analysis

Supervised learning basically revolves around training a dataset derived from a data source which should already have the relevant classification technique assigned. These classification techniques are usually used in Multilayer Perceptron (MLP) or feedforward models. The characteristics associated with MLP include: the input and output layers of the network may have one or more layers of hidden neurons that are completely unrelated to each other. This will thus enable the neural network to learn and address any problematic issues. The other characteristic of MLP refers to the nonlinearity found within its neural activity is differentiable and the fact that the interconnection model of the network can demonstrate a high level of connectivity.

Therefore, the mentioned characteristics, as well as training the dataset for learning lead to diverse and complex problems to be solved. The concept of learning based on training data within a supervised ANN model is also known as error back-propagation algorithm. What this algorithm does is it trains the network on the basis of the input and output samples and later uses those samples to find error signals. This is achieved based on the comparison made between the difference of the output calculated and the desired output. The weights of the neurons proportional to the error signal and the input of the synaptic weight are adjusted accordingly. The following foundation enables error back propagation learning to be performed in two phases, a forward and backward pass. The former presents the input vector to the network which propagates forward one neuron at a time through the network comes out of the output end of the network as an output signal. The output calculated reflects output layer which is compared against the desired response and as a result identifies the error related to the neuron. The synaptic weight associated with the network during the forward pass does not change.

In regards to the Backward Pass, this refers to the error signal created at the output neuron of that layer which does the opposite and propagates backward through the network. The local gradient of each neuron is successively calculated for each separate layer. This thus allows the synaptic weights of the network to adjust accordingly. This recursive process continues like a loop, which basically entails the forward pass

This recursive computation is done for each input pattern until the network has fully converged. Supervised learning related to ANN is an effective tool as it finds good solutions for a number of linear and non-linear problems which include classification, prediction, forecasting, robotics etc.

- Classification

Classification is when a pattern/object needs to be appointed to a predefined group. This grouping is done based on multiple attributes associated with that object. Classification is used to address a diverse range of industrial problems such as; system failure prediction, bankruptcy prediction, stock market prediction, weather forecasting, Medical diagnosis, Speech recognition, Character recognitions etc. The solutions for addressing the mentioned classification problems can be either solved mathematically or in a nonlinear form. An issue of solving such classification problems

mathematically refers to the impact on accuracy as well as how widespread the data is distributed and the capability of the model utilised.

According to recent research conducted, ANN has been considered as the most effective classification model. This can be due to it accepting non-linear sequences, adaptable and having functional approximation principles. The way a Neural Network classifies an object is based on the output activation. This means when a number of patterns give as input are provided to the network, the nodes which are inside the hidden layers of the network retrieve the features of the pattern provided. To explain this as an example, assuming the ANN model has 2 hidden layers, the nodes that lie within the first hidden layer creates a boundary between the group of patterns are and the hidden nodes which reside in the second layer. The second layer hence creates a decision area consisting of the hyperplanes that were created in the previous layer. This, therefore, leads to the nodes in the output layer being combined with the decision area made by the nodes in the hidden layer. This will, as a result, organise them into group 1 or group 2 based on the number of classes defined during the training stage. Likewise to SOM, classification occurs according to the features extracted. This means transforming the input pattern observed as m-dimensional into q-dimensional feature output space and furthermore classifying a set of objects on the basis of the similarity it shares with the input pattern. The architectural frameworks of the most popular supervised and unsupervised learning algorithms will be presented using a pattern classification scenario. This will help discuss the efficiency of these models which will be tested on an educational sample dataset. Although education is different to the dataset used in this project, it will still provide a better insight on the type of measures that must be undertaken to ensure the learning algorithm utilised helps achieve more efficient and accurate results, regardless of the application.

The purpose of a classification system has always been to achieve a functional connection between the class association and the attributes of the object. For example, classifying multiple students in a specific course based on their grades is commonly recognised as a classification problem. In recent times, higher education has started to gather a significant amount of attention. This is mainly because of the competitive environment of high schools as both the students as well as the educational institutions have reached a crossroad when trying to assess the performance and ranking accordingly. Each higher education institution is trying hard to maintain its high rank by attempting to group students with great potential and a high skill set together. This will help the students improve their performance and hence lead to the overall institution to improve their own rank. The following scenario is thus considered as a classification problem, whereby the two learning algorithms will be studied to see how it will address the classification problem.

In the process of addressing a classification problem associated with any ANN model, the main focus should be trying to learn the observed objects in space. This paper analysed seeks to observe many of the pattern classification attributes related to the two algorithms. This reflects the proposed Supervised ANN and Unsupervised ANN constructed to address the problem described above. The dataset that will be used to test the supervised and unsupervised classifications consist of 10 important attributes. These attributes entail important information about the academic scores related to students', their mathematical knowledge, as well as the scores achieved from their eligibility test which is carried out by the educational institution. There were three classes of groups identified based on the input observations. The architectural depiction of the ANN model, along with the training process and the results achieved related to the learning ANN model will be presented.

Unsupervised learning analysis

Self-Organising Maps (SOM) is an unsupervised learning algorithm which basically discovers hidden patterns from unlabelled input data. Unlike supervised learning, unsupervised is capable of learning and organising information without having to provide an error signal to calculate a possible solution. Though unsupervised learning does not have much direction or structure, it does have its own credits. This is since it lets the algorithm go back and look for patterns that have not been previously

considered. Some of the main features of SOM include: It can transform a signal pattern provided as input into a one or 2-dimensional map.

Another feature of SOM is that the network has a feedforward structure consisting of only one computational layer whereby the neurons in the network are presented as rows and columns. Also during each phase of representation, each input signal is held in the correct context and the neurons handling closely related portions of information are in short distance of each other and they exchange information through synaptic connections.

The layer used for computations is also referred to as a competitive layer. This is because the neurons found in the layer are recursively competing with each other to become active, hence explaining why this learning algorithm is called competitive algorithm. Unsupervised algorithms related to SOM function via three individual steps consisting of a competition step, a corporative step, and an adaptive step. The competitive step involves, for each declared input pattern x provided to the inner product with synaptic weight w is calculated. What happens next is the neurons found in the competitive layer tries to find a discriminant function that spreads competition in the neurons and the weight vector that has the least distance to the input vector measured with the Euclidean distance is determined as the match in the competition. The following neuron is said to be the best matching neuron.

The best matching neuron determines the centre neighbourhood of cooperating neurons. This is done by the lateral interaction of the centre neuron amongst the cooperative neurons. And lastly, the adaptive step allows the best matching neuron and its corporative neurons to increase all their values of the discriminant function which reflects providing the input pattern with suitable synaptic weight adjustments. The Self-Organizing Model can be described as having natural neuro-biological behaviour, which explains why it is used in many real-world applications. Examples of such applications are quite diverse such as clustering, speech recognition, vector coding, texture segmentation, etc.

- Correlation Clustering

An example of clustering method commonly used in data mining and machine learning which is quite unique in contrast to other unsupervised techniques is correlation clustering. This form of clustering works in situations where the relationships between the objects are well known. Unlike other clustering methods such as k-means clustering, it does not require the algorithm to choose a number of clusters during the initial stages. How correlation clustering operates is given a weighted graph in which the weight of the edge determines whether two nodes are similar (positive edge weight) or different (negative edge weight), the challenge would be to find a clustering solution that maximises the positive edge weights within a cluster plus or minimises the negative edge weights within a cluster plus. Generally speaking, it is very difficult to have perfect clustering where all similar objects are grouped in a mutual cluster while all dissimilar objects are grouped in different clusters. What you can do to ensure effectiveness is to maximise the agreements while minimizing the disagreements. This consequently leads to the nondeterministic polynomial problem (NP) which refers to the multiway cut problem and the other problem of being able to partition it into triangles. A very well known correlation clustering algorithm refers to the KwikCluster. This algorithm consecutively clusters vertices that are neighbours, and obtains a 3-approximation ratio. One of the drawbacks of this clustering algorithm is that it requires a large number of consecutive clustering rounds, which could thus lead to a potential overhaul when dealing with large graphs [11].

Experimental observation (Supervised ANN vs Unsupervised ANN)

In regards to the Supervised ANN, a fully connected multi-layer perceptron (MLP) was constructed with error back-propagation learning algorithm. 300 dataset was extracted from the domain and out of the 300 datasets; only 50 were used for running tests to determine the performance of the system. The algorithm randomly selects a pattern and presents that pattern to the input layer along with providing the desired output to the output layer. At first, each of the synaptic weight vectors related to the neurons is randomly assigned. It is then randomly altered during backward pass according to the local error, whereby during each epoch the values are normalised. A non-linear activation function utilised refers to 'Hyperbolic tangent function'. A number of different learning rates were experimented and finally assigned between the ranges of (0.05 - 0.1). This, as a result, implements the sequential mode of backpropagation learning. The point where the learning algorithm converges is tested with average squared error per epoch that lies in the range of (0.01 – 0.1). The input patterns are grouped into three output patterns which can be accessed in the output layer. The trial and error procedure used to create the ANN architecture is shown below Table 1.

No. of hidden neurons	No. of Epochs	Mean-squared error	Correctness on training	Correctness on Validation
3	5000 - 10000	0.31 – 0.33	79%	79%
4	**5000 - 10000**	**0.28**	**80% - 85%**	**89%**
5	5000 - 10000	0.30 – 0.39	80% - 87%	84%

Table 1: Supervised Learning observation

Moving onto the Unsupervised ANN based on Self Organizing Model (SOM), this model entails an unsupervised ANN that is constructed with 10 input neurons and 3 output neurons, respectively. The dataset made use of for the supervised model is utilised for training the network. In order to modify the model for adaptability, the synaptic weights are initialized with $1/\sqrt{}$ referring to the number of input attributes to initially have a unit length. The outcome achieved is dependent on how the pattern is presented to the input vector to use for training the data. This means the patterns that have gone through training will be presented to the neural network sequentially. The Euclidean distance measure was calculated during each iteration in order to find the best neuron. The learning rate parameter which was set to 0.1 had decreased over time but did not go below 0.01 as it maintained at the convergence phase. Since the competitive layer consists of a one-dimensional vector that has three connected neurons, the neighbourhood parameter does not necessarily affect the activation. The network convergence could only be calculated when there were no changes happening anymore in the adaptation. Table 2 outlines the results:

Learning rate parameter	No. of Epochs	Correctness on training	Correctness on Validation
.3 - .01	1000 - 2000	85%	86%
.1 - .01	**1000 - 3000**	**85% - 89%**	**92%**

Table 2: Unsupervised learning observation

Results and Discussion

The classification process used for grouping the students were categorised under certain characteristic. The first group observed consisted of the students who have excellent academic score and eligibility score. The other class contained students who came from an underprivileged quota and students who did not have good academics were in another class.

From the observations carried out, the results of the first two were better when unsupervised learning algorithms were used for classification. This is because the correct matching percentage was higher in contrast to the supervised learning algorithm. Even though the result achieved was not significantly different and if the supervised algorithm had one extra hidden layer it could have increased its correctness percentage, the time taken however to construct the network itself was longer compared to unsupervised learning algorithms such as SOM. Some of the problems confronted and addressed as a result of back-propagation algorithm refer to the network size. In general, regardless of the linear classification problem, a hidden layer is not essential. However, the input patterns require at least 3 classifications, which refer to a trial and error. Furthermore, selecting the right number of neurons in the hidden layer is another problem the experiment encountered. Table I shows that the performance of the system was calculated on the basis of the number of neurons found in the hidden layer, in which case this study chose 4 hidden neurons since it provided the most optimal result. The other problem faced was the local gradient descent. This is important as the gradient descent is used to minimise the output error by gradually changing the weights. By changing the weight vector causes the error to get stuck in a range and thus prevents it from further decreasing. This problem was addressed by initialising weight vectors randomly, and after each iteration, the error of current pattern is used to update the weight vector.

Another problem encountered in this process refers to stopping criteria. Generally speaking, the training conducted on ANN models are stopped once it realises all the patterns are successfully established. How understands this is by calculating the total mean squared error of the learning. It is unfortunate that the total error of the classification consisting of 4 hidden neurons is 0.28, which could not be further decreased. Every time it was attempted to reduce the minimum, the validation error started to increase. This forced them to stop the system based on the correctness of the validation data which came to about 89% as shown in the table. In the process of adding one more neuron in the hidden layer just like in the last row of Table 1, will raise the possibility of overfitting the trained dataset while encountering less performance when validating.

A training problem that had also to be considered when training the KSOM refers to finding a way to declare the learning rate parameter as well as its reduction. They managed to decrease this over a period of time significantly. They also tried to train the system to learn different parameter set up and content with 0.1 to train and 0.01 at convergence time as depicted in Table 2. Also, in contrary to the MLP model used for classifying objects, the unsupervised learning algorithm KSOM uses a single-pass for learning which is potentially faster and more accurate as opposed to multi-pass supervised algorithms. This means the KSOM unsupervised algorithm is much more suitable for dealing with classification problems. Considering classification plays a significant role in helping humans with active decision-making tasks, the following classification techniques proposed in this paper might help educational institutions to mentor their students better and overall improve their performance. As much as classification helps the institution itself, it just as well enables students to become aware of their lack of domain and way in which they can improve a particular skill which will thereby be suitable for the student and the institution.

Summary

The paper analysed developed a classification network of given input patterns based on learning from observation. The following observations made it possible to initialise a new class and allow one to assign patterns from a new class to an existing class. This type of classification identifies new theories and knowledge which is embedded in the input patterns. The learning behaviour related to the neural network model improves the properties of the classification classes. The paper used two different learning algorithms such as supervised and unsupervised learning. It then experimented with the properties reflecting the classification of students based on how well the students performed during their period at the institution. The study found out that even though the error back-propagation supervised learning algorithm is very fast and efficient for many non-linear real-time problems, in the case of student classification, an unsupervised model such as KSOM performed much more efficiently than supervised learning algorithm [12].

2-2 Latest pattern recognition techniques for processing time series data

2-2.1 Pattern recognition algorithms

Pattern recognition algorithms focus on identifying and recognising patterns in data which are in either trained from labelled or unlabelled data (Supervised learning) and (Unsupervised learning). Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to perform "most likely" matching of the inputs, taking into account their statistical variation. The pattern recognition algorithm used depends on the data and the ultimate goal. Unsupervised types of pattern recognition algorithms will be researched since our data consist of unlabelled data, i.e. not much information is known about the data. This section of the project will describe, compare and evaluate existing clustering algorithms, and will also touch on how the proposed Meta-clustering approach addresses the drawbacks faced by some of these techniques. The four algorithms investigated include the k-means algorithm, hierarchal clustering algorithm, self-organisation map (SOM) algorithm and expectation maximization (EM) algorithm. The following algorithms were selected as they are commonly used to process time series data, they are flexible, they have been applied in various experiments and data sets and they have been often used to handle high dimensional data. The datasets used to test the clustering algorithms and compare the results were obtained from time series data collected from http://www.kdnuggets.com/datasets/index.html. The datasets used for testing varied between 600 rows and 60 columns vs 200 rows and 20 columns in order to study the impact of size on the data set i.e. large and small on the algorithms.

- K-nearest mean clustering algorithm (K-NM)

A cluster is generally thought of as a group of items (patterns, objects) in which each item is closed. Clusters can be viewed as “high-density regions” of some multidimensional space. Cluster analysis methods aim at partitioning n observations into k clusters in which each observation belongs to the cluster with the nearest mean. Cluster analysis is another but different from those in other groups. A popular heuristic for k-means clustering is Lloyd's algorithm. A simple and efficient implementation of Lloyd's k-means clustering algorithm is called the filtering algorithm. This algorithm is easy to implement, requiring a kd-tree as the only major data structure. The efficiency of the filtering algorithm is established using a data-sensitive analysis tool. The data-sensitive analysis tests the algorithm's running time, which shows as the separation between clusters increases the algorithm runs faster [13].

