B. TECH. PROJECT REPORT

On

TIME SERIES CLASSIFICATION IN RESOURCE CONSTRAINED ENVIRONMENTS

By

KRISHNA BHANUSHALI

DISCIPLINE OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

Time-Series Classification in Resource Constrained Environments

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

ELECTRICAL ENGINEERING

Submitted by:

Krishna Bhanushali

Guided by:

Dr. Abhishek Srivastava

(Professor, Computer Science and Engineering)

DISCIPLINE OF ELECTRICAL ENGINEERING

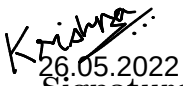
INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"TIME-SERIES CLASSIFICATION IN RESOURCE CONSTRAINED ENVIRONMENTS"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Electrical Engineering' completed under the supervision of **Dr. Abhishek Srivastava, Professor, Computer Science and Engineering, IIT Indore** is an authentic work.

Further, I/we declare that I/we have not submitted this work for the award of any other degree elsewhere.



Krishna Bhanushali

26.05.2022

Signature and name of the student(s) with date

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my/our knowledge.

23.5.2022

Abhishek Srivastava
Professor, CSE

Signature of BTP Guide(s) with dates and their designation

Preface

This project is primarily concerned with finding a scalable algorithm for classifying time series vectors in resource constrained environments. To this effect, various methods for choosing landmark vectors for representation learning Nystrom's method have been studied. A brief introduction to time series data and classification algorithms has been provided in Chapter 1, followed by a literature review in Chapter 2. Chapter 3 provides an in-depth explanation regarding the methods have been used in the proposed methods in Chapter 4. Chapter 5 provides detailed information about the various experiments carried out in order to analyse the proposed methods, followed by the conclusion in Chapter 6.

Krishna Bhanushali
B.Tech. IV Year
Discipline of Electrical Engineering
IIT, Indore

Acknowledgements

I wish to thank Dr. Abhishek Srivastava for his kind support, expertise and valuable guidance. He provided a perfect environment for critical thinking and research acumen and was always available for discussions, doubt clearance and guidance at every part of the project. He has constantly motivated us to take the project to its very culmination. I would also like to thank Mr. Arun Kumar for guiding me through the nitty-gritty of this project. Without their support, this report would not have been possible.

Krishna Bhanushali
B.Tech. IV Year
Discipline of Electrical Engineering
IIT, Indore

Time Series Classification in Resource Constrained Environments

Abstract

The analysis of time series is becoming prevalent across various scientific and engineering disciplines, where the effectiveness and scalability of time series mining techniques depend on the design choices made while representing, indexing and comparing the time series. A lot of the existing algorithms in the field of time series classification can be resource intensive, making it difficult to apply them in an IoT environment, where we have several constraints on resources, with the need to process data being streamed from multiple sensors at the same time. The primary aim of this project is to implement an online time series classification algorithm which can work even in constrained environments.

Keywords: Time Series Vectors, Classification, Clustering

Contents

List of Figures

List of Tables

Chapter 1

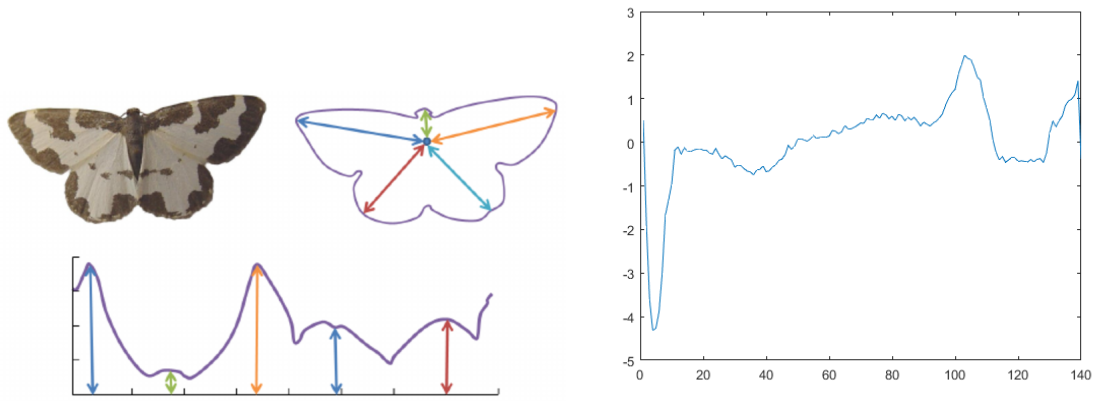
Introduction

Time series consist of a sequential list of some data measured with respect to time of some natural processes like weather patterns, climate, earthquakes or of human processes like speech patterns and biomedical signals etc. The time series vectors may also be composed of multiple signals being measured at the same time, like ECG signals measured using 5 different leads at different position on the body, which may be interrelated and therefore need to be analysed together for accurate analysis. Therefore, time series vectors may be univariate or multivariate depending on the number of variables each time series analyses at the same time.