Researchers have shown special interest for speeding up the K-Means algorithm, by either reducing the computation complexity or by adopting K-Means implementations for parallel and distributed platforms. An efficient implementation of the Lloyd’s (K-Means) algorithm called the filtering algorithm which employs kd-trees for storing the data elements was proposed. This algorithm which reduces the computations for determining the closest centroid of a data element, based on an observation that if a data element gets closer to the centroid defined at the previous iteration, it won’t switch the cluster it belongs to. There are also other methods which focus on parallelizing the K-

Means algorithm and taking advantage of powerful parallel and distributed environments, addressing issues specific to those environments, like data availability, synchronization, etc., and adapting the K-Means algorithm to different distributed architectures (client-server, peer-to-peer, etc). The results were satisfactory for very big data sets [14].

What makes the k-means algorithm popular is its time complexity O(nkl), where n refers to the number of data points, k is the number of clusters and l shows how many times the algorithm iterates in order to converge all the objects together. Therefore since k-means is linear, it often makes it scalable and efficient in processing large data sets. Also, its space complexity is modest because only the data points and centroids need to be stored. In reality, the storage required is O((m+k)n), where m is the number of objects (data points) and n is the number of attributes considering n-dimensional objects. The algorithm is also ordered independently, so no matter in which order the data is presented, it will still produce the same set of partitions.

When using K-means as your clustering method there some considerations to make. These include the ability of K-means to compute the clusters fast even provided that the data set is Large. This is since the time complexity is O(nkl) where n is the number of patterns, k is the number of clusters and l is the number of the iterations. Also it important to consider that it heavily relies on the distance measure chosen. For instance, choosing Euclidian distance means it will work well with numeric values with interesting geometrical and statistic meaning. Furthermore, the initial assumption of the value selected for K is very important, however, there isn't a formal description that determines the value for K. Therefore different values set for K will output different numbers of clusters. The centroids chosen at the beginning of the algorithm is also important because if the centroid selected is very distant from the cluster centre of the data itself then it will cause the algorithm to go into continuous loops, hence leading to incorrect clustering. Lastly, K-means clustering struggles to effectively handle noise found in the dataset [15].

A drawback of the K-means algorithm is its computation time that can be seen from most standard algorithms. The drawback is that distance is calculated during each iteration while knowing that in some iteration's the data object will still belong to the same cluster. If data object x remains z times in the same cluster then algorithm takes x(k-1) time more to execute the algorithm [16].

- Hierarchal Clustering

There are different types of classification approaches established over the years as benchmarks in effectively organising and grouping your specific data domain. Hierarchical clustering is an example of one that produces groups of individual partitions while having the single main clustering point at the top and having the rest of the other points at the bottom. These points from bottom to the top get eventually merged together or divided depending on the level of similarity/dissimilarity shared by the clusters. MIN or single link is a hierarchical technique where the proximity of two clusters is defined to be of maximum similarity and minimum distance between any two points in the different clusters. This is why this technique is named as a single link since at the start all the points are represented as

singleton clusters, and then links are added between the points. The links added starts with the strongest links being added first and it is, therefore, these single links which combine the points into clusters.

In contrary to top-down approaches, the bottom-up approaches tend to be the most accurate but face higher computational cost. However, the increase in computation cost does mean that it is conceptually or algorithmically more complicated. This is because the clustering process is based on a hierarchy format which can be organised as a number of basic cluster merging. Hierarchal clustering can be divided as follow: Agglomerative hierarchical clustering which is a bottom-up approach. The procedure initially identifies an object which represents a cluster of its own, then similar clusters are repeatedly merged to the point where the desired cluster structure is achieved. This algorithm containing N samples starts with N clusters. Each N cluster has in itself a single sample. Then the two clusters sharing the highest similarity will be merged together until the number of clusters is transformed to a single cluster or a specific number of clusters chosen by the user. Three criteria's used to implement this algorithm include min distance, max distance, average distance, and centre distance.

The Divisive hierarchical clustering (top-down approach) is the opposite of the bottom-up approach. This algorithm involves the objects at first belonging to a single based cluster and then after a number of iterative partitions into clusters, they are further sorted into sub-clusters. Hierarchical algorithms are generally of great interest for a number of application domains and are thought to produce good quality clusters. The advantages of Hierarchal clustering is mainly attributed to the algorithm's flexibility in relation to it being able to handle any type of distance/similarity measuring tool and it being compatible with various attribute types. Nevertheless, there two main disadvantages for these types of methods: The first is in hierarchical clustering you cannot return to the clusters once they have been constructed. The reason this is an issue is that there are times where the results outputted are not correct and hence needs improvement [17].

Two recently developed methods for hierarchical clustering of large-scale datasets, namely ESPRIT-Tree and AHDC. The newly proposed method unifies the two methods and thereby effectively overcomes their respective drawbacks. A. ESPRIT-Tree ESPRIT-Tree is a HAC method that addressed the time complexity issue of conventional HAC methods through fast nearest-neighbour search using an advanced data structure defined by a PBP tree. A PBP tree is an ordered, equal-depth tree that partitions a sequence space with a series of Hyperspheres. Each node of a PBP tree represents a hypersphere in the space with a probabilistic sequence as the center. By organizing sequences into a PBP tree and performing a branchand-bound searching, the nearest neighbour of a given sequence can be efficiently found in quasi-linear time, thus avoiding the computation of the entire pairwise distance matrix. On the other hand, insertion and deletion of a sequence into or from the PBP tree takes almost constant computational time. By iteratively merging the closest pair into one cluster and inserting the new cluster back to the PBP tree, hierarchical clustering can be conducted very efficiently. ESPRIT-Tree achieved a computational complexity of (N1.2) to (N1.3) on various benchmark datasets and was identified as one of the best methods for taxonomy independent analysis in terms of clustering quality.

Despite its excellent clustering performance, ESPRIT-Tree still does not scale well to ultra-large-scale datasets, especially when the average length of sequences in a dataset is long. Moreover, ESPRIT-Tree can only be executed on one processor core due to the requirement of finding and merging the closest pairs one at a time. Thus, it is difficult to utilize parallel computing for further speedup [18].

The advantages of hierarchal clustering stated below explain why this method is chosen to compare against other clustering algorithms such as k-means. These advantages include embedded flexibility regarding a level of granularity. This means that regardless of the level of detail in a series of data, flexible cluster extraction can be achieved as existing groups can be further divided or combined which creates a hierarchal structure. Another advantage is ease of any form of distance or similarity between different data points. Lastly, hierarchal clustering algorithms are more versatile, as in they are able to adapt to different types of data structure and functions. Recognised disadvantages attributed to the above method are; although the algorithm can update the created clusters, it, however, cannot undo what has been done previously. Also based on the type of distance matrix chosen for merging different algorithms, they are often known to suffer from noise and outliers and breaking large clusters [19].

- Self-organisation map clustering algorithm

Self-organising map (SOM) algorithm is inspired by neural networks which use competition and cooperation mechanism to achieve distinct clusters. SOM involves arranging a set of nodes as a geometric pattern, normally a 2-dimensional lattice. Each of the nodes set out represents a weighted vector that has the same dimension as the input space. The ultimate goal of SOM is to successfully map the high dimensional input space onto the 2-D representation of the nodes. In regards to clustering, SOM considers data points/objects in space which is represented by the same node as grouped into a cluster. The training stage would involve presenting each data point in space to the map and hence identifying the best matching node. Normalising the input and weight vectors for input sample x(t) the best match c is identified.

SOM is generally viewed as a well-structured neural network model, considering that it has been used by many researchers to better show the different performance of this type of algorithm when applied to different applications. SOM based on Neural Networks was used for classifying images taken from space. The following research didn't require any type of pre-training (i.e. similar to unsupervised learning) and produced very good results that were carefully analysed to find endangered areas. SOM has also been utilised to improve Support Vector Machines (SVM) through exploiting a new iterative learning technique. This technique enhanced the classification process of SVM related to uncertain samples found in low density regions in the vector space.

Other instance where SOM was used is as an intelligent tool to design a Large-Scale map of carbon stock in a Forest. The result gathered from this showed that the following approach is able to re-construct unique patterns of the Forest Corg stocks with very good estimation percentage. The high accuracy required for the control process made SOM a very valuable approach. This is because SOM

provided a high-pressure sterilization control using a Self-Organizing Controller based on Neural Networks. The algorithm was used to examine the input of this controller as a one-dimensional unit that connects all the scalar valued input signals. The results produced by the following application shows how important and valuable SOM can be in specific areas.

As previously mentioned, the classification process of SOM revolves around Neural Networks which is used for competitive learning. Competitive learning basically distributes multiple scenarios of learning into clusters. The main task of this type of learning process is to find the right neurons that correspond to the input vector. There are multiple neurons that are continuously competing against each other and the neurons that form the centres of the clusters are chosen based on the vector of the synaptic weights that is closest to the vector representing the input object. SOM consists of an input and output layer. The objects of the map are scattered in space. The idea of the learning process is to first apply an approximation method which performs a random selection from the available cluster centres. The algorithm is then changed to improve the cluster centres from time to time in order to realise the clustering of the learning data. Therefore the learning process goes through a number of iterations, as for each of the learning vectors processed separately, they are subsequently presented to the network input. The desirable output vectors are defined when the neurons that are topologically similar reactions to similar input vectors [20].

The advantages of SOM include: Even though the veroni regions established on the map units are convex, the combination of several map units allow the construction of no-convex clusters to be achieved. Another advantage is various types of distance measuring tools can be utilised to form large clusters, thus making it a versatile option for clustering time series data as the type of distance measuring tool selected does indeed affect the accuracy of the objects grouped in clusters. Other advantages are it has been successfully applied for vector quantization and speech recognition which will be explained more in the later sections.

- The Expectation maximization clustering algorithm

Expectation maximization (EM) is commonly adopted by the statistics community. It is a distance-based algorithm that assumes the data set can be designed as a linear combination of multivariate normal distributions and the algorithm tries to identify the distributed parameters that would maximise the model quality measure, known as log likelihood. The main consensus of also choosing EM for comparing it against different clustering algorithms is because of its strong statistical basis, it has a linear database size, it can handle noise very well, it handles high dimensional data well and it is able to converge provided that the algorithm has been initialized well.

The EMC algorithm tries to remedy some of the issues faced by the standard algorithms such as slow convergence. The novelty of EMC is that though EMC is an unsupervised technique, it takes on a more statistically meaningful classification approach. The reason it does this is to make it easier to interpret the results associated with the correlation of the data points. If using an unsupervised clustering technique, then it is important to distinguish between cluster identification and cluster

semantics. For example, although a classical implementation of the EMC can generate statistically good clustering configurations, it will be difficult to interpret the relationship between the data points, caused by not having clear semantics. Therefore when using the EMC algorithm to achieve meaningful clustering, a set of parameters must be devised (denoted as delimiters). Delimiter reflects a value set which divides the range of a variable into a binary discretization. The results achieved for different empirical datasets shows that the EMC algorithm performs relatively well when used within various tracking technologies, species, and ecological contexts [21].

Comparison of clustering algorithms

The strength and weaknesses of the following algorithms were compared based on a range of factors such as the size of the data set, number of clusters, type of dataset and type of software. For each factor, five tests were carried out, one for each algorithm. The algorithms were executed 32 times on both a big dataset and on a small dataset. Two types of software were used to compare these algorithms including LNKnet (UNIX Environment) and TreeView (Windows Environment). Every clustering algorithm compared here apart from hierarchal clustering require k to be set in advance (as even SOM, k refers to the number of nodes in a lattice). As a result, the performance of different algorithms is tested based on the performance of k. The number of clusters chosen for k is equal to 8, 16, 32 and 64 and the lattices related to SOM are the square of them. Considering hierarchal clustering does not require to set k, the hierarchal tree is split into two different levels to achieve the same amount of clusters as the other algorithms (8, 16, 32 and 64). This subsequently results in as the value of k becomes greater; the performance of SOM algorithm becomes lower. However, in respect to k-means and EM, their performances become better than hierarchal clustering.

Number Of Clusters (K)	Quality			
	SOM	K-Means	EM	HCA
8	1001	1112	1101	1090
16	920	1089	1076	960
32	830	910	898	850
64	750	840	820	760

Table 3 – Accuracy of the algorithms

In reference to the accuracy of the algorithms (Table 1), SOM appears to show the highest level of accuracy when it comes to grouping most of the objects into their clusters compared to other algorithms. However, as the number of k clusters increases, the accuracy of the hierarchal algorithm becomes better. It is found that in terms of accuracy, k means and EM appear to be least accurate when tested on all small size and large size k clusters [22].

However, when the following 4 algorithms were tested on a larger dataset, K-means and EM performed better. This is since the previous test carried out was on a small sized dataset consisting of 200 rows and 20 columns, whereas when these algorithms were tested on a larger dataset consisting of 600 rows and 60 columns the quality of the clusters grouped were more accurate as evident in (Table

2). In conclusion, partitioning algorithms such as k-means and EM are mainly used for huge dataset while hierarchal algorithms are used for smaller datasets. The experiments conducted by journal also stated all the algorithms tested had some ambiguity in some noisy data when clustered which could have been a factor affecting the reliability of the results outputted. This was especially the case for clustering using the k-means algorithm as it was found that the k-means and EM algorithm was very sensitive to noise. The reason noise has been emphasised here is because noise makes it difficult for the algorithm to cluster an object into its correct cluster in terms of sharing the highest level of similarity with the other objects in the group.

K=32				
Data Size	SOM	K-Means	EM	HCA
36000	830	910	898	850
4000	89	95	93	91

Table 4: The effect of data size on algorithms

Overall, the k-means algorithm performed better than the other algorithms in most areas and even in the areas where it was weaker, there seems to be room for improvement. Therefore the Meta-clustering approach proposed in this project will try to improve the k-means algorithm to perform better when handling small size data set considering that k-means struggled compared to the other algorithms when the dataset size was small. The project will also increase the k-means performance by ensuring the algorithm produces the clusters at a faster rate by clustering the time series data in two separate stages which would involve first creating the inner clusters and then generalising those inner clusters into a single optimised general cluster.

2-2.2 Reviewing common clustering algorithms tested on Time-Series data

Since the establishment of many of these clustering techniques, their overall architecture has not changed much. Any improvements made were mostly concerned with either the similarity/distance measure used to assess dissimilarity between objects, or related to the function used to calculate prototypes. This is also the case for time-series clustering, whereby Dynamic Time Warping (DTW) distance has become a typical distance measure choice in that context. However, DTW is computationally expensive, hence leading to it being researched quite extensively over the years for optimisation. Clustering algorithms tend to perform differently depending on the dataset used. Therefore different techniques need to be tested and further compared to find a common infrastructure that is advantageous. The following study provides an overview of shape based time-series clustering. This also includes many specific attributes related to DTW as well as other modern techniques introduced. The dtwclust package used in R statistical software platform will also be briefly described which will show how it can be utilised to evaluate many different time-series clustering techniques.

Introduction

Clustering is the task of creating a group of patterns. The ultimate objective is to try to ensure that all patterns in the same cluster are similar to each other as much as possible, while at the same time dissimilar as much as possible from patterns in different clusters. Clustering cannot be outlined as a single. In addition, the characteristics related to the patterns to be clustered differ. Therefore a multitude of algorithms can be used to cluster patterns as each one defines a cluster, measures the similarity of the patterns in a cluster and establishes its own approach on how to find groups efficiently. It is also important to consider that every application will have its own set of goals, hence explaining why a particular clustering algorithm may be preferred as it depends on the type of clusters sought.

How the Clustering algorithms are organised can vary as it depends on how the data is handled and how the clusters are created. In the case where the user is dealing with static data (implying the values do not change over time), five main clustering techniques have to be mentioned. These include partitional, hierarchical, density-based, grid-based and model-based clustering methods. The following algorithms mentioned can be considered as an intermediate step as well as a pre-processing step. Time-series is a very common type of data applied in many tests and simulations. It is a type of dynamic data that can be seen in a wide range of scenarios. Some examples of these are medical data, stock data, and machine monitoring. Many of these scenarios that handle time series data are faced with challenging problems. This is mainly because of the large size of these data and the high dimensionality commonly related to time-series. As a result of this, the dimensionality of a series is related to time, whereby time can be recognised as the length of the series.