In addition to signals, many types of data which may not be true time series lend themselves well to analysis using methods primarily used for time series analysis and therefore can be easily translated to time series data. These include, but are not limited to problems consisting of analysis of DNA sequences, textures, core samples, ASCII text, hand-writings and even shapes[11]. The Figure 1.1 shows a few examples of time series vectors.

Due to the periodicity of such measurements, time series vectors lend themselves effectively to shape based comparison methods, as in most cases, the visualisation of time series will be able to accurately define the trends in data and any anomalies present. For example, visual inspection of ECG signals and timing measurements are the primary methods with which diagnosis is carried out, while for Earthquakes, rapid and high amplitude spikes would indicate earthquakes.

With the recent advances in data collection methods, time series mining methods are now capable of collecting enormous amounts of data using scientific instruments including neuroscience[9] and astronomy[2], and in industrial settings([19], [3]). Now, with the explosion of IoT devices, the number of large time series vectors,



(a) Converting the Shape of a Moth to a time series vector as in [7]

(b) ECG as a time series

Figure 1.1: Examples of Time Series Vectors

with sizes in millions of data points, have increased, and are expected to continue to massively increase.

This is therefore a challenge for existing time series classification algorithms for which scaling to the massive levels of time series lengths is an arduous task. Depending on the start time and additional noise introduced during measurement, temporal ordering of the time series combined with the high dimensionality also introduce challenges which are not seen in other types of data. These temporal distortions particularly affect similarity measurement methods, as ideal methods would compare the shapes of the time series methods while allowing for invariances to distortions in amplitude, phase and timing. The algorithms which would address these issues therefore need to be able to a) use low dimensional representations of the data to preserve time series characteristics, b) compare time series data while providing necessary invariances and c) index the data to enable fast querying of large time series vectors from huge databases.

In order to address these issues, [23] introduce a two step approach for a framework for querying, visualisation, classification and clustering of time series datasets. They rely on representing the data in terms of a few landmark vectors, chosen using k-shape[24] clustering, using the Nystrom's method and by further approximating the representations using a matrix sketching algorithm[16]. This method is very accurate and fast, with a small time and space complexity, making it a good choice when implementing time series algorithms on the Edge. However, using k-Shape to find the landmark sequences in an online setting requires repeating multiple calculations for refining the cluster centroids at each step, which may lead to inefficiencies

while classifying time series vectors in an online setting.

This project therefore aims to analyse different methods to select the landmark vectors required by Nystrom's decomposition which may lead themselves well to classification in online settings On the Edge.

Chapter 2

Related Works

[6] compare various available algorithms for classifying time series datasets and conclude that 1NN-DTW is the best method to classify datasets, however, due to high computational requirements for repeated calculations of DTW, the method is not suitable for datasets with a large number of time series vectors of high dimensionality. Therefore, multiple methods of representing time series datasets using fewer dimensions have been studied.

Kernel based methods which map the input to a high dimensional feature map for an improved bias-variance tradeoff by calculating the Gram Matrix, where each entry is a measure of similarity of pairwise input vectors have been very popular for classification problems. However, due to the inherent nature of these kernels, they generally require $O(n^3)$ in time complexity, thus making the use of approximation methods necessary. Over time, multiple algorithms for this approximation have been suggested, such as Incomplete Cholesky decomposition [4], Nystrom decomposition[35], KPCA [33] and Random Fourier Features[37]. But as Cholesky decomposition needs the entire Gram Matrix before calculations and as the rank to which we can decompose the matrix is much smaller than that for Nystrom decomposition, Nystrom decomposition is generally preferred. Random Fourier Features approximates the gram matrix sampling the input using a randomised map to a lower dimensional vector (ideally the resulting vector has a much smaller dimension). As the quality of the approximations generated by the Nystrom's method is directly dependant on the quality of the feature vectors, [13] suggest a greedy algorithm while [20] use a recursive method to identify the feature vectors iteratively, by operating on the entire Gram Matrix.[38], on the other hand, select the landmark feature vectors using k-means clustering instead.

According to [17], even though exact linear and nonlinear dimensionality reduction methods are highly accurate, they have a tradeoff, that is, they are extremely costly in time and space complexity. Therefore, in [17], representation methods reliant on spectral decomposition have been used to project the high dimensional time series data to fewer dimensions in order to lower the time and space complexity of time series analysis algorithms. Some other time series represent methods represent time series datasets using a set of sinusoidal coefficients (DFT) in [1] , as a set of Wavelets (DWT) in [26], and using Chebychev polynomials[10], without using any knowledge specific to the data being used. Additionally, data aware methods have been used to augment these methods to improve the effectiveness and rely on spectral decomposition for the same.

The field of clustering time series data is vast, however much of the works in time series clustering calculate clusters based on features extracted from the time series. The shape based clustering algorithms for time series data however fall into one of two categories. Algorithms like [24] tries to include all the objects inside clusters , while those like [29] can leave some of the anomalous unclustered.

Chapter 3

Background Theory

3.1 Nystrom's Method

Nystrom's method, introduced in [35], is a low rank matrix decomposition method which can approximate a nxn symmetric positive semi-definite kernel matrix using a nxl matrix, by choosing l landmark sequences. Additionally, it can also calculate a kxn representation of the objects present in the dataset mapped to k dimensions.

For a Gram Matrix G, which can be represented as:

$$G = \begin{bmatrix} W & E \\ E^T & X \end{bmatrix} \tag{3.1}$$

where, W is the landmark-landmark similarity matrix and E is the landmark-dataset similarity matrix as:

$$G = \begin{bmatrix} W & E \end{bmatrix}^T \tag{3.2}$$

Now, in order to calculate the representation of the dataset, Nystrom's method carries out SVD to represent W^{-1} as :

$$W^{-1} = Q_W \Lambda_W^{-1} Q_W^{-1} \tag{3.3}$$

And therefore, the approximated value of the kernel matrix can be written as:

$$\bar{K} = (EQ_W \Lambda_W^{-1/2})(EQ_W \Lambda_W^{-1/2})^T = Z_d Z_d^T \tag{3.4}$$

$Z_d = EQ_W \Lambda_W^{-1/2}$ can then be used to represent the entire dataset in terms of their similarity with k landmark vectors. Theredore, in order to calculate the

Nystrom's approximation of some Gram Matrix, we need not calculate the entire Gram Matrix, given that the landmark vectors are available. Given the landmark vectors have already been chosen, Nystrom's method requires a time complexity of the order of $O(l^3 + nl^2)$, and therefore, as $l << n$ landmark sequences can be chosen for approximation, Nystrom's method is linear in the number of time series chosen.

The accuracy of the representations generated by Nystrom's method, however, depends a lot on the quality of landmark sequences used for calculating the projection. Therefore, methods belonging to mainly three main classes, namely, randomised methods, clustering based methods and deterministic methods have been studied by researchers to identify the landmark sequences.

Randomised methods select the landmark sequences from the dataset randomly, therefore allowing for faster calculations. Methods which take into account the skewness of the dataset and other parameters dependant on the nature of the dataset which may make some type of landmark sequence more accurate have been studied, but among the randomised methods, selecting the landmark sequences using a uniformly random distribution has been shown to be most effective in terms of accuracy and robustness among the randomised methods. Clustering based methods calculate the clusters of the objects present in the dataset and use the centroids of these clusters as the landmark sequences. The accuracy of these methods, therefore, heavily depends on the quality of clusters and centroids formed by the clustering algorithms, but the randomisation is avoided, trading some of the time complexity for robustness. Deterministic methods however, primarily operate on the entire Gram matrix to iteratively calculate the Nystrom's representation while choosing the best landmark sequence available at each step. These methods are extremely accurate, however, due to operating on the entire Gram matrix, are unusable in cases where the calculation of the entre Gram matrix is costly, such as in time series datasets, where distance and similarity measures can be particularly slow.

Therefore, here, the methods of selecting the landmark sequences for Nystrom's method using a uniform random distribution and Clustering based approaches have been studied.

3.2 Distance and Similarity Measures

Distance and Similarity measures used to obtain the Gram matrix (and by extension, some submatrix of the Gram Matrix) are of course essential components of repre-

sentation learning and classification methods using kernel methods, even in the case of time series vectors. These measures also form crucial components of clustering methods, where clusters are defined based on distances between the objects present in each cluster. For time-series datasets, due to variations while measuring the data, and due to noise, members of the same dataset may show slight distortions and deviations. Therefore, in order to properly accomodate such deviations, a good distance measure must provide some degree of invariance to shift and warping.

3.2.1 Euclidean Distance

Euclidean Distance or the $L2$ Norm is measured between two vectors of equal length by measuring the square root of the sum of the distance between each entry of the vector. Therefore the ED between vectors $\vec{x}$ and $\vec{y}$ is given as:

$$ED(\vec{x}, \vec{y}) = \sqrt{\sum (x_i - y_i)^2} \tag{3.5}$$

Here, as the amplitude of the time series at each point are compared to each other directly, only $O(n)$ time is needed to compute this distance and since it is a metric distance, the triangle inequality can be used to optimise algorithms where pairwise distances are repeatedly used, such as clustering algorithms like k-means. However, in the case of time series data, as we Euclidean distance does not provide any measure of shift invariance, ED does not provide a good accuracy and is therefore avoided as a distance measure, atleast for small datasets.