In addition, another issue that should be considered is even a single time-series object can consist of a large number of values that change on the same timescale. This is what refers to as multivariate time-series. This is why many dimensionality reduction techniques have been proposed which modify the time-series data. These techniques mainly handle the way time-series are represented. On that note, modifying representation can be considered an important task, as it is not only practised for time series clustering but also researched in other areas as well. As discussed earlier, the choice of representation can have a significant effect on the clustering results, but since it is often considered as a different step, it will not be further expanded in this section.

It is apparent now that time-series clustering is aimed at of clustering algorithm developed to manage and handle dynamic data. One of the most important components to take into account with time series clustering is the distance measure, the clustering algorithm, the cluster evaluation method and the prototype extraction prototype. In most scenarios, the algorithms developed to handle times series clustering adopt static clustering algorithms. And while doing this, they either adjust and change the distance measure or the prototype extraction function to something relevant to their application, or conduct a transformation to the series in order to obtain static features. Thus the fundamental basis for a variety of these clustering procedures stays approximately the same across all clustering techniques. The most popular approaches used to alternate between partitional, hierarchical, and fuzzy clustering. A research carried out by Aghabozorgi et al. (2015) groups time-series clustering algorithms by looking at how the data is handled and how the underlying grouping is achieved. As the study shows, the results achieved from one of the classifications performed heavily depended on how the time series was to be clustered i.e., clustering the whole series, a subsequence, or as individual time points. This is considering the clustering itself could be feature-based, shape-based or model-based. A distinction also made by the following study between online and offline approaches show online approaches mostly handle grouping incoming data streams seamlessly, while offline approaches handle data that will not go through any more changes. When dealing with shape-based time-series clustering, Dynamic Time Warping (DTW) is an effective distance measuring function to utilise. The distance of DTW is basically calculated by implementing a dynamic programming algorithm that searches for the best warping path between two series under certain restrictions. Nevertheless, the disadvantage of the DTW algorithm is it's computationally expensive to implement in terms of memory and time usage. There have been many alternative solutions proposed to address this issue to either accelerate or optimise the calculation of DTW. These techniques will be discussed later on in this section.

The nature of clustering procedures opens itself for parallelisation. This is because a number of similar calculations can be carried out that are entirely independent of one another. This could potentially have a considerable impact on the clustering results, especially if the complexity of the data stream increases over time or highly computationally expensive algorithms are used which consume a lot of memory and time. Selecting the right time-series representation, preprocessing, and clustering algorithm will profoundly affect the performance of a clustering approach in terms of the quality of the clusters created and the time taken to form those clusters.

The other factor which could also impact the clustering technique refers to the different programming languages used as they may have different user interfaces and run-time characteristics. Furthermore, although there are many authors that make their algorithms publicly available, combining them may lead to significant results. It is, therefore, best to stick to a comfortable platform prefer in order to perform more reliable tests and comparison on different clustering algorithms. R statistical software provides the dtwclust package which provides such functionality and has many implementations of recently developed time-series clustering algorithms and optimisations. It can bridge the gap between classical clustering algorithms and time-series data. It also provides visual routines and evaluation frameworks that are able to handle time-series effectively. The algorithms provided in R software are all custom implementations, apart from the hierarchical clustering techniques. From the tests and experiments carried out on many of the algorithms, it has been recognised as an excellent package whereby a vast amount of effort has gone into it, and the algorithms itself have been implemented efficiently, and the functions have been designed are flexible and can be extended easily.

The majority of these algorithms and optimisations are mostly designed to effectively handle DTW distance, hence explaining the name of the package (dtwclust). That being said, the primary clustering

procedure remains to be flexible, thus allowing it to be tested on a wide range of clustering techniques. Therefore, all the algorithms included in the dtwclust package will be described in-depth, in which highly relevant characteristics of each algorithm and the procedures the package uses to evaluate these algorithms will be pointed. In addition, the research also looks into the various forms of DTW and other popular similarity distance measures will be explored.

In addition to dtwclust, other R packages also exist for clustering different types of data. For example, the flexclust package is used to implement partitional clustering solutions and cluster package which looks at implementing hierarchical solutions. However, neither of the mentioned packages are specifically meant to handle time-series data. On the other hand, TSdist and TSclust are packages that can better deal with applying distance measures for time series. Another package used refers to the pdc package where the clustering algorithm is based on processing permutation distributions. Another package associated to DTW is the dtw package which is used for implementing a more advanced functionality. However, the dtw package does not cover the lower bound techniques that can be particularly useful in time-series clustering. There have been new clustering algorithms proposed such as k-Shape and TADPole which were implemented in MATLAB and can be accessed upon request. This explains why the dtwclust package can be seen as a reliable and valid package for interacting with classic and new clustering algorithms, considering how easily the smallest changes in time-series data can impact the clustering outcome.

Distance measuring techniques with respect to dealing with time series data will be discussed in depth and algorithms that support prototype extraction will also be covered later on. Then the report will cover some of the main clustering algorithms, and the clustering algorithms will be further evaluated as well. A comparison of the results achieved from the analysed clustering algorithms will be presented and discussed. The following research undertaken intentionally uses brief examples, hence does not represent an in-depth procedure to choose or evaluate a clustering algorithm.

Distance measuring

A distance measure enables the system to make a distinction between two time-series. One of the most important functionalities of a time-series clustering algorithm is being able to calculate distance, as well as cross-distance matrices between a set of time-series patterns. This is an iterative task and is applied to all data patterns when trying to find a suitable distance function that can measure the distance between two patterns as accurate as possible. A highly optimised framework 'proxy package' is extensively used by dtwclust to improve these calculations. The framework provides a large number of standard metrics where custom functions can be registered to its database. The Euclidean and Manhattan distances also referred to as $l1$ and $l2$ vector norms are the most commonly used distance measures respectively. It can be argued they are also the most competitive lp norms when measuring the distance between two patterns.

The only way the two mentioned distance measures can efficiently compute the distance between two patterns is for the series experimented to be of equal length. They are also very sensitive to scaling and noise issues as well as time changes. In order to better address these issues and overcome some of these limitations, there have been other distance measures introduced that are customised to handle time-series type of data better. These distance measures account for multiple variables, serial correlation, etc. The distance functions included in dtwclust, which is associated with shape-based time-series clustering, and supports DTW or a modified version of it will be described and analysed in

the following section. It is important to note that there is a wide range of other distance measures that can be used for time-series, but the focus will be on the most common distance measures such as Euclidean, Manhattan, and DTW. The following time-series will be denoted as a vector or a set of vectors (in cases where there are multivariate series) x. For a given time series, each vector will have the same length. Generally speaking, x v/i depicts the i-th element of the v-th variable when dealing with a multivariate time-series x. In order to avoid the time index explicitly, all the elements will be assumed spaced out equally in time.

Dynamic Time Warping Distance

To shed more light on Dynamic time warping distance, DTW is a dynamic programming algorithm that compares two series. By doing so, it tries to find the best warping path between the two series under some restrictions. DTW started to be commonly used among researchers in the data mining community in order to address some of the shortcomings related to Euclidean distance. DTW tries to align two time series samples together. With this in mind, the first and last points of the series need to match, while other points in the series are repeatedly warped in time so that more accurate matches can be found. However, DTW is computationally expensive to implement as compared to other distance measures such Euclidean and Manhattan. To describe this with an example, if x has length a and y has length b, the distance between the two can be computed in O(nm) time, which makes this quadratic if a and b are similar. Also, DTW is known for being difficult to implement. This is because the calculations made cannot easily be vectorised and also the calculations are extremely slow when using programming languages that have not been compiled. In respect to the dtw package, this includes uses a C programming language to implement the dynamic programming segment of the algorithm. This makes the distance measure faster, even though its performance may suffer due to its generality. Another option is to use a customised version of the DTW algorithm implemented which is also included with dtwclust in the dtw_basic function. Although this has less functionality, the main functionalities, however, are supported, and the C core it has allows one to optimise the memory to make the algorithm compute faster. This is particularly useful when dealing with calculating the DTW distance between several pairs of time-series. DTW distance can potentially handle time series data that have different lengths. However, this not considered a positive point, because based on many experiments conducted, it has been explored that performing linear reinterpolation to achieve equal length between m and n (assuming they do not vary significantly) may be an appropriate solution. With that said, there is some crucial point that must be raised. Firstly, in order to successfully implement DTW, a local cost matrix (LCM) of n × m dimensions must be created. This type of matrix must be created for every pair of distances compared. This results in memory requirements growing fast since the dataset size is expanding. Let's consider an input series of x and y, for each element (i, j) of the LCM, the lp norm between xi and yj must be computed. If the multivariate series have the same number of variables, then it is supported. It, therefore, makes sense to discuss DTWp distance, as p refers to the lp norm used to build the LCM. Nevertheless, the following norm has an essential task in the next step of implementing DTW.

In the next step, the path that reduces the alignment between x and y as much as possible is iteratively conducted by the DTW algorithm. This iterative process steps through the LCM by beginning at lcm (1, 1) and ending at lcm(n, m). A proxy is also used by the dtw package to calculate the LCM. This allows a dist.method argument to be used by the proxy allowing it to change the norm. It is important to note that only when multivariate series are used can the norm affect the LCM. The dtw package considers the way lp norm based on a special version of DTW2. The package also makes use of dtw for the most important calculations, but also during the second step uses the l_2 norm. How the

algorithm manages to traverse through LCM is mainly dependent on the step of the pattern selected. A local constraint assigned by the step pattern determines which directions are allowed to be taken when choosing to progress with LCM while the cost is being aggregated, and the corresponding per-step weights. The majority of articles published in the data mining community never specify precisely the pattern they use. However, generally speaking, the symmetric1 pattern considered to be standard. In contrary to this, the dtw function uses the symmetric2 pattern by default. With that said, it can be easily modified by enclosing the appropriate value in the step.pattern argument. Also, the chosen step pattern also determines whether the step pattern linked to the DTW distance can be normalised or not. This is important for situations where the series has different lengths. To summarise the key points of DTW, triangle inequality cannot be satisfied by DTW distance, and generally speaking, it is not symmetrical.

DTW Constraints

DTW can be modified through the use of global constraints. This constraint is also known as window constraints and what this does is it reduces the area of the LCM that can be reached by the clustering algorithm. Many Different types of windows exist, but one of the windows used commonly by researchers refer to the Sakoe-Chiba window. The Sakoe-Chiba window allows a boundary to be constructed diagonally across the LCM. What such a constraint does is it slightly makes the DTW calculation faster, regardless of the primary purpose it is used for which is to avoid pathological warping. In many situations, a window with a size of 10% of the series' length is often used. This is considering that there have been cases where smaller windows have led to a better outcome. An important note that must be considered is if the length of the series compared has different lengths, it is possible that such a constrained path would not exist. This is because the band related to the Sakoe-Chiba may not allow the finishing point of the LCM to be reached. For such an issue, a slanted band window is the better option, as it can constrain a series of different lengths just like the Sakoe-Chiba window. Inside the dtwclust, if such a window constraint utilised, a slanted band will be employed. In advance, it is difficult determining what window size will be best to be applied to a specific application. However, in most circumstances, it is often agreed that choosing a constraint is the correct decision. This explains why it is advised to conduct tests and experiments with the data in which it best works with, ideally testing it on a subset of the data to avoid long running times.

Time series prototypes

A key essential step involved in clustering time series data refers to calculating time series prototypes. The primary objective of a cluster is to ensure that all series within a cluster are similar to each other and have similar attributes. This does depend on what the user seeks to base the similarity on, for example, researchers in data mining may try to define a time-series that can effectively summarise the most significant characteristics of all series in a given cluster. The following series which is based on characteristic can also be referred to as an average series. Prototyping itself can also be referred to as time series averaging. These prototypes are capable of performing visualisation aid as well as classifying the time-series data. However, when it comes to clustering, partitional clustering techniques heavily depend on prototyping, considering the results outputted by the prototypes are subsequently used as the centre points of a cluster. The type of prototype function chosen is associated with the distance measure chosen and, similar to distance measures, it is not easy to know which type of prototype will produce the best results. A wide range of procedures exists for prototyping time-series data. However because of their high dimensionality, it is difficult to work out what exactly can be considered as an average time-series, and such factors could significantly slow down the performance of a clustering algorithm. The most popular prototype approaches implemented by in

default in dtwclust will be described. The possibility of creating modified prototype functions and provide them in the centroid argument will also be described.

Partition around medoids

Partition around medoids (PAM) is a highly used approach which is also available in the dtwclust package. A medoid represents an object from a cluster, so in this aspect, it can also be a time-series, in which its average distance compared to the other objects in the same cluster is very small. Now considering that a medoid object is always considered as an element comprising of the original data, rather than using mean or median to calculate the average, PAM is often chosen, since it prevents the time-series structure from changing. The other advantage of PAM as compared to mean or median is, considering the data does not change, the whole distance matrix can be computed beforehand and repeatedly used during each iteration. Furthermore, random repetitions can be made across various clusters. Although it is not necessary to precompute, it does most often save a decent amount of time overall; hence it is a default functionality inside the dtwclust. This is not however recommended for dealing with much more extensive datasets. This is because the entire distance matrix has to be allocated at once, meaning it is also possible for the precomputation to be deactivated. As part of the dtwclust package, k series from the time series data are randomly selected as initial centroids.

Once the centroids are determined, the distance between the centroids and all the series around it are thus calculated or retrieved depending on whether the whole distance matrix was computed in advance, whereby each of these series is assigned to the appropriate cluster showing the highest level of similarity to the centroid. The distance between all members of the series is calculated for each cluster created, and if any series has a minimum sum of distances, then they are selected as the new centroids. The following process is continuously repeated to the point where the series no long change clusters or it has gone beyond the maximum number of iterations permitted.

DTW barycenter averaging

DTW distance is commonly used for handling time-series data, and as because of this, a prototyping function based on DTW has also been developed. The following procedure is known as DTW barycenter averaging (DBA). This is an iterative, global method which means that the outcome remains the same regardless of the order in which the series enter the prototyping function. The following procedure does not have to be accessed through the dtwclust, as it is available as a separate function which is named DBA. A series must be used as a reference, i.e., centroid. This starts typically by randomly choosing one of the series in the data. Now during each iteration, the DTW alignment between the centroid and each series in the cluster C are processed. As a result of the warping performance in DTW, this can mean several time-points inputted from a given time-series are required to map it to a single time-point in the centroid series. Therefore, for each time-point in the centroid, all the corresponding values from all series in C are organised as one based on the DTW alignments, and furthermore, the mean for each centroid point is computed using the values inside each of their respective group. The following test is iteratively repeated until it has reached many iterations, or until the points are considered to have converged. This is implemented using C++ in dtwclust which also comes with a number of memory optimisations. It is however computationally expensive to implement because of the many calculations that must be performed by DTW. Nevertheless, it is beneficial when using DTW distance and, because of DTW, different length series can be directly supported by it. A limitation of this is that the length of the resulting prototype will be equal to the length of the reference series that was initially chosen by the algorithm.

Time series clustering algorithms

- Hierarchical clustering

The Hierarchical clustering method as indicated by its name is an algorithm that groups a series of objects as a hierarchy whereby, the clusters are formed one by one as the level in the hierarchy increases. The following clusters are created when the clusters from the next lower level are merged to the lower level, which as a result obtains a sequence of grouping in order of relevance. The merging between the different levels is performed by determining the distance measure to use for grouping the objects together. Also, note a separate distance measure must also be used to calculate pairwise similarities. Unlike other clustering algorithms, the hierarchal clustering algorithm does not require the user to initialise a specific number of clusters. The process of the algorithm is determined in advance, meaning regardless of the similarity measure adopted, the same results will be the same. The algorithms proposed for implementing hierarchical clustering can either be agglomerative or divisive. Agglomerative algorithms have proven to be more widely used by the data mining community. Agglomerative procedures are when every group member derived from a data must start in its own cluster, and then the members are mapped together one by one based on the similarity measure. This is repeated until all members are grouped inside a single cluster. Contrary to divisive procedures, this does the complete opposite, as divisive procedures begin with including all the data in a single cluster and then sequentially spreading them out to point where each member has become a single entity. The two strategies have a constraint such as lack of flexibility since they are not able to change once a split or merger has been done.

The dissimilarity between the intergroup is also known as linkage. For example, a single linkage considers the intergroup dissimilarity to be an object which has the shortest distance to the closest pair, i.e., least similar pair. A large number of linkage methods exist, but it does not make much of the difference in the sense that if it is easy to group the data, then they should all provide similar results. In regards to the dtwclust package, the native R function hclust is used as a default function which supports every linkage methods. That being said, there are also other clustering functions that can be used, and these clustering functions are open to some limitations. A binary tree can be used as a visualisation to depict the hierarchy created in which the height of each node is proportional to the value of the intergroup dissimilarity between its two daughter nodes. The following plot is known as a dendrogram. Plots provide a convenient way to summarise the entire data in a way that can be easily interpreted. However, this can be seen as misleading, considering the algorithms are bound to a hierarchical structure.

The dendrogram does not directly indicate a certain amount of clusters, but one can be sought after. A way to possibly achieve this is to evaluate the dendrogram visually. By doing so, the dendrogram will be able to examine the height at which the most significant change in dissimilarity occurs. This consequently causes the dendrogram be cut at said height and the clusters that are created to be extracting. An alternative option is to predefine the number of clusters that are wanted, and once the initial clusters have been specified, the dendrogram is cut in such a way that the selected number of clusters is obtained. The second option allows several cuts to be made, and furthermore, validity indices can be utilised to determine which value leads to the algorithm producing the best results.

After obtaining the clusters, it could be useful to make use of them to create prototypes. The type of prototyping function applied is not essential at this stage. However, it must be emphasised that

choosing a prototype function is an optional step in hierarchical clustering. A negative aspect of hierarchical clustering is that for any given dataset, it is a requirement to calculate the distance of the whole matrix. In most situations, this contains the time and memory complexity of O(N2) if N is the total number of objects in the dataset. Therefore, hierarchical clustering techniques perform best when they are tested on datasets that are not too large.

- Partitional clustering

A different form of clustering used to create partitions is partitional clustering. This type of clustering involves explicitly assigning the data to only one cluster out of the k total number of clusters that exist. The desired number of clustered are pre-defined. Despite the fact that this procedure appears to be a limiting factor, clusters can, however, be amended by using validity indices. Applying Partitional strategies can be expressed as a combination of optimisation issues. These issues reduce the intra-cluster distance while increasing the inter-cluster distance. In order to find the most optimal solution, all possible groupings must be accounted for, which it would require a lot of resources and time even for small datasets. This is why more confined strategies are used instead, which mainly observe a small segment of the search space until all points have converged. How partitional clustering algorithms function is as follow: The first step involves randomly initialising the k centroids, which involves randomly selecting k objects from a given dataset. These centroids are assigned to individual clusters. At this point, the distance is calculated between the centroids initialised and the rest of the objects in the data, whereby, each object is assigned to the closest centroid. The next centroids are updated accordingly by applying a prototyping function. This process iteratively updates the centroids and distances to the point where the object in the clusters does not change anymore. Also, after a certain number of iterations, the cluster might become empty, which therefore means a new cluster can be once again initialised by randomly choosing a new object from the data. This is included in the dtwclust which tries to maintain the right number of clusters, but can in some cases become unstable. Possible changes that may partially address this problem could be trying out a different distance function or lowering the value of k.

One of the optimization capabilities included in the dtwclust package that optimises the centroid functions is that, for each iteration, a centroid function is only applied in cases when it actually causes the cluster to change either by losing or gaining data objects. The optimisation is also able to keep track of the data indices; the changes they go through as well computing which cluster centroids are selected, hence avoiding the data being modified. Commonly used partitional algorithms refer to k-means and k-medoids. Both of these algorithms mainly use the Euclidean distance to group the objects in the correct clusters, respectively. The following algorithms proposed for time-series clustering utilise the same procedure, while mostly altering the distance and/or centroid function. Partitional clustering techniques are recognised to be stochastic because of how the cluster centroids are randomly initialised at the beginning. Therefore, it is advised to spend a lot of time at the start of the process, testing different centroids. On this basis, several evaluate several optimal solutions can be evaluated, helping you select the best result. This type of clustering is suitable for dealing with much larger dataset as it produces rounded clusters.

- TADPole clustering

Another type of partitional clustering refers to TADPole clustering, and it is also included in the dtwclust as the TADPole function. The clustering framework it uses is new and the framework is capable of applying for clustering time series data while also being able to handle DTW distance. The functionality of the algorithm can be viewed as a kind of PAM clustering. The reason for this is since

the centroids are always considered as elements of the data. In the face of this, the algorithm outcome depends on the cutoff distance (dc) value. The algorithm functions by first using the DTW distances associated to the upper and lower bounds. The distance measures used to find the series with many close neighbours (inside the DTW space) refer to Euclidean distance and LB_Keogh. Any result that is below the dc value is considered as the neighbour. By having this information, the algorithm tries to speed up the clustering procedure by pruning as many DTW calculations as possible. Any series within space that is located in very dense areas (i.e., surrounded by lots of neighbours) are chosen as cluster centroids. It is important to note, TADPole heavily relies on the DTW bounds, and the following bounds can also be defined for time-series when their lengths are equal to each other. In addition, Euclidean distance is only valid as a DTW upper bound if the symmetric1 step pattern is used. Lastly, a high number of distance matrices need to be assigned, consequently meaning just like hierarchical algorithms, it requires a lot of memory-wise, hence limiting its applicability and by doing so making the algorithm mainly suitable for relatively smaller datasets.

- k-shape

K-Shape clustering is another type of partitional clustering algorithm whereby both the distance measure and centroid function are custom made. It is also relatively random in nature, and hence requires z -normalization to be its default definition. The distance and centroid functions are included in the dtwclust as individual functions. The reason it is implemented individually is to allow the user to independently combine and use with other algorithms. This clustering algorithm requires the main clustering function (tsclust) to be called with SBD as the distance measure. Shape extraction reflects the centroid function, and the pre-processing step is comprised of the z -normalization.

- Fuzzy

Fuzzy clustering can also be used for partitional clustering. Other clustering methods are known to result in hard partitions, although this is true with hierarchical clustering only when the dendrogram is cut. Hard partitions basically mean when each member of the data belongs to only one cluster, and the clusters formed are independent of each other. Contrary to this, fuzzy clustering operates by creating a soft partition in which each member belongs to each cluster to a certain degree. The degree which each member of the data belongs to is constrained, resulting in its sum to equal 1 across all clusters.

Parallel computation

Parallelisation in data mining has never been explored a lot in literature, with that said it could be advantageous to apply in practical applications. One of the primary uses of parallelisation in reference to time-series clustering is that parallel computation can lead to the execution times of the algorithm to be immensely reduced. Key points to take into account when using parallelisation is to, first of all, make a distinction between multi-process and multi-threaded parallelisation. The ideal amount of parallel workers (subprocesses) whereby each one of these subprocesses can handle a given task, depends on the processing power of the computer being used as well as the number of physical processing cores and logical threads it can handle. Each worker may require more RAM since R is only able to handle data that is loaded onto the RAM from before. Lastly, the overhead which is introduced for orchestrating the parallelisation may prove to be too big when compared to the amount of time required to complete each task. This is definitely true for in cases where the user is dealing with workloads that are considered basic. So this means that using parallelisation does not necessarily lead to faster execution times, and as a result, the tests carried out must be tailored to the specific

application. The different software applications in R has hugely simplified the way in which parallelisation is handled.

How it is implemented in dtwclust for example, is it is able to multi-process by using the 'foreach' package, and it can multi-thread by using the RcppParallel. Both multi-process and multi-threading are used during the clustering procedure. Before introducing all the different situations in which dtwclust can immensely benefit from multi-processing, a significant point that must be considered is by default, only one level of parallelisation can be implemented in that case. What this implies is, every task which is performed by a given subprocess is done in a specific order, and further, they are not able to benefit from additional parallelisation even if more workers are still available. In regards to clustering objects based on partitional clustering, a high number of iterations are expected to be carried out. These will thereby have different random seeds to account for different starting points. In many of these iterations are directly specified to tsclust, a different worker will be assigned each time the process has been iterated. This also relates to when the function has to handle several values of k, which happen when there is more than one cluster that needs to be tested. The following scenario is commonly found in partitional, TADPole, and fuzzy clustering. The way the TADPole clustering algorithm is implemented is it is able to support multi-processing for different values associated with the cutoff distance. In reference to partitional clustering, the time-series prototypes are calculated a number of times, during each iteration. Now considering these clusters are mutually independent of each other, the calculations made for the prototype can be performed in parallel. This however only should be conducted if the calculations take long to compute. For this reason, multi-processing, when used in partitional clustering, is only executed by the dtwclust for DBA, shape extraction, and the soft-DTW centroid

Cluster Evaluation

Clustering algorithms are generally considered to be an unsupervised technique. This means depending on how you try to evaluate its performance, the results achieved can be quite subjective. The following issue has led researchers to spend a huge amount of time trying to create a standard cluster evaluation metrics through the use of cluster validity indices (CVIs). There have been many indices introduced in recent years, and the following indices are categorised into their very own research area, but there are some overall specific points that should be mentioned about indices. Generally speaking, the following CVIs can be tailored to be implemented as either crisp or fuzzy partitioning. In regards to crisp partitioning, CVIs can be classified as internal, external or relative, which depends on how it is computed. In regards to the first two, the key difference between them is that internal CVIs only consider the data that has already been partitioned and based on this principle, it tries to propose an evaluation criterion that can measure the purity of the cluster formed. In the case of external CVIs, they try to basically compare the partition achieved with the correct one which means external CVIs is only valuable when the ground truth is has been identified. It is important to understand that there are fuzzy CVIs tailored explicitly for fuzzy clustering, in addition to the fact that it can also be modified into a crisp CVI, hence making it compatible with many of the existing "crisp" CVIs available. Fuzzy partitions that are formed do not have much knowledge associated with them, but in some cases, this can be false as it mainly depends on the primary task that needs to be achieved. Each of the following indexes defines how much values it can hold and whether they are to be maximised or minimised.

The point here is that many of these CVIs can be readily applied to the clustering algorithm in order to evaluate how well they perform. The reason it is easy to evaluate is that the CVI does not need to know how the clustering algorithms internal functions, or how the groups created were formed. An

example of an internal CVI used is the Silhouette index, and an example of an external CVI used is the Variation of Information. The dtwclust package includes a wide range of CVIs that have excellent performance. These CVIs are under the cvi function, but it is also possible for the user to implement their own indices. Nevertheless, the original internal and fuzzy CVIs defined, the Euclidean distance and a mean centroid were used. This is changed when implemented in dtwclust, as instead utilises distance/centroid measuring function based on the distance measuring function used during clustering. It still, however, must be noted that many CVIs assume symmetric distance functions. It is difficult to determine which CVI will best evaluate the clustering algorithm in prior. Therefore it is suggested that all the CVIs be tested for each specific application. In most case, the different CVIs are tested, and their results are compared to each other. Furthermore, many of these CVIs can perform additional distance and centroid calculations when being computed, which can be very suitable for DTW. To put this into perspective, in regards to the Silhouette index effectively, the whole distance matrix needs to be between the original series to be calculated, hence limiting its performance in some instances. There are also other evaluation frameworks available effective for assessing the clustering results other than CVIs. This refers to the clue package which can also evaluate the clusters formed. This evaluation framework does not directly deal with the clustering algorithms; it instead tries to find ways of quantifying the agreement and consensus between the results obtained from the clustering algorithm. This allows it to be easily compatible with the results obtained from the dtwclust, considering that it does not matter how the partition or hierarchy was created.

Summary

A general understanding of the most common shape-based time-series clustering methods was discussed. DTW distance was mainly emphasised whereby information was provided regarding the functionalities of DTW as well as specifying the optimisations available for DTW such as constraints and lower bounding techniques. While going through this, the dtwclust package within R was also thoroughly described, and several examples were shown. This package provided a complete infrastructure, demonstrating how it can be used to carry out various tests and effectively compare different clustering algorithms together in an unbiased way. The dtwclust was able to implement a wide range of different procedures, which were mainly associated with the DTW algorithm. With that said, the way in which it is structured makes it possible for the user to change and improve the functionality as part of the package, for example by customising algorithms or even other packages included in, as shown with TSdist, cluster, and clue. These packages are tailored to deal with specific tasks such as distance calculations, hierarchical clustering, and cluster evaluation. Furthermore, due to their specialised nature, it also makes it more difficult to extend. In contrary to this, the dtwclust provides a much more generalised clustering infrastructure. The flexibility associated with the dtwclust package allows the most common approaches to be used. Although the following algorithms and packages were not explained in-depth, a clear overview was provided relating to the most recent research done in the literature thus far, which as a result provides a more precise picture to the reader and allows the reader to see what clustering algorithm would provide the best outcome for their specific application. This is considering that it is a difficult task when trying to choose a specific clustering algorithm for a given application. This is since there are a number of essential factors keep in mind and it is also difficult to know in advance which clustering and distance measure was chosen would lead to valid results. This study implemented the algorithms using heavily optimised R functions. They relied on compiled code where needed. This provided a good understanding of time-series clustering using the dtwclust and will help the research project when trying to choose an optimal clustering algorithm that would provide effective results when tested on the given application [23].

2-2.3 Compressive sampling

Vector Quantization approaches

Now moving onto an effective compression/classification methodologies, vector quantization techniques provide a useful solution for grouping and coding data gathered from various sensors. Being able to develop robust image processing algorithms is very important for filtering and extracting meaningful features or patterns from images [24]. One of VQ's attributes is that it provides a method to compress large sized images by which it can be stored and transmitted in very efficient manner. The process of compressing images at this stage involves decoding the data inside the images at the receiving end and then reconstructing that image. At this stage, fidelity and meaning of the original image must not be lost but preserved.
Vector Quantization is being used in various scientific fields for processing data. It basically operates through three steps, codebook design, encoding and decoding processes. For example, when applying VQ to an image, a codebook is first generated using a trained dataset. The codebook consists of segmenting an image into a set of image blocks which are non-overlapping and thus producing image vectors that are stored within the codebooks [25]. So each codebook vector produced is then compared to each image vector, and accurate image vector-codebook vector is combined. This combination helps to find when distortion is at a minimum. During the process of transmission, a series of index numbers are attached to the codebook or the codeword, where each index number represents the index of the codebook vector that distinctly represents an image vector during which the image has been reconstructed at the receiving end i.e. compressed. Thereby the data processed by the original image is very much sorted into a concise codebook.
VQ codebook optimization algorithm used for Speech Recognition

Another type of optimization algorithm used for speech recognition illustrates the importance of optimizing our methods and techniques. Usually, the performance of the vector quantization process is heavily dependent on the initial codebook chosen. The 'Immune Cat Swarm Optimization' (ICSO) presented in this paper [26] helps to resolve this issue and by doing so, reduces the error rate of speech recognition. Vector Quantization used for this research reduces the dimensions of the characteristic in speech recognition. The main emphasis here in regards to VQ is designing an optimal codebook. The main reason for proposing an improved particle swarm optimization VQ algorithm is because of the slow convergence rate in codebook clustering.
The results achieved from this paper [27] showed that the performance of the codebook was not necessarily dependent on the initial codebook selected as a result of using the ICSO algorithms. The results gathered from the simulations conducted did show that when applying the ICSO algorithm, an optimal codebook was designed which effectively solved the problem of relying on the initial codebook.