3.2.2 Shape Based Distance

Shape Based Distance, introduced in [24], is a distance measure which computes the distances between all possible shifts of data by means of cross-correlation and then finds the amount of warping present in the two time series. this is done be using Fourier transforms as the convolution function, also known as the cross-correlation function, slides the second vector over the first after padding both the vectors with zeros and reports the distance at each position. Therefore, the Shape Based Distance is given as:

$$SBD(\vec{x}, \vec{y}) = 1 - max(\frac{F^{-1}(F(x) * F(y))}{\|x\|\|y\|}) \tag{3.6}$$

The Discrete Fourier Transforms and Inverse Fourier Transforms need an overall complexity of $O(n^2)$ where n is the length of the sequence. However, Fast Fourier

Transforms and Inverse Fast Fourier Transforms are able to approximate DFT and IDFT in $O(nlogn)$ [18][22] and are therefore used while calculating the SBD. SBD is a metrix measure, however, as it does not satisfy the triangular inequality, it is not a metric distance and therefore does not enjoy any optimisations which may be available while repeatedly calculating metric distances.

3.2.3 DTW

Dynamic Time Warping measures the warping between two time series by measuring the edit distance between the two time series and stating that the length of the minimum warping path is the distance determines the distance between the two time series. This is done by first constructing a m x m matrix where:

$$M_{i,j} = \sqrt{(x_i - y_j)^2} \tag{3.7}$$

and then calculating the warping path $L = \{l_1, l_2, ..., l_n\}$ with the minimum warping path giving distance:

$$DTW(\vec{x}, \vec{y}) = \sqrt{min(\sum(l_i))} \tag{3.8}$$

which can easily be calculated for two time series using Dynamic Programming in $O(m^2)$. The matrix M can be visualised as in Figure 3.1, where we can easily see the warping path. This path was calulated using the recurrence relation:

$$D_{i,j} = dist(x_i, y_j) + min(D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}) \tag{3.9}$$

Where $D_{m,n}$gives the DTW for two time series $\vec{x}$ and $\vec{y}$ of lengths m and n respectively. This version of DTW is called the Full DTW and because of its quadratic time complexity, algorithms such as constrained DTW and fast DTW have been developed. This algorithm takes $O(n^2)$ in time and space when a suitable dynamic programming algorithm is used. However, as this algorithm requires only the current and previous row, the space complexity can be reduced to $O(m)$ and therefore, the constraint on the efficiency of DTW is defined by the processing speed of the CPU rather than Memory. Constrained DTW speeds up DTW by limiting the region where a warping path can be present to either a band (Sakoe-Chiba band [31]) or a parallelogram (Itakura Parallelogram)[[15]] along the diagonal with a fixed radius or width, which is determined in percentage of the length of time series. Fast DTW [32] tries to approximate the full DTW by aggregating it into small pieces and then refining those pieces. However, cDTW is faster and more accurate than fast DTW

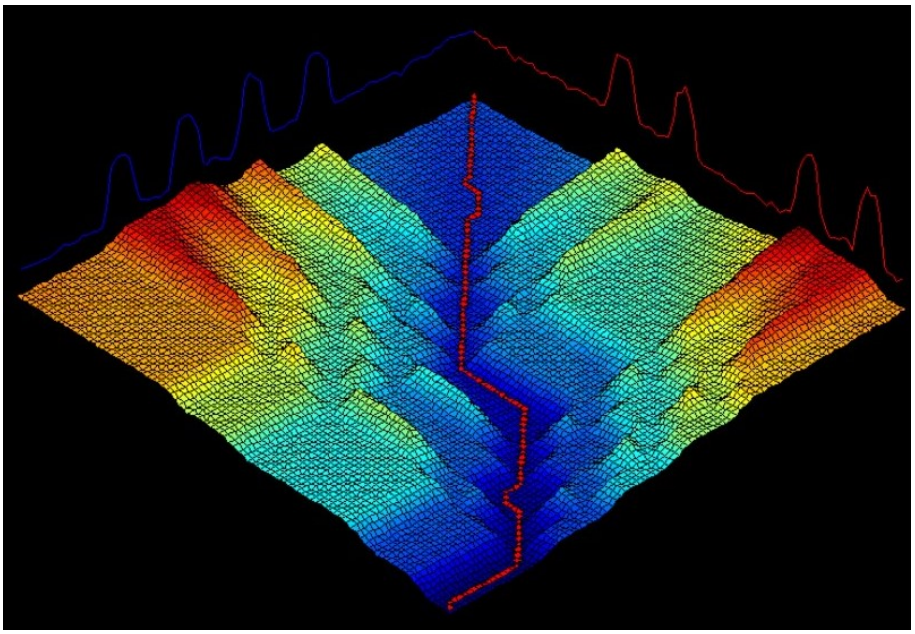


Figure 3.1: Visualisation of the path chosen by Dynamic Time Warping for two given time series

for classification problems as shown by [36]. cDTW however introduces an additional parameter w which needs to be tuned. For classifiers, w is generally tuned using LeaveOneOut methods, increasing the complexity and time required during training models. It can also be seen that at w=0, cDTW becomes the Eucliedan Distance, and that at w=1, cDTW represents full DTW, as the entire matrix M would be traversed here.