2-2.4 Similarity measures for pattern recognition

Different types of similarity measures are discussed in this section. The main task of similarity measures is to measure the distance between two time series T1 and T2, denoted by Dist (T1, T2). The distance measures will be divided into two groups. Distance measures that compare the i−th point of one time series to the i−th point of another are referred to as lock-step measures (e.g., Euclidean distance and the other Lp norms) and distance measures that allow comparison of one-to-many points (e.g., DTW) and one-to-many/one-to-none points (e.g., LCSS) are known as elastic measures.
One of the most common and straightforward similarity measures is the Euclidean Distance and its variants. Apart from being straightforward, the Euclidean distance and its variants are also easy to implement and can be indexed using any access method; for instance Spatial access method. And the interesting point is that even though the Euclidean distance though is pretty easy to implement, it's surprisingly very competitive with other more complex distance measure approaches, particularly if the size of the training data set is large. However since Euclidean distance measures and its variants are fixed when mapping the points between two time series, they are very sensitive to noise and

misalignments in time and most importantly cannot handle local time shifting. Local time shifting refers to instances where similar segments of two points in space are out of phase [28].

A distance measure that tries to handle time warping in similarity computation refers to Dynamic Time Warping (DTW) introduced by Berndt and Clifford [29]. This was originally a classic speech recognition tool within the data mining community. DTW is used for many applications as it allows a time series to be "stretched" or "compressed" to provide a better match with another time series. There are also a number of lower bounding measures added to DTW in order to speed up the overall similarity search, and also it has been exploited that the cost for computing DTW on large data sets is relatively low. The warping feature of DTW can also be used for developing the lower-bounding distance and for indexing time series [30]. The other types of similarity measures for time series are developed based on the notion of the edit distance for strings. A popular distance measure based on this is the LCSS distance. LCSS matches time series characters and has a threshold parameter, which basically states that two points from two time series match if their distance is less than E. A similar type of distance measure also based on edit distance is EDR [31]. EDR also uses a threshold parameter, except the only difference is that its role is to quantify the distance between a pair of points to 0 or 1. As opposed to LCSS, when EDR finds a match between two time series it assigns penalties to the gaps between two matched segments according to the lengths of the gaps. In summary, what is important to consider when choosing a distance measure for your search engine is to test a number of distance measures and see which is best able to compute the similarity between two time series points efficiently as different data sets react differently to the different distance measure. Euclidean distance measure seemed to be the most suitable measuring tool when using an optimised k-means solution to measure the similarity between different points. The reason Euclidean measure will be used in this project is that while using k-means clustering, Euclidean is able to compensate for instances where the points in a metric space are in opposite directions but still may fall into the same cluster, as the distance of both points from the centroid is the same [32].

2-2.5 Indexing Scheme for Processing Time Series Data

Employing an indexing scheme helps us to organise the time series data efficiently. By doing so objects of interest can be retrieved quickly among large databases. The indexing schemes often proposed usually involve some kind of dimensionality reduction method. This allows the data represented to be indexed using a spatial access method. Most studies indicate that different representations suggested mainly differ in terms of indexing power, the quality of results and speed of querying [33]. In the process of developing an indexing algorithm two major issues must be considered. The first issue refers to the completeness of the indexing scheme i.e. no false dismissals and secondly the soundness of the indexing scheme i.e. no false alarms whilst querying.
For a time series T being an ordered sequence of n real-valued variables
$T = (t_1; : : : ; t_n)$; t_i 2 R (2)

Some of the important properties that an indexing scheme must encompass are the following:
(1) It needs to be much faster than sequential scanning.
(2) The technique should require less space.
(3) It also needs to handle various sized queries.
(4) The indexing scheme must also allow for alterations to be made such as configuring minor information (inserting a new object or deleting an object) without having to reconstruct the index from the beginning.
(5) It also must not have any false dismissals i.e. include all the correct items within the indexing scheme.
(6) Two other properties that is very useful for an indexing scheme to have include: the index must be able to be built at a decent amount of time [34] and the index should also be capable of handling different distance measures.

Now let's go through the principles of employing an indexing scheme on time series data. For instance, given that a time series A can be considered as a point in an n-dimensional space. What this

suggests is that Spatial Access Methods (SAMs) can be used for effectively indexing the time series points in space. As SAMs allows you to partition space into different regions along a carefully constructed hierarchical structure for efficient access and retrieval. B-trees is an example of a hierarchical indexing structure method that is suitable for one-dimensional data but has been upgraded to also handle multi-dimensional data.
An example of a hierarchical indexing structure that can effectively handle multidimensional points in space refers to R-tree proposed by Beckmann et al. R-tree's uses data that is organized in minimum bounding rectangles (MBR), however, organising data in minimum bounding regions, means that the sequential nature of time series cannot be captured. The main shortcoming of this indexing method is that MBR leads to a large amount of empty space. This consequently causes queries to thus intersect with many of these MBRs.

Processing time series data can be difficult for most SAM approaches considering that time series data often contains over thousands of data-points and that SAM approaches are often known to worsen as the dimensionality increase above 8 to12. This issue of high dimensionality forces the method to access almost the entire dataset by random I/O. As a result, this makes indexing lose its values, considering that R-trees and other indexing methods similar to it are victims of the phenomenon known as the 'dimensionality curse' [35]. Also, there is a scalability problem to account for as the larger the data size gets, the more difficult it is for the indexing scheme to organise the data. This is considering that these data must be stored, hence causing the indexing scheme to run into a storage problem [36]. The best way to address this problem is to first perform dimensionality reduction. X-tree (extended node tree) is an example of this as it uses a different split strategy to reduce overlap. How it does this is through a vector quantization approach which basically quantizes the data space to store both lower and upper bounds. Another method which is ultimately an extension of R-tree is the TV-tree (telescopic vector tree). This indexing technique uses a number of minimum bounding regions such as spheres, rectangles or diamonds to reduce the layers of dimensionality.

Suffix trees proposed by Park et al. provided another new approach to index time series. The concept is based on computing the distance by firstly relying on comparing the prefixes first, hence allowing you to store every time series that share identical prefixes in the same set of nodes. The issue with this approach is that it's faced with scalability problems as it seems to not be able to handle longer time series. The Generic Multimedia indexing method (GEMINI) introduced is capable of handling any dimensionality reduction method in order to generate efficient indexing.
Yi and Faloutsos introduced an indexing scheme for time series where the distance function can be any Lp norm and only one index structure is needed to cover all Lp norms. A framework proposed by Vlachos et al allows you to index multidimensional time series with DTW as well as LCSS. Another indexing framework proposed by Assent et al is the TS-tree. The TS-tree enables you to efficiently perform similarity search on time series. The benefits of TS-tree is that It provides compact metadata information on subtrees and two it avoids overlap. This, as a result, reduces the overall search space, meaning that fewer computations would be needed to be made to query an object in space [37].

3 Detailed description of meta-clustering approach

3.1 Implementing meta-clustering approach via proposed Optimised K-NM algorithm for clustering time series data

The motive behind implementing an algorithm capable of segmenting, compressing, extracting and clustering a set of time series data is so that we could enhance the k-nearest mean algorithm with producing faster and more reliable results. By utilising a newly devised pattern recognition algorithm for processing large sets of time series data would lead to accurate clusters to be formed at a faster rate and less space to be consumed as opposed to older more conventional techniques.

An optimised K-nearest mean algorithm is implemented to process and cluster the time series data (representing the health state of IGBT components). The following algorithm executes the meta-clustering solution through two stages. The first stage consists of identifying the most relevant patterns related to each of the 22 components and as a result clustering them into separate clusters. How it does this via K_NM is it sets k separately for each different component and the n objects closest to k are thus clustered together using the Euclidean distance to determine the closeness of n objects to k. The following process is repeated for each of the 22 components in order to output 22 clusters. This stage would hence get rid of unnecessary data that is meaningless and thus prepare it for further processing. The second stage processes the 22 inner clusters into a single generic cluster. It thereby sorts all 22 gathered clusters into one optimised cluster that would effectively represent the failure state of all 22 clusters, accordingly. K_NM does this by examining the means for each cluster on each dimension to assess the level of similarity between each of the k clusters. Once the similarity of these clusters is established based on the Euclidean distance, k is randomly selected from the set of clusters as the general cluster and the remaining clusters are labelled as n objects. As a result, n clusters are assigned to the nearest centroid k which generates the optimised cluster.

The reason an optimised k-means algorithm was used to implement the meta-clustering approach is to render a drawback of the standard algorithm. This refers to its computation time, as the standard algorithm calculates the distance in each iteration even though in some iteration it is obvious that the object will belong to the same cluster. The optimised k means algorithm addresses this issue as well as sorts the failure objects within their respective clusters more accurately. It saves this extra time wasted by calculating whether the failure object from the new cluster centre is smaller than the distance of the failure object from the previous cluster centre. If it is smaller, then the object will belong to the same cluster, meaning there will be no need to repeatedly measure the distance of the object from other cluster centres. This is done by having two data structures where the label of the cluster and the distance of the object will be stored in the corresponding cluster centre and can be further used in subsequent iterations. This way we could keep track of the distances from the cluster centre and thus decide on either keeping the object within the same cluster or getting rid of it as it adds no value. The process of implementing the following optimized k-means algorithm is as follow:

- What we give as input is the number of desired clusters, k, and a time series dataset D=(d_1, d_2 ,……,d_n) containing n objects in space.
  - To obtain a set of k clusters
    1- A set of k data pattern are randomly selected from the dataset D as initial clusters.
    2- The distance between each data pattern (1<=i<=n) and each k cluster centre initially determined cj (1<=j<=k) is calculated to see which cluster it best fits using the Euclidian distance d(d_i, c_j) measuring tool.

3- Next for each data pattern d_i, find the closest cluster centre c_j and subsequently assign d_i to that cluster centre c_j which showed the highest level of similarity.
4- Store the label and distance as: Set cluster[i]=j, and dist[i]= $d(d_i, c_j)$. Where j is the cluster label in which data d_i reside and $d(d_i, c_j)$ is the distance of data object d_i to the cluster centre labelled by j.
5- For each cluster j (1<=j<=k) recalculate the cluster centre.
6- Repeat
7- For each data pattern d_i, calculate it's closeness to the centre of the current cluster; a) If this distance is less than or equal to dist[i], the data pattern will remain in its initial cluster; b) Else For every cluster centre c_j(1<=j<=k), it calculates the distance $d(d_i, c_j)$ and assigns the data pattern di to the nearest cluster. Set cluster[i] = j; Set dist[i] = $d(d_i, c_j)$.
8- For each cluster centre j (1<=j<=k), recalculate the centres to the point where the centre doesn't change anymore.
9- The final clustering results are respectively outputted.

As you can see from the algorithm above, the optimized algorithm requires two separate data structures to be implemented, one for storing the label of the cluster and the other for storing the label of the distance for each iteration. The distance between the two vectors x=(x1,x2,…….,xn) and y=(y1,y2,……,yn) are measured using the Euclidean distance measuring tool d(xi ,yi) as denoted in equation (1).

$$d(x_i, y_i) = \left[\sum_{i=1}^{n} (x_i - y_i)^2\right]^{1/2} \quad (1)$$

The key attribute of this optimised algorithm that makes it different as opposed to standard K-means algorithms is that it does not require the distance to be calculated during each iteration. This is based on the time complexity of this algorithm denoted by O(nk). So if for instance, a data point remains in its initial cluster then the time complexity will be O(1) otherwise O(k). However if half of the data points move from its initial cluster to another cluster due to the higher correlation the data points will have a new cluster, then the time complexity will be (nk/2). As such, the following optimised algorithm implemented effectively increases the speed of the standard k-means algorithm. The only remaining issue that remains for further research refers to the need for k value to be determined in advance. Thus for an even more optimal solution, it would be best to test it for different values of k which we look forward exploring and testing in the near future.

3.2 Results of Implementing Optimised Clustering Technique

This section describes the data experimented (case study), the results produced from the optimised clustering technique as well as the implications of using such a novel meta-clustering approach.

2-2.1 An overview of the case study (IGBT components)

IGBTs are the application of solid-state electronics. The main task of IGBT's is to control and covert electric power. It is also commonly used in the subject fields of electronic and electrical engineering which deals with the design, control, computation and integration of nonlinear, time-varying energy-processing electronic systems with fast dynamics. Common types of power electronics are DC to DC, AC to DC, DC to AC and AC to AC convertors which reflects the gathered time series data that will be used for advanced pattern recognition tests.

2-2.2 Importance of clustering IGBT components

Now many large systems are working under critical harsh environments are faced with a wide range of various reliability and safety issues in their everyday activities such as fatigue, degradation, challenging working conditions, and communication failure. Therefore it is apparent that such components can highly affect the overall condition of a large system which is why such advanced pattern recognition solutions will be implemented to effectively predict the prognostics of IGBT components by looking at the similarity shared with existing IGBTs and thus grouping them into one cluster. As such, the knowledge extracted from clustering these IGBT components refers to being able to establish when multiple IGBT components start to degrade and ultimately fail. Electronic engineers can learn from this knowledge and try to implement solutions to prolong the degradation of such components.

This is why clustering this data is very useful for electronic engineers, as it allows them to understand the health state of a number of components. As a result, they wouldn't have to test each individual component to identify whether they are healthy or degrading anymore as the clusters representing the health state of multiple components will effectively achieve this while consuming less time and space. This is because if more than one component is found to be similar, then they are grouped as one cluster, enabling electronic engineers to identify the health state of a number of components much faster and providing them enough time to avoid any potential faults and further be able to address these faults before they become critical and lead to major crashes in huge systems and applications such as Boing.

2-2.3 Conventional prognostics and diagnostic technique

In the field of condition-based management, prognostics is a controlled engineering discipline that focuses on the prediction of the future lifecycle of a system. This ultimately enables us to estimate when systems develop irregularities due to any kind of malfunction or failure. It is obviously a key step in arranging predictable maintenance for systems that experience irregularities in harsh environments. There are four technical approaches for building prognostic models: data-driven; model-based; hybrid; and finally, service-based approaches. Advantages and disadvantages of each approach are discussed below.

The model-based prognostic is a model definition that requires one to understand the physics-of-failure of a system. This is often a difficult task because establishing an accurate physical model is difficult to achieve. Data-driven approaches, on the other hand, start with the process of accelerated aging tests, followed by a filtering and classification phase [38]-[39].

The various techniques enable us to conduct accelerated aging and life cycling tests, presented in [40]. The two main types of life cycling tests for IGBTs are known as power cycle and thermal cycle. The first one is used to capture reliability information; and also, to evaluate the useful life of the bond wires of IGBT. The second IGBT life cycle test is used to evaluate solder joints and module base plate. In contrast with power cycling, temperature intervals are much longer in the case of a thermal cycle.

Filtering and classification steps are required to eliminate noise from the sensory data, and to recognize life durations of the system in the form of health states. Life durations are optimised among a number of different degradation profiles that are collected from different aged components/systems. The optimised life durations are then used to create the damage model needed for obtaining the initial threshold (Th) under normal operating conditions, ideal for RUL (IR), and the prognostic model. The following conventional techniques suggested so far for achieving the prognostics of systems are expensive, inaccurate and time consuming to develop which is why we look at developing pattern recognition solutions.

3.3 Experimented Time Series data set

2-3-1. Understanding the tested time series data

The following research requires experimented results to be collected from the reliability information of a real component (an IGBT component as a case study). As briefly described above, IGBTs are important components found within large systems which carry electrical and thermal contact, and electrical insulation where needed. It is no secret that these components are vulnerable to potential faults and failures, which could consequently affect the overall reliability of the whole system. The reliability information collected from such components is monitored by a wide range of sensors including temperature, voltage and current sensors. These sensors collectively monitor the health states of the experimented IGBTs. Once this information is monitored, it is then stored and transmitted as time series data, and thus prepared for further processing such as pre-processing and clustering.

The gathered time series data (kindly provided by Cranfield University) is recorded in real time and represents a collection of values that were obtained from sequential measurements over a continues time interval. Time series data is defined as follow:

A time series T is an ordered sequence of n real-valued variables; $T = (t_1,...,t_n), t_i \in \mathrm{R}$

The time series data being analysed and processed are obtained from 22 IGBT components, which are monitored by a set of sensors. The time series values represent the health state of the components that are recorded every 2 seconds until the components eventually fail. Therefore the following time series data obtained from the actual life cycle of the power modules being monitored are stored as vectors in a 2-dimensional matrix, consisting of 5130 rows by 22 columns. 5130 refers to the data values (also known as failure patterns) obtained from the sensors recording the life cycle of the power modules every 2 seconds and 22 refers to the number of power modules that were recorded. This means that there are 22 components that were experimented and from those 22 components 5130 data entries

were obtained which reflect the life cycle of each power module from the time which they are healthy to the time where they start to degrade and eventually fail. Further information on the meaning of these time series data and how they were processed and represented can be found in a IEEE transactions on power electronics journal published by me 2015 [41].

It is important to consider three main implementation components whilst dealing with time series data. The first one refers to the representation of data, so how can the time series data be represented; such that the dimensionality of data can be reduced while its essential characteristics are preserved. The next two components refer to the similarity measurement and the index methodology employed, accordingly. Similarity measurement 'within the context of time series' is about distinguishing and/or matching any pair of time series. Hence, how can a reasonable distance between two series (that are not mathematically identical) be recognised that is both accurate and efficient. And finally, the Indexing methodology that is implemented to organise the large set of time series data for effective clustering. The indexing mechanism used in this project will provide minimal space consumption and computational complexity. These implementation components are the essential aspects of time series data mining that comprise of the overall implementation of this research.

1. Processing the extracted multi-variate time series data

The meta-clustering approach was tested on the time series dataset provided by Cranfield University. These time series data reflect the health state of multiple power modules, ranging from the time which they were healthy to the time to which they start to develop faults and ultimately fail. The time series data extracted reflects the values that were recorded by n sensors at time (t0). The past data and samples sensed before t0 ranging from t_{-1} to t-h along with q number of mean values from samples taken before t-h are stored as a vector. By applying this technique to every *n* sensor, a 2-dimensional matrix, A(n.m) is created. Therefore the values found inside the time series data show how each of the 22 power modules is degrading until they ultimately fail.

As such, an optimised k-means algorithm is applied to this data to first of all cluster the failure profiles related to each of the 22 power modules into separate clusters and then once these 22 inner clusters are formed they are further clustered into a general cluster that would effectively represent the failure state of all 22 power modules. The reason this preliminary step is initially conducted is to gather the right set of failure patterns and remove any data that adds no value towards better understanding the failure health state of the power module. And then they are further clustered into an optimised generic cluster so that the failure health state of all 22 power modules clustered can be understood based on the similarity shared. This, as a result, allows us to more efficiently and effectively process and cluster multi-variate time series data as opposed to other pattern recognition approaches that do not go through these extra stages. Considering that these extra stages lead to a lot of space to be saved, this means more accurate results will be achieved and less time will be consumed when computing the clusters.

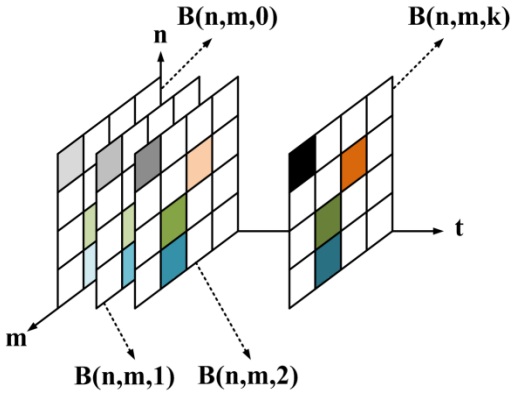


Fig.1 time series data stored as a two dimentional matrix:

The Size and complexity of matrix A(n,m) are dependent on the complexity of the power module, but it still continues to remain as a two-dimension matrix. Considering the power module is operating in real time, new samples are collected from time to time, so during each time step, t_1 will be fed to t_0, t_0 to t_{-1} and so on, which eventually gets rid of the sample that was taken at t_{-m}. Hence, the matrix of sensory and mean values shown in Fig.1 is updated; accordingly, that in turn creates a 3-dimension time-dependent matrix, B(n,m,t). The first initial 2-dimension matrix, A(n,m) that refers to a healthy system is equal to B(n,m,0) at t_0. Any other 2-dimension matrix from B(n,m,t) at given *t* would be a degraded form of B(n,m,0). Therefore, B(n,m,t) is a collection of 2-dimensional matrices that presents a movie if we were to see 2-dimensional matrices as images. Failures appear as shape, colours, and edges of objects that are changed in the movie-like degradation profile of a system. Fig.1 shows an example of such a movie-like degradation profile. Further information on how these time series data were processed and represented as a two dimensional matrix can be found in a conference paper which was published by me in 2014 [42].

MATLAB 7.11.0 (R2010b)

File Edit View Graphics Debug Parallel Desktop Window Help

Current Folder: C:\Windows\system32

Shortcuts How to Add What's New

Variable Editor - all_data_johnson

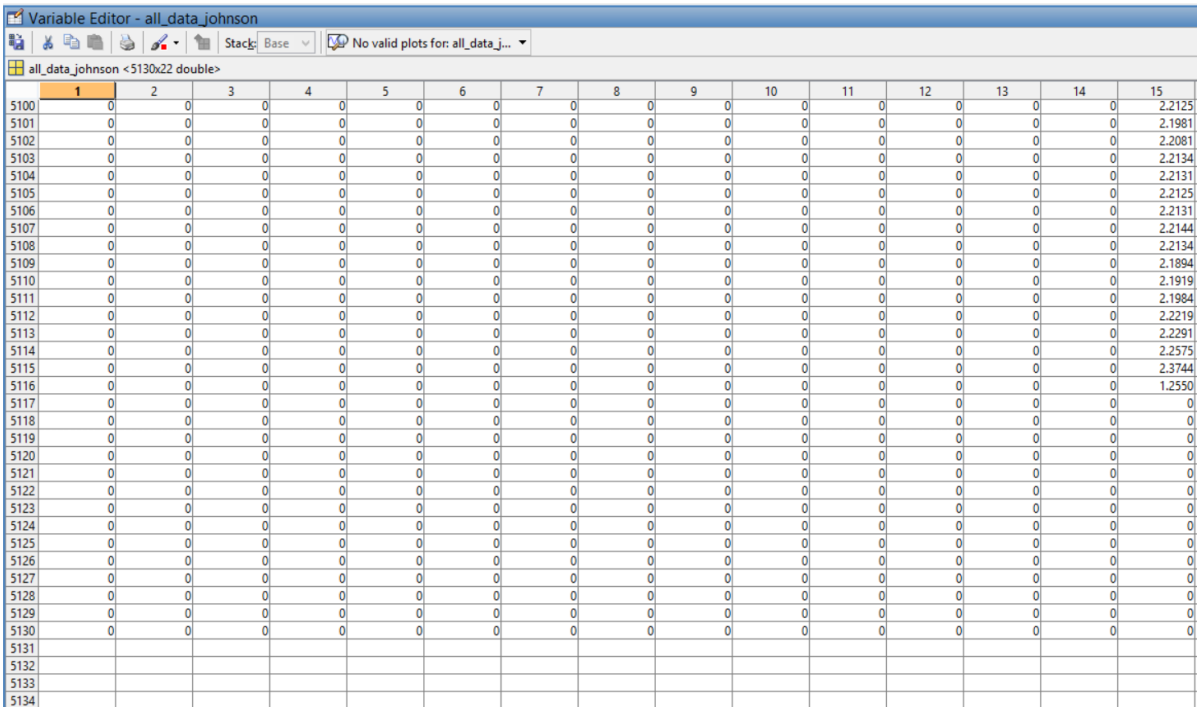
Stack: Base No valid plots for: all_data_j...

all_data_johnson <5130x22 double>

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.0387	0.0370	0.0015	0.0429	8.3333e-04	0.0023	0.0492	0.0518	1.1155	0.1018	0.0505	0.0521	0.0437	0.0034	0.0569
2	1.8491	1.7721	2.0056	2.0565	1.9656	2.0683	1.8947	1.9666	1.9279	1.9139	1.9339	2.0192	1.8934	1.7724	1.9806
3	2.0283	1.7670	2.0319	2.0452	1.9883	2.1017	1.9816	2.0316	1.7453	2.0221	1.9297	2.0055	1.9932	2.0132	2.0303
4	2.0066	1.9409	2.0298	2.0450	1.9894	2.0948	2.0382	2.0342	1.8658	2.0424	1.9297	2.0071	1.9821	2.0118	2.0366
5	2.0043	1.9434	2.0298	2.0467	2.0029	2.0967	2.0411	2.0345	2.1405	2.0255	1.9305	2.0071	2.0105	2.0092	2.0113
6	2.0036	1.9366	2.0308	2.0485	1.9994	2.0988	2.0376	2.0300	2.1853	2.0089	1.9305	2.0045	1.9226	2.0321	2.0103
7	2.0036	1.9451	2.0304	2.0483	2.0042	2.0952	2.0397	2.0313	2.1061	2.0053	1.9318	2.0047	1.9313	2.0084	2.0113
8	2.0043	1.9449	2.0302	2.0496	2.0113	2.0973	2.0395	2.0326	2.0361	2.0092	1.9300	2.0055	1.9445	2.0084	2.0103
9	2.0036	1.9428	2.0304	2.0481	1.9944	2.0977	2.0400	2.0342	2.0087	2.0116	1.9300	2.0021	1.9455	2.0068	2.0113
10	2.0023	1.9421	2.0300	2.0406	1.9927	2.0960	2.0397	2.0353	2.0105	2.0108	1.9300	2.0011	1.9468	1.9855	2.0103
11	2.0036	1.9474	2.0331	2.0438	1.9948	2.0969	2.0400	2.0347	2.0039	2.0092	1.9295	1.9984	1.9424	1.9934	2.0116
12	2.0036	1.9462	2.0335	2.0398	1.9958	2.0965	2.0400	2.0350	2.0116	2.0108	1.9300	1.9982	1.9413	1.9879	2.0119
13	2.0019	1.9447	2.0325	2.0396	1.9940	2.0942	2.0400	2.0326	2.0108	2.0134	1.9297	1.9989	1.9355	1.9939	2.0103
14	2.0019	1.9394	2.0300	2.0388	1.9935	2.0944	2.0397	2.0303	2.0116	2.0116	1.9297	1.9995	1.9379	1.9845	2.0109
15	2.0074	1.9445	2.0317	2.0398	1.9915	2.0925	2.0400	2.0321	2.0105	2.0118	1.9292	1.9961	1.9403	1.9992	2.0109
16	2.0072	1.9415	2.0304	2.0408	1.9913	2.0885	2.0400	2.0276	2.0132	2.0134	1.9300	1.9984	1.9374	1.9826	2.0109
17	2.0057	1.9379	2.0300	2.0402	1.9910	2.0967	2.0400	2.0316	1.9976	2.0126	1.9300	1.9979	1.9434	1.9884	2.0106
18	2.0057	1.9421	2.0296	2.0402	1.9888	2.0940	2.0400	2.0263	1.9834	2.0087	1.9300	1.9979	1.9379	1.9926	2.0106
19	2.0057	1.9438	2.0296	2.0392	1.9856	2.0954	2.0389	2.0292	2.0150	2.0111	1.9300	1.9989	1.9368	1.9800	2.0238
20	2.0077	1.9443	2.0302	2.0390	1.9894	2.0977	2.0395	2.0255	1.9876	2.0079	1.9300	1.9982	1.9387	1.9805	2.0288
21	2.0081	1.9453	2.0300	2.0358	1.9913	2.0963	2.0400	2.0268	2.0032	2.0053	1.9295	1.9979	1.9361	2.0111	2.0291
22	2.0047	1.9413	2.0310	2.0406	1.9860	2.0948	2.0400	2.0226	2.0234	2.0071	1.9300	1.9976	1.9400	1.9837	2.0275
23	2.0038	1.9432	2.0304	2.0358	1.9919	2.0977	2.0400	2.0284	1.9932	2.0089	1.9300	1.9968	1.9416	1.9832	2.0269
24	2.0064	1.9447	2.0304	2.0373	1.9873	2.0954	2.0400	2.0292	1.9745	2.0124	1.9300	1.9958	1.9376	2.0050	2.0269
25	2.0064	1.9449	2.0304	2.0398	1.9888	2.0977	2.0400	2.0268	2.0053	2.0113	1.9300	1.9966	1.9379	1.9947	2.0278
26	2.0043	1.9428	2.0300	2.0392	1.9933	2.0952	2.0403	2.0271	2.0171	2.0139	1.9300	1.9961	1.9366	1.9716	2.0275
27	2.0045	1.9451	2.0310	2.0396	1.9921	2.0969	2.0400	2.0276	1.9961	2.0058	1.9300	1.9979	1.9384	1.9676	2.0263
28	2.0083	1.9430	2.0302	2.0371	1.9938	2.0956	2.0397	2.0271	2.0016	2.0108	1.9300	1.9976	1.9403	2.0045	2.0266
29	2.0055	1.9436	2.0304	2.0400	1.9865	2.0979	2.0384	2.0237	2.0200	2.0082	1.9300	1.9966	1.9366	1.9795	2.0269
30	2.0040	1.9434	2.0300	2.0388	1.9904	2.0965	2.0384	2.0255	2.0055	2.0076	1.9300	1.9979	1.9366	1.9755	2.0256
31	2.0045	1.9409	2.0310	2.0396	1.9892	2.0975	2.0392	2.0237	2.0103	2.0089	1.9300	1.9958	1.9413	1.9845	2.0250
32	2.0068	1.9447	2.0317	2.0379	1.9860	2.0981	2.0395	2.0268	1.9863	2.0071	1.9300	1.9961	1.9379	2	2.0259
33	2.0049	1.9447	2.0317	2.0396	1.9963	2.0977	2.0397	2.0279	2.0224	2.0100	1.9300	1.9953	1.9400	2.0092	2.0253
34	2.0062	1.9409	2.0313	2.0390	1.9885	2.0963	2.0395	2.0284	2.0068	2.0045	1.9300	1.9963	1.9408	1.9847	2.0266
35	2.0062	1.9443	2.0313	2.0396	1.9902	2.0958	2.0392	2.0261	2.0124	2.0111	1.9300	1.9974	1.9382	2.0195	2.0241

Table 5: Time series data processed and experimented (Degrading state of IGBTs)

The above table (Table 5) illustrates the raw data that was processed within the Matlab environment. Evidently, it is a 5130 by 22 matrix which indicates that there 5130 data values that were recorded by the sensors every 2 seconds and as time went on the values start to decline indicating that the power modules are steadily degrading. As it reaches the 5130 time point, all 22 power modules have failed, as shown below in Table 6. Therefore this is the data that is effectively clustered in order to make better sense of the data, save a large amount of space and cluster the failure health state of all 22 power modules as one optimised general cluster, hence saving a lot of time and resources.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2125
5101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.1981
5102	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2081
5103	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2134
5104	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2131
5105	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2125
5106	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2131
5107	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2144
5108	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2134
5109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.1894
5110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.1919
5111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.1984
5112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2219
5113	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2291
5114	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.2575
5115	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.3744
5116	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.2550
5117	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5118	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5119	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5121	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5123	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5124	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5125	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5126	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5129	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5130	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5131															
5132															
5133															
5134															

Table 6: Time series data processed and experimented (Failed state of IGBTs)

3.3.2 Results obtained from implementing a Meta-Clustering approach

As previously discussed, the failure health state of the power modules represented as multivariate time series data are clustered via two stages: The first initial stage refers to the inner clusters whereby the 22 power modules are separately clustered and the second stage refers to the optimised general cluster which generalises the health state of all 22 power modules as one.

3.2.3 Inner Clusters

Figure 1 below illustrates the results of separately clustering all 22 power modules into the initial inner clusters.