Additionally, incase the start and end points are not exactly matched, DTW may not give accurate classification as the warping for these points is not calulated by the current algorithm. However, this can be resolved by either smoothing the dataset or by relaxing the start and end point conditions for the warping path to allow it to start and end at any point. In order to optimise indexing and classification by DTW, multiple lower bounding methods have been developed in order to prune most of the calculations.

3.3 Clustering Methods

In order to identify the landmark sequences for Nystrom's method, the centers of clusters inside the dataset can be considered to be representative of the clusters. These can therefore act as landmark sequences effectively. For clustering of time series data, TADPole [8], k-Shape [24] and Dynamic Time Warping Barycenter

Averaging(DBA) [25] have been discussed in subsequent subsections.

3.3.1 TADPole

TADPole, introduced in [8], is a clustering algorithm which clusters a given set of timeseries data using the Anytime DP algorithm [29] while using a pruning strategy using both upper and lower bounds on DTW to speedup the clustering process and following a 'most useful first strategy' to further reduce the required computation time.

In [27], the authors show that the ability to ignore certain timeseries from the dataset is extremely important while clustering time series data because this enables the algorithm to avoid anomalous objects which are by themselves unclusterable and also affect the labels of objects which can be clustered by affecting their distances from other objects, which can be done using the Density Peaks algorithm as clusters objects on the assumption that the clusters may form arbitrary shapes.

The Density Peaks algorithm needs the distance between all pairs of objects in the dataset in order to calculate the local densities at each point. However, as this would be prohibitively expensive for very large time series datasets, TADPole takes advantage of the fact that some of the calculations may be pruned by taking advantage of Lower Bounds on DTW like Keogh LB [30] as well as the fact that Euclidean Distance is an Upper Bound for DTW, which can both be calculated in $O(n)$ time where n is the length of time series.

The TADPole algorithm takes the pairwise distance matrices made with upper and lower bounds, i.e. the UBMatrix and the LBMatrix, in check when cDTW needs to be calculated during the density peaks algorithm. However, for very large datasets, even this level of pruning may be inefficient and therefore, the algorithm is cast on an Anytime DP framework. The UBMatrices and LBMatrices introduce an additional space complexity of $O(n^2)$ where n is the size of the dataset, however, as DTW is primarly constrained by speed and not memory, this memory usage is allowable. Additionally, the calculation of these matrices take less than 1% of the time required for the entire clustering operation and are therefore inconsequential. Additionally, in order to prune the DTW calculations, TADPole uses a cutoff distance (d_c) hyperparameter, and requires the number of clusters to be formed (k).

Given the required inputs, TADPole first calculates the local densities (ρ) for each object in the dataset, which refers to the number of objects belonging to the dataset

which are within a distance of d_c from the i^{th} element. Here, they consider four cases, namely, case 1: both the objects are identical, i.e. LB and UB will be equal and therefore, DTW has to be equal to the UB case 2: $UB \leq d_c$. i.e. the objects are in the locality of each other and therefore the distance between them can be safely considered to be d_c without loss of any accuracy, case 3: $LB \geq d_c$, i.e. the objects are definitely too far away from each other and case 4: When none of the above conditions are true, we need to calculate the cDTW between the two arrays, which remains the only case where we actally need to calculate cDTW. Now, depending on the case each comparison belongs to, two matrices, namely SparesFlagMatrix and SparseDistMatrix are maintained, which contain the distance and the case for each of the pairwise comparisons. Counting the number of elements in the local vicinity of each object gives us the value of the local density ρ.

Then in order to calculate the distance from points of higher densityδ, required to find the centers, for each item we need the distance to the nearest neighbours in the sorted list of ρ. However, this again requires us to calculate cDTW multiple times and therefore, UBMatrix and LBMatrix are again used to quantify this distance. Therefore, an Upper Bound for this nearest distance (ub) is calculated to facilitate pruning. Here, for each item, we have either calculated the actual distance $D_{i,j}$ or the Upper Bound. Therefore we can scan all ρ_i's with density greater than that of i and set the upperbound for the nearest neighbour ub_i to either $D_{i,j}$ or $UB_{i,j}$ depending on which is available, while keeping the maximum. The ub matrix is then used to optimise the pruning during calculation for distance from the nearest neighbour.

Now to calculate the distance from the nearest neighbour, we only need to find the element with the smallest distance from the higher density objects list. This needs to be calculated using cDTW only when $LB_{i,j} \leq ub_i$ i.e. the two objects are closer than the maximum possible distance the nearest neighbour can have, and when the distance $D_{i,j}$ is not known. This effectively prunes the number of cDTW calculations which need to be made, and gives us the value of δ.

Now, as per [29], the cluster centers will be the objects with the top k values of $\sigma_i\delta_i$, where k is defined as the number of classes present in the dataset as we need to choose k landmark sequences for Nystrom's method, and we choose the k cluster centers as the landmark sequences.

3.3.2 k-Shape

k-Shape, introduced in [24] is a partition based method which, in a fashion similar to k-means, refines the clusters iteratively, while minimising the sum of squared distances for elements of clusters to produce well-seperated and uniform clusters. As for each iteration, each object is evaluated only once, k-Shape is linear in terms of the number of time series and is therefore scalable.

K-Shape is a to stage method, where for every iteration, it first assigns each time series to the closest centroid to update the cluster it belongs to and then updates the cluster centroids to better represent the new clusters. This process is repeated until either a set number of iterations have been reached or until the algorithm converges, i.e. no changes in cluster assignment can be observed in subsequent iterations.

The initial seed of cluster centers is randomly assigned and the in-cluster distances are measured using SBD, to provide the required invariances. The clusters centers, however, need to be calculated again, and these are calculated by finding a vector $\vec{\mu_k}$ which maximizes the sum of $NCC(\vec{x}, \vec{\mu_k})$ for all $\vec{x}$ belonging to the cluster. Here, all the sequences are alligned with the previous center in order to calculate X', which is then used to calculate $S = X'^T.X'$ and after z-normalisation, the first eigenvenctor of S' is chosen as the new cluster center.

k-Shape has a time complexity of $O(max(nkmlog(m), nm^2, km^3))$ per iteration as $O(nkmlog(m))$ is required for assigning each vector to the nearest cluster, $O(nm^2)$ is required for normalisation and $O(nm^3)$ is required for eigenvalue decomposition to find the first eigenvector. k-Shape also provides a benefit of being invariant to start and end point alignments of time series vectors, by virtue of SBD being used as the distance measure.

3.3.3 DTW Barycenter Averaging

DTW Barycenter Averaging as introduced in [25] is another clustering algorithm which performs clustering by initialising with a random set of cluster centroids chosen from the dataset and then iteratively refining the clusters by assigning each vector to the nearest cluster and then updating the cluster center.

The primary difference between k-Shape and DBA is the method with which the cluster centroids are updated. Instead of using SBD, DBA calculates the warping path used by DTW and then claims that the average sequence is actually the average

of the elements the $i^(th)$ element of the warping path obtained while calculating DTW maps to in both the series, known as the barycenter.

This, therefore, provides a very accurate, but slow method to obtain the centroids at each step, as each iteration requires $O(n^3)$ in time to calculate the Barycenter, by virtue of requiring the DTW path. Additionally, there is no scope for pruning the DTW comparisons as the warping path requires the calculation of DTW. This algorithm, can however still be used along with cDTW allowing a higher accuracy while forming the clusters.

3.4 GRAIL

GRAIL, introduced in [23], is a framework which intends to provide a unified approach for fast and accurate analysis of large time series datasets. Given a comparison function, GRAIL extracts landmark time series, optimizes necessary parameters and uses approximation methods based on kernel methods to represent each time series as a combination of landmark time series in linear time and space with an aim to: preserve pairwise similarities and serve as feature vectors for machine learning methods; lowerbound existing comparison functions; allow prefixes for scaling methods under limited resources; support efficient computation of new data to enable operations in online settings; and support eigendecomposition of the data to data similarity matrix to exploit highly effective kernel based methods. given a comparison function, the learned representations

In order to learn representations of time series in linear time and space, GRAIL needs to first approximate the sequence to sequence similarity matrix and then approximate its eigendecomposition. As Nystrom's method is agnostic to the choice of kernel function, Nystrom's method is used to approxmiate the sequence to sequence similarity matrix. Now, as Nystrom's method requires a dictionary of landmark sequences in order to approximate the Gram Matrix, GRAIL selects the landmark sequences using k-Shape and then calculates the dictionary to dictionary and dictionary to sequence similarity matrices to then compute the representation. The representations learned using Nystrom's method may have a higher dimensionalty than the original vectors and may not be the optimum representations in case the necessary parameters are not accurately estimated. GRAIL provides a methodology to learn these parameters in an unsupervised manner along with using these representations to learn the final representations which would have the required dimensionality.