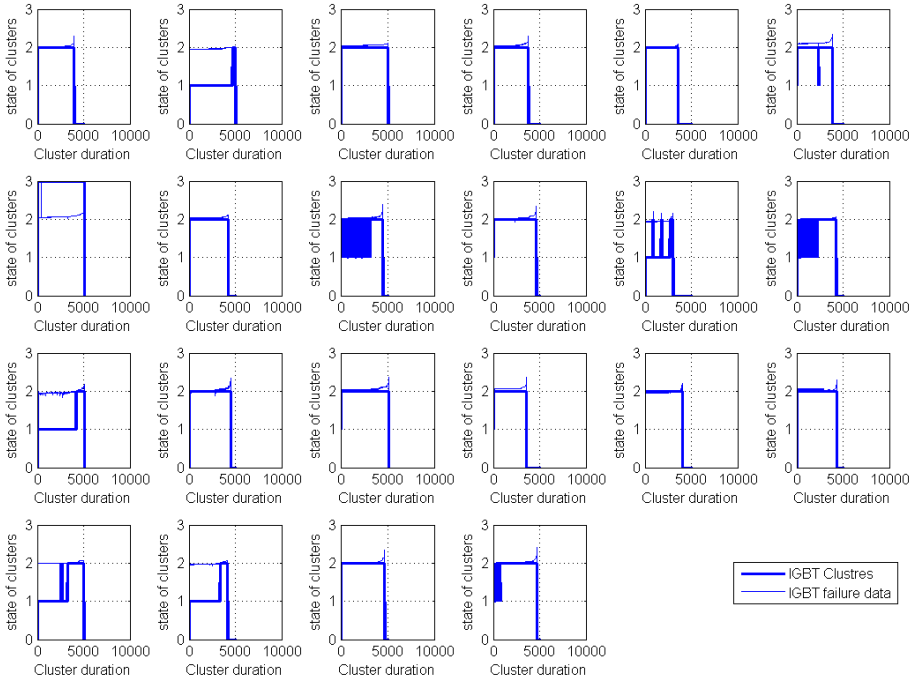


Fig.2 Clustering 22 power modules into separate meta-clusters

As indicated by figure 2, 22 separate clusters representing the health state of each power module have been effectively partitioned. The graphs in figure 2 identify the failure profile of each power module. The light blue line indicates the failure data that shows each individual power module degrading until it completely fails shown by the flat line. The thick blue line is the initial clusters that are formed around the failure profile for each individual power module in order to successfully prepare it for further clustering.

Figure 3 below is a much more in-depth representation of figure 2, which provides a much clearer understanding of the failure data and the outcome of the clusters formed.

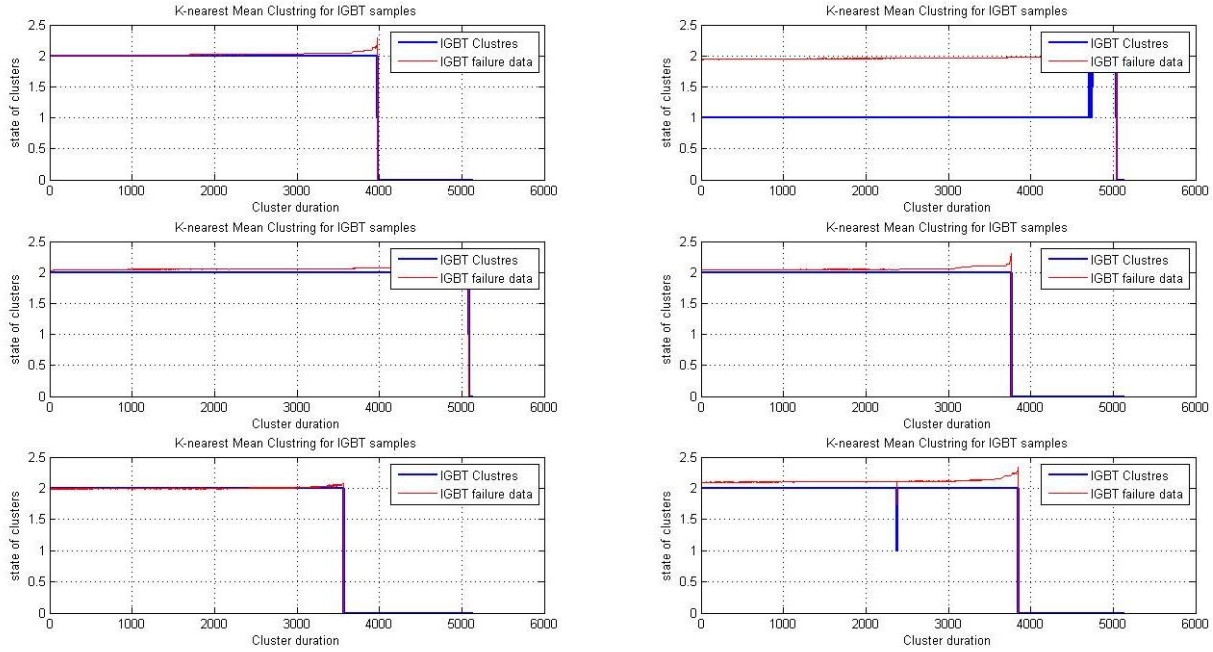


Fig.3 In-depth look at 6 out the 22 Clustered IGBT components

To gain a better understanding of IGBT components, IGBT is a three-terminal power semiconductor device. IGBT's are commonly used as a switch. As further progress was made, it came to be well known for being very efficient and fast for switching electric powers in many modern devices [43]. These components found in most devices have a certain duration to which they start to develop faults. IGBT's main characteristics refer to voltage (V) and current. Based on these characteristics of an IGBT component, the failure data during a certain time period is captured and clustered, accordingly.

Figure 3 shown above depicts a more detailed representation of the failure state of the IGBT components and the clusters formed. To shed more light on what the failure data represents (showed by the red line in the graphs; figure 3); during the initial stages, general and worst failing scenarios are considered for degrading IGBTs using a test rig created. In this regard, IGBT's power is cycled so that IGBT's temperature is kept between the minimum and maximum limits (Tmin and Tmax, respectively).

The power cycling test is repeated for 22 IGBTs; and all the monitored parameters including on-state voltage, off state current, environment temperature are saved within the nth degradation profile for the chosen nth IGBT under aging test, $1 \leq n \leq 15$ and $n=\{1, 2, \ldots, 25\}$. However, for simplifying the process, we proceed with just one set of working conditions, a fixed Tmin, Tmax, and load. Run to failure results of power cycling tests conducted on five samples of the same IGBT. Tests are processed under working conditions of Tmin=60°C and Tman=120°C.

Now that it is clear what the failure data shown in figure 3 represents, the failure data (red line) related to each IGBT component is subsequently clustered (as shown by the blue line). So the blue line formed around the red line reflects the clusters that are formed for each separate component depicted by the graphs. This, as a result, provides us with 22 separately clustered IGBT components. This is considering that the red line shows the actual failure data from the point where the IGBT component starts to degrade and ultimately fail. And then the blue line formed around the red line of each IGBT component reflects the clusters that were formed. The reason these inner clusters were initially created was to prepare and organise the best set of refined clusters for ultimately clustering all

22 components together as a single cluster. As these initial inner clusters better represent the health state of these IGBT components as it gets rid of any unnecessary data that adds no value which also leads to the clustering process to be faster as there will be fewer data to process. The advantage of this is that individual tests will no longer need to be conducted on different IGBT and new IGBT's can be immediately sorted into their right clusters based on the similarity they share with existing IGBT's. The overall implication of this approach is that the health state of a component (represented as time series data) can be clustered at a faster rate while consuming less space.

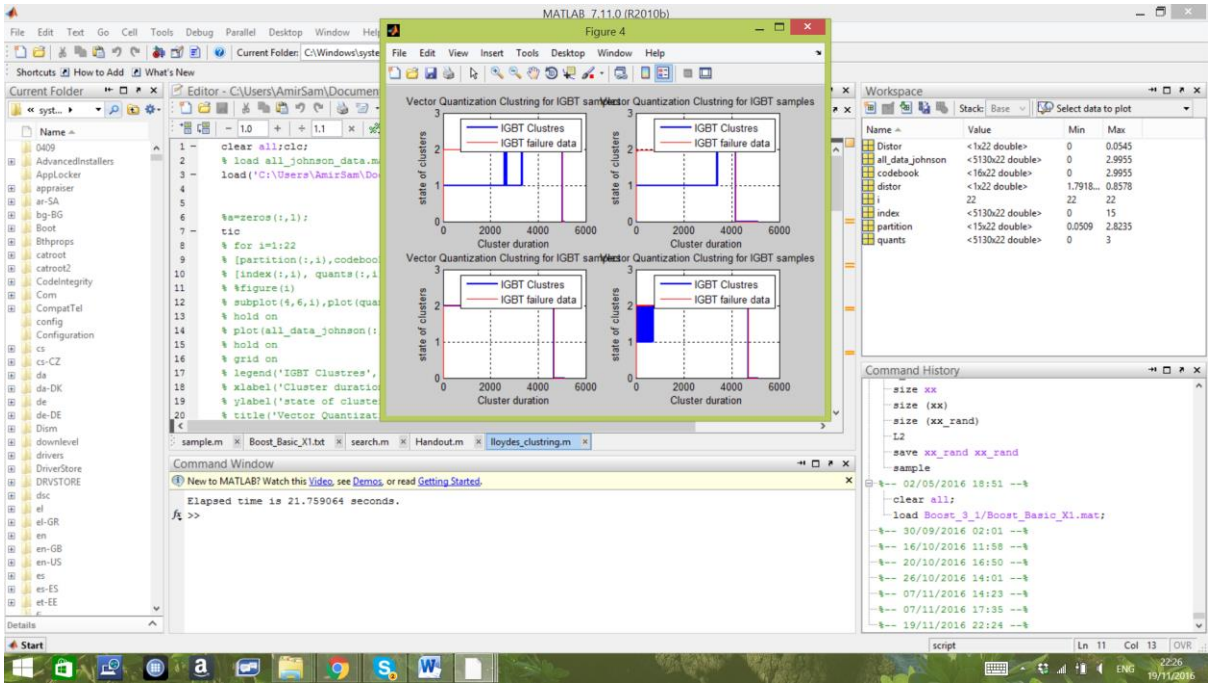


Fig.4: Screenshot of creating the initial inner clusters

The screenshot above illustrates the results obtained and the time taken to run the algorithm within the Matlab framework. After running the algorithm, the results shown in the graphs of figure 3 are outputted, and the time it took to output these initial inner clusters is also calculated. In the first stage of the meta-clustering approach, the time taken to initiate these 22 inner clusters came to around 21.76 seconds as shown in the lower panel of the screenshot in figure 4. This shows that the required clusters can be achieved quite fast and therefore is worth creating these inner clusters in order to prepare it for the second stage which groups the inner clusters into a refined general cluster that can more effectively represent the failure state of all 22 clusters by consuming less time and space.

3.2.4 Optimised General Cluster

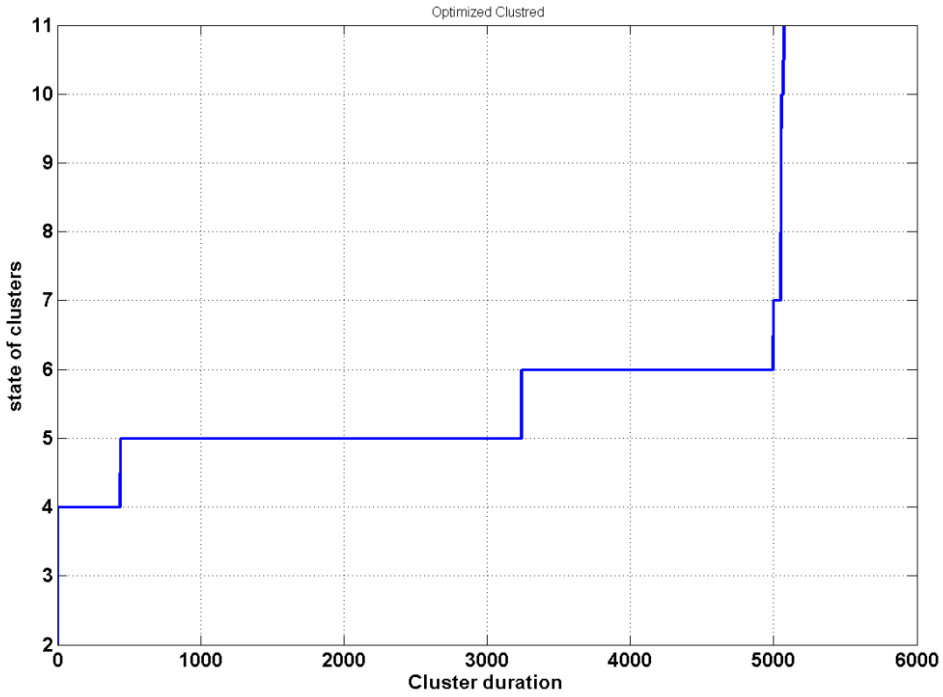


Fig.5 Optimised General Cluster

Figure 5 shows the result obtained from optimising 22 clustered IGBT components into one general cluster. This means that the failure health state of 22 IGBT components that were initially clustered separately are now generalised as a single cluster. The blue line in figure 5 represents the degradation cycle of these 22 IGBT components. The graph above shows at what point the 22 IGBT components fails which when it reaches the 5100 time mark. The reason these 22 IGBT components could be clustered as a single component is that they were similar and therefore could be grouped together using this enhanced meta-clustering approach.

Therefore by clustering 22 different IGBT components, the health state of 22 IGBT components can be identified just by using a single general cluster. Applying this meta-clustering approach enables us to cluster these 22 components at a much faster rate and ensures the results achieved are more reliable as a result of the initial clusters that were created (shown in figure 3). Furthermore, by clustering these 22 components as a single cluster means that each component will no longer have to be tested separately. This is particularly helpful as there are more than 1000s of IGBTs. Therefore by grouping a majority of these IGBTS into a single optimised cluster will save a lot of time and resources.

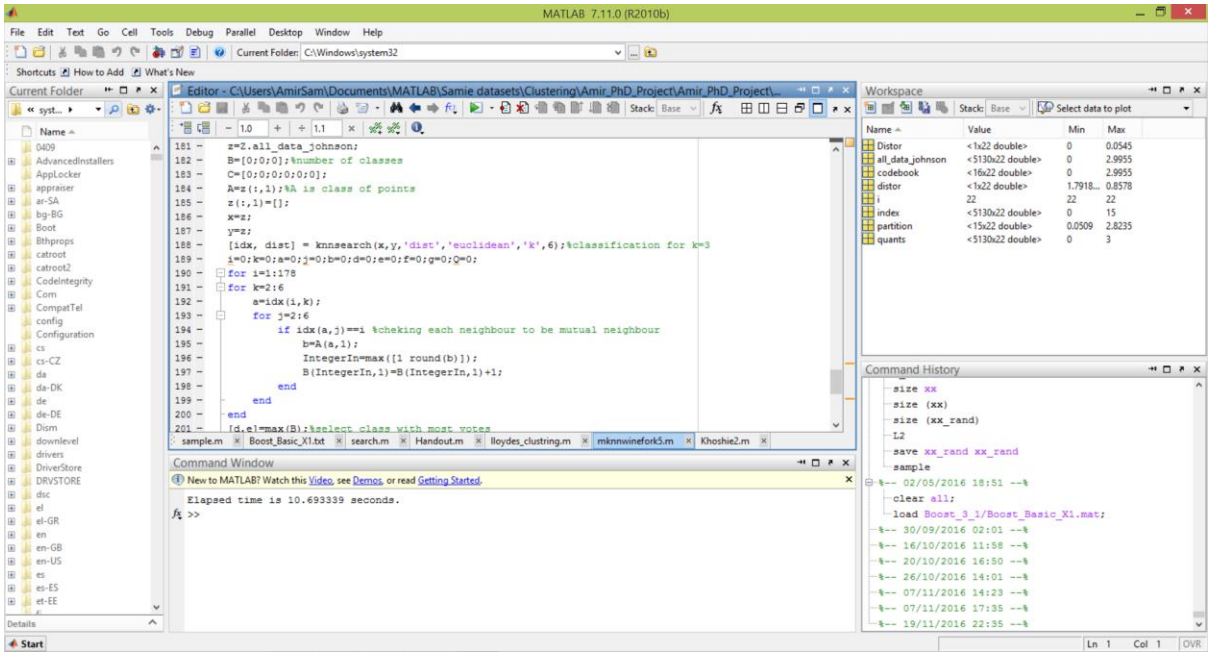


Fig.6: Screenshot of creating the optimised general cluster

The screenshot above shows the time it took, and a snippet of the coding implemented to run the second phase of the meta-clustering approach. This refers to clustering the 22 inner clusters created into a single general cluster. As shown in the coding section, the 'k-nearest neighbour' algorithm was used to create this generalised single cluster, and the distance between the 22 inner clusters was calculated with the Euclidean distance measure. Once the algorithm checks that each of the clusters is classed as a mutual neighbour, the inner clusters were automatically grouped as a general cluster.