Shift Invariant comparison is essential in time series comparisons to account for timing delays and warping which may occur while measuring, storing and transmitting real world signals, and therefore similarity functions like DTW, SBD and NCC have been considered for developing kernels. However, DTW and SBD do not lead to positive semidefinite kernels [5] which leads to convex solutions for many kernel based learning problems. In order to address this problem, GRAIL introduces SINK, a shift invariant kernel based on NCC.

In order to obtain a p.s.d. kernel, and to consider all possible shifts in the time series, SINK considers all possible shifts and in a process similar to SBD, calculates the kernel as:

$$k_s(\vec{x}, \vec{y}, \gamma) = \sum e^{\gamma NCC(\vec{x}, \vec{y})} \tag{3.10}$$

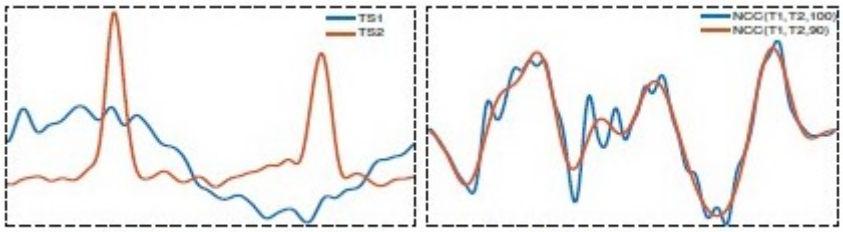
$$NCC(\vec{x}, \vec{y}) = \frac{F^{-1}(F(x) * F(y))}{\|x\|\|y\|} \tag{3.11}$$

This might however lead to a Gram matrix with off diagonal values higher than values on the diagonal (i.e., A vector is more similar to some other vector than itself). And therefore, SINK can be normalised as:

$$k'_s(\vec{x}, \vec{y}, \gamma) = \frac{k_s(\vec{x}, \vec{y}, \gamma)}{\sqrt{k_s(\vec{x}, \vec{x}, \gamma) k_s(\vec{y}, \vec{y}, \gamma)}} \tag{3.12}$$

Here, we can observe that SINK requires operations in Frequency space, and therefore calculates the Fourier Transform using the Fast Fourier Transform Algorithm at every step. Now, in order to speed up calculations, we can calculate and save the Fourier representation of each time series before hand, so that the cross correlation operation can be calculated efficiently. This is then followed by an inverse FFT back to the time domain. Now, as we are considering only the frequency representation of time series, an efficient method of storing these representations and eventually speeding up the following calculations would be to save the first few fourier coefficients. This is possible because most of the natural time series vectors have a skewed energy spectrum, where the first few frequencies contain most of the signal's energy, and therefore, determine most of its shape. This also helps by somewhat denoising the time series.

Therefore, as the FFT step, and consequently, the IFFT steps while calculating SINK determine its time complexity, SINK has a time complexity of O(mlogm), where m is the length of the time series vector. SINK also satisfies the p.s.d. property



(a) Time series $T1$ and $T2$ (b) NCC sequences

Figure 3.2: NCC Sequences produced by SINK for time-series vectors T1 and T2 while preserving 90% of the energy

as it computes p.s.d. kernels computed for each possible alignment between two time series. Additionally, using Figure 3.2, [23] show that a time series of length 1024 can be represented using a length of 21, while accurately calculating the NCC, by preserving only 90% of the signal's energy.

Once GRAIL has learned the dictionary by means of clustering using k-shape, the parameter γ still needs to be estimated as it can affect the compactness and accuracy of the learned representations In [23], the authors observe that, time series similarities with larger variance are much more likely to be present in the low dimensional space and that the loss of information is unavoidable due to the projection to a low-dimensional space. Therefore, the values of the bandwidth γ need to be chosen such that this variance is maximised. They also state that we need to measure the variance each eigenvalue explains, so that we can select a small number of r eigenvalues to effectively determine the bandwidth.

Therefore, considering the eigenvalue decomposition of the matrix $W = Q_W \Lambda_W Q_W^T$ the scoring function is calculated as:

$$Score(\gamma) = var(W, \gamma) \frac{\sum_{i=1}^{r} \Lambda_W(i)}{\sum_{i=1}^{d} \Lambda_W(i)} \tag{3.13}$$

Where $var(W, \gamma)$ denotes the value of W calculated using a SINK with bandwidth γ. Now, as the algorithm depends on the computation of the eigenvalue decomposition of W, it has an overall complexity of $O(d^3)$.