The following algorithm took about 10.7 seconds to compute; meaning much less time was consumed compared to the first phase (figure 4) which took 21.6 seconds. This is mainly because of the meta-clustering approach developed as it initially prepares a more refined set of inner clusters and then generalises these inner clusters into a general cluster, hence providing a novel approach for pattern recognition techniques to efficiently cut down on time, resources and space.

4. Evaluation and lessons learned

4.1 Evaluation and review

In the following, we will be contrasting this meta-clustering approach with those approaches which are considered as the most relevant and competitive ones.

Another type of clustering method that also tries to partition patterns in similar and identifiable groups is the Hypergraph-based clustering [44]. The data that was used by hypergraphs for clustering are transactions of store items purchased by customers. This data can be represented by a set of very sparse binary vectors, i.e. one for each transaction. Hypergraphs are an enhancement over regular graphs which allow edges in a graph to join multiple vertices, instead of the usual two vertices. The necessary steps involved in the process of hypergraph clustering are as follow: 1) Setting out the rules and condition for connecting several objects, (considering each object is represented as a vertex of the hypergraph) by a hyperedge. 2) Defining a measuring procedure to assess the strength or weight of a hyperedge. 3) Then, a graph-partitioning algorithm is used to segment the hypergraph into two parts so that it would result in the weight of the hyperedges cut to be minimised. 4) Carry on with the partitioning algorithm until a sufficient number of partitions are achieved whereby a cluster would no longer be needed. This algorithm's primary attribute is that it transforms the problem from high dimensional data space into a well-studied graph-partitioning problem that can be efficiently solved.

As compared to the optimised k-means clustering used in this project, Hypergraph-based clustering is an approach that is more suited to deal with a high number of dimensions, via hypergraphs. At this stage, the time series data reflecting IGBT components does not have too many dimensions which is why it was not a major concern for this particular project. However in the future, when dealing with data that has a higher number of dimensions such as larger systems, clustering approaches such as Hypergraph-based clustering will be considered to better deal with high dimensionality.

In spite of hyper-based clustering method performing better in regards to dimensionality, the optimised k-nearest means algorithm has the advantage of clustering the objects/patterns within their right groups at a faster rate. This is evident from applying the hyper based clustering algorithm on the tested time series data set. The time taken to cluster 22 components into 1 cluster was about 8 minutes whereas the meta-clustering approach took about 10 seconds. This can be mainly contributed to the preliminary stage that was initially conducted to achieve the 22 inner clusters, as this saves time by getting rid of the vast amount of unnecessary data as well as ensuring that the most relevant set of objects are prepared for the generalised clustering. Also since the initial inner clusters prepared results in a lot of useless data to be erased means that when running the optimised algorithm to generalise the 22 inner clusters leads to fewer data needing to be processed, hence making the overall algorithm faster.

This cannot be said for hyper based clustering, as a vast amount of vertices has to be set out to represent each object. This will consequently take longer to compute each object while trying to achieve a sufficient number of partitions. As by using the meta-clustering approach, we no longer have to calculate the distance of each object from each cluster centre during every loop execution. How it does this is by calculating if the distance of the object from the new cluster is smaller than the distance of an object from the previous cluster. If it is smaller, then the algorithm keeps that object within the same cluster. This means that it does not repeatedly calculate the distance from other data objects, and hence by doing so, it can cluster the objects at a faster rate.

OptiGrid is a grid-based clustering method used for data dependent partitioning. OptiGrid partitions each dimension using a variable number of "adaptive intervals", which more extensively captures the degree of distribution in that dimension, rather than dividing the data into a pre-defined number of evenly spaced intervals. While doing so, OptiGrid simply looks for the best cutting planes and creates a grid that is not likely to cut any clusters. It then locates potential clusters among this set of grid cells and further partitions them if possible, making it a useful technique as compared to other clustering methods [19, 33].

With that said, OptiGrid main trait is it being very effective at getting rid of noise. OptiGrid can automatically identify and eliminate noise in data by understanding that noise often appears to correspond to uniformly distributed data. As a result, it uses a more centralised type of distribution such as the 'Gaussian distribution' to filter out the noise. This is particularly useful for the time series data used in this project, as the gathered data from IGBT components and systems are known to be subject to much noise. In this project, standard filtering techniques are used to pre-process the data before clustering, but the only problem is that the results obtained were still quite noisy. Therefore, in the future, much more emphasis will be placed on effectively getting rid of noise by using a technique similar to OptiGrid.

4.2 Conclusion

The work done so far shows that an optimised k-nearest means algorithm can be used to cluster a number of electronic components (extracted as time series data) in order to obtain their health state and by doing so be able to predict its future state in a much more effective and efficient way as compared to standard prognostic methods that were extremely time consuming and required significantly more resources. In the future, I intend to further improve the k-means algorithm so that it can better deal with high dimensional time series data. By doing so, the clustering algorithm can be applied to much larger and more diverse applications and systems. This would overall, further validate the implications of my project, by showing that the clustering algorithm can indeed handle more diverse and complicated data and thereby can deal with many of the pattern recognition and clustering issues faced in the research community today such as dimensionality, scalability, noise, usability and a lack of intelligent communication between different systems; regardless of the size or complex nature of the data.

A pattern search feature will also be added to the k-means clustering approach. This would overall make the algorithm more effective, as the failure state of a specific system or component that is similar to the query can be retrieved. So in addition to clustering the failure state of similar components in a set of groups, they can now be queried as well which would mean the failure state of a system/component can be identified more precisely and at a faster rate. Lastly, the Ph.D. project will conclude by evaluating the tests and experiments of the project against strict evaluation search criteria to justify how and why and to what extent it is better than the current state of the art pattern recognition and search approaches.

References:

[1] J. Zhao, Z. Dong and Z. Xu (2006), **Effective feature processing for Time Series Forecasting**, International Conference on Advance Data Mining and Applications, pp 769-781, Available at: https://link.springer.com/chapter/10.1007/11811305_84

[2] Yousef M. Hawwar, Ali M. Reza, Robert D. Turney (2000), **Filtering (Denosing) in the Wavelet thresholding Domain**, DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, Available at: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C308311E351238B275ABE9F8844E5F43?doi=10.1.1.621.9184&rep=rep1&type=pdf

[3] E. Keogh and M. Pazzani, (1998). **An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback**. In Proceedings of the 4th International Conference of Knowledge Discovery and Data Mining. AAAI Press, 239–241.

[4] S. Jeng, and Y. Huang, (2008). **Time series classification based on spectral analysis**. Communications in Statistics-Simulation and Computation 37, 1, 132–142.

[5] J. Rodriguez and L. Kuncheva, (2007). **Time series classification: Decision forests and SVM on interval and DTW features.** In Proc Workshop on Time Series Classification, 13th International Conference on Knowledge Discovery and Data mining.

[6] P. Esling and C. Agon. (2012), **Time series data mining,** ACM Computing Surveys (CSUR) Volume 45 Issue 1, Article No. 12

[7] P. Berkhin, (2006), **A survey of clustering data mining techniques**. Grouping multidimensional data, 25–71.

[8] M. Corduas, M. and D. Piccolo, (2008). **Time series clustering and classification by the autoregressive metric**. Computational statistics & data analysis 52, 4, 1860–1872.

[9] M.Vlachos, J. Lin, E. Keogh and D. Gunopulos, (2003). **A wavelet-based anytime algorithm for k-means clustering of time series**. In Proc. Workshop on Clustering High Dimensionality Data and Its Applications. 23–30.

[10] G. Cormode, S. Muthukrishnan and W. Zhuang, (2007). **Conquering the divide: Continuous clustering of distributed data streams**. In IEEE 23rd International Conference on Data Engineering, 2007. 1036–1045.

[11] X. Pan, D. Papailiopoulos, S. Oymak, B. Recht, K. Ramchandran and M., I. Jordan, (2015)**. Parallel Correlation Clustering on Big Graphs**, Available at: https://papers.nips.cc/paper/5814-parallel-correlation-clustering-on-big-graphs.pdf

[12] R.Sathya and A.Abraham (2013), **Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification**, (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No. 2, Available at: https://thesai.org/Downloads/IJARAI/Volume2No2/Paper_6-Comparison_of_Supervised_and_Unsupervised_Learning_Algorithms_for_Pattern_Classification.pdf

[13] S. Tiwari and T. Solanki (2013). **An optimised approach for k-means clustering,**
9th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Available at: http://research.ijcaonline.org/qshine/number1/qshine1302.pdf

[14] C., M. Poteras, M., C. Mihaescu and M. Mocanu (2014). **An optimized version of the K-Means clustering Algorithm,** Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 695–699

[15] U. R. Raval and C.Jani (2016), Implementing & improvisation of K-means Clustering Algorithm, International Journal of Computer Science and Mobile Computing, Vol.5 Issue5, pg. 191 – 203, Available at: http://www.ijcsmc.com/docs/papers/May2016/V5I5201647.pdf

[16] S. Tiwari and T. Solanki (2013). **An optimised approach for k-means clustering,**
9th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, Available at: http://research.ijcaonline.org/qshine/number1/qshine1302.pdf

[17] Z.Zahra, D.Kang, M. R Ashrafi (2015), A new hierarchical clustering algorithm, Intelligent Informatics and Biomedical Sciences , Available at: http://ieeexplore.ieee.org/document/7439517/

[18] W. Zheng and L. Wang, (2015). **Parallel hierarchical clustering in Linearithmic time for Large-scale sequence Analysis**, Data Mining (ICDM), 2015 IEEE International Conference, Available at: http://ieeexplore.ieee.org/document/7373335/

[19] C. Shah and A. Jivani, (2013), **Comparison of data mining clustering algorithms,** Engineering (NUiCONE), Available at: http://ieeexplore.ieee.org/document/6780101/

[20] R.Lasri (2016), Clustering and Classification using a self-organising MAP: The main flaw and the improvement perspectives, SAI Computing Conference (SAI), Available at: http://ieeexplore.ieee.org/document/7556150/

[21] J. Garriga, J. R.B Palmar, A. Oltra, F. Bartumeus (2016), Expectation-Maximization Binary Clustering for Behavioural Annotation, Available at: http://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0151984&type=printable

[22] O. Abu Abbas, (2008), **Comparison between data clustering algorithm,** The International arab journal of information technology, Vol 5, No. 3, Available at:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.167.671&rep=rep1&type=pdf

[23] Sardá-Espinosa, Alexis. (2017). **Comparing Time-Series Clustering Algorithms in R Using the dtwclust Package**, Available at: **https://cran.r-project.org/web/packages/dtwclust/vignettes/dtwclust.pdf**

[24] N. Sanyal, (2015), **Fuzzy VQ based image compression by bacterial foraging optimization algorithm with varying population**, Available at:
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7060121&filter%3DAND%28p_IS_Number%3A7060104%29

[25] A.H Daptardar, J.A Storer, (2006), **Reduced complexity content-based image retrieval using vector quantization,** Avaialble at:
http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1607269:&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D1607269%3A

[26] S. Yang, T. Shen, X. Liu, W. Qi, (2014), **Vector Quantization Codebook Design and Speech Recognition Based on Immune Cat Swarm Optimization Algorithm, Pattern recognition and Artificial Intelligence**, Vol. 27 Issue (7): 577-583, Available at: http://manu12.magtech.com.cn/Jweb_prai/EN/abstract/abstract9883.shtml#

[27] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh (2008). **Querying and mining of time series data: experimental comparison of representations and distance measures**, The ACM Collection on Digital Content, Available at: http://www.cs.ucr.edu/~eamonn/vldb_08_Experimental_comparison_time_series.pdf

[28] D. J. Berndt and J. Clifford, (1994). **Using dynamic time warping to find patterns in time series**. In KDD Workshop.

[29] E. J. Keogh and C. A. Ratanamahatana (2005). **Exact indexing of dynamic time warping**. Knowl. Inf. Syst., 7(3)

[30] L. Chen, M. T. Ozsu, and V. Oria, (2005). **Robust and fast similarity search for moving object trajectories**. In SIGMOD Conference.

[31] Hadi Nasooti, Marzieh Ahmadzadeh, Manijeh Keshtgary and S. Vahid Farrahi (2015), **The Impact of Distance Metrics on K-means Clustering Algorithm Using in Network Intrusion Detection Data**, International Journal of Computer networks and Communication Security, Vol. 3, No. 5, 225-228

[32] E. Keogh and S. Kasetty, (2003). **On the need for time series data mining benchmarks: A survey and empirical demonstration**. Data Mining and Knowledge Discovery 7, 4, 349–371.

[33] E. Keogh, K. Chakrabarti, and M. Pazzani, (2001). **Locally adaptive dimensionality reduction for indexing large time series databases**. In Proceedings of ACM conference on management of data. 151 – 162.

[34] C. BOHM, S. BERCHTOLD, and D. KEIM, (2001). **Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases.** ACM Computing Surveys 33, 3, 322–373.

[35] A.M. Makedonsky (2014), **Problems of data organisation for data mining techniques application in decision support systems**, Microwave & Telecommunication Technology (CriMiCo), 2014 24th International Crimean Conference, Available at: http://ieeexplore.ieee.org/document/6959464/

[36] P. Esling and C. Agon. (2012), **Time Series Data Mining,** ACM Computing Surveys (CSUR) Volume 45 Issue 1, Article No. 12

[37] M. Daigle and K. Goebel, (2012) **Model-based prognostics under Limited Sensing**", IEEE Aerospace Conference, USA, March 2012, pp. 1-12.

[38] Mohammad Sami, Amir M.S Motlagh and Epaminondas Kapetanios, (2015) **Developing Prognostic Models Using Duality Principles for DC-to-DC Converters,** IEEE Transactions on Power Electronics, VOL. 30, No. 5

[39] J.M. Thebaud, E. Woirgard, C. Zardini, C., K. H. Sommer. (2000) **High Power IGBT Modules: Thermal Fatigue Resistance Evaluation of the Solder Joints**, IEEE International Workshop on Integrated Power Packaging, IWIPP. pp. 79-83.

[40] Mohammad Sami, Amir M.S Motlagh and Epaminondas Kapetanios, (2015) **Developing Prognostic Models Using Duality Principles for DC-to-DC Converters,** IEEE Transactions on Power Electronics, VOL. 30, No. 5

[41] M. Samie and A., M. S. Motlagh. (2014), **Realising Duality Principle for Prognostic Models**, The Ninth International Multi-Conference on Computing in the Global Information Technology, At Spain

[42] Peter Zwanziger, (2015). **IGBT Applications in Power Converters,** Research article, Available at: http://www.tandfonline.com/doi/abs/10.1080/09398368.1993.11463322, Accessed on: 08/10/2016

[43] M. Steinbach, L. Ertöz, and V. Kumar, (2003), **The challenges of clustering high dimensional data**, New Directions in Statistical Physics, pp 273-309, Available at: http://www-users.cs.umn.edu/~kumar/papers/high_dim_clustering_19.pdf

[44] M. Ishida, H. Takakura and Y. Okabe, (2011). **High-Performance Intrusion Detection Using OptiGrid Clustering and Grid-Based Labelling**, Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium, Available at: http://ieeexplore.ieee.org/document/6004129/