Now, as SINK can be calculated and as the dictionary is known, GRAIL constructs the temporal representation Z_d where d is the number of landmark sequences, to approximate the Gram Matrix as $Z_d Z_d^T$ using Nystrom's method and then uses Z_d to learn the final representation Z_k where, $k \leq d$.

$$W^{-1} = Q_W \Lambda_W^{-1} Q_W^{-1} \tag{3.14}$$

$$Z_d = EQ_W \Lambda_W^{-1/2} \tag{3.15}$$

Where E contains pairwise similarities between n time series and d landmark time series, and where d is significantly smaller than the size of the dataset. However, for very large datasets, a large number of time series need to be included, which may lead to a dimensionality much higher than the length of the original time series and therefore, further approximation is carried out by choosing only the top k eigenvalues and eigenvectors of k. However, as the approach requires $O(nd^2)$time complexity, this becomes prohibitively expensive for a large number of landmark sequences. Therefore a matrix sketching algorithm named Frequent Directions [16] is applied on Z_d instead, which retains a sketch of size b of Z_d in $O(ndb)$ instead, to construct the representation of the input data as:

$$Z_k = Z_d Q_{C(1:d,1:k)} \tag{3.16}$$

3.5 Statistical Methods

In this project, we use multiple datasets as inputs for different algorithms and we therefore want to compare the overall performance of the different algorithms on all the datasets. Therefore, we need to use statistical methods to rank the efficacy of different algorithms, including tests like Wilcoxon signed rank test[34], Friedman test[14] and the Nemenyi test[21], following [24],[6]and [12].

The Wilcoxon signed rank test is a test which tests whether two sets of pairwise data come from the same distribution. Therefore, with regards to the accuracies of various datasets, the test explains whether the difference in performance of the two algorithms being compared is statistically significant.

The Friedman test is generally used to test whether multiple experiments on the same data give consistent results. Therefore, for classification on multiple datasets, it can be used to check whether the difference in accuracies over multiple datasets is statistically significant. Successful results from the Friedman test are generally followed by a post-hoc Nemenyi test which calculates pairwise computations to finally give a ranking for the algorithms with performance over all the datasets in mind.

Chapter 4

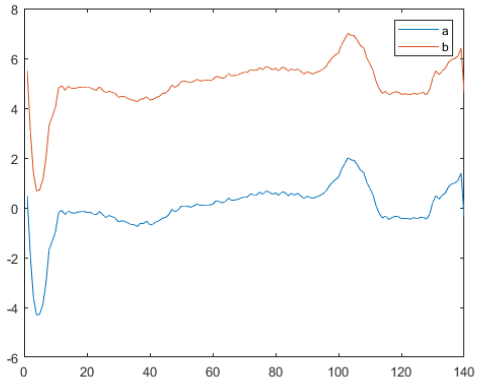
Methodology and Data generation

4.1 Datasets

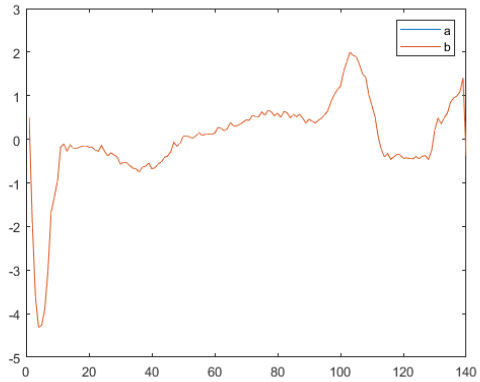
The primary aim of this project is to find a suitable replacement to the algorithm responsible for finding the landmark sequences used while learning representations using the Nystrom's method in the GRAIL framework. Therefore, as authors of GRAIL evaluated their algorithm using the UCR Archive, All experiments in this project were carried out on 83 datasets from the UCR archive[11]. The UCR Archive is a collection of univariate time series datasets, belonging to multiple domains and consisting of both simulated and measured data. These include domains like images, spectral data, simulated data, sensor measurements, motion capture data, ECG signals, traffic, device usage data, trajectories and Power consumption data. As the data in the UCR Archive is already separated into train and test sets, the same division of test-train data was used for analysis.

4.2 Preprocessing

As we are primarily concerned with classification of time series datasets based on their shapes, we need to account for offsets in the measurements. This is primarily because a slight offset due to consitions in the wild do not directly change the shape, however, they can directly affect the similarity measurements, as these directly change the value of $(x_i - y_j)^2$ in the case when $\vec{y}$ has some offset. For example, when comparing $vecx$ and $vecy$, where $\vec{y} = \vec{x} + c$, the shape of both the vectors is the same and therefore, we expect 0 as the DTW distance between them, to ensure



(a) Two identical vectors offset by constant value, DTW = 58.75



(b) Normalised vectors, DTW = 0

Figure 4.1: Comparison of DTW distance for two identical vectors with offset with and without Z-score Normalisation

ideal results during classification. However, due to the offset c, the distance of c^2m is obtained, where m is the length of x, which will lead to errors in similarity calculation and distance measurement, a real world example is shown in Figure 4.1. Therefore, before analysis the data from the UCR Archive needs to be preprocessed. This is generally done by applying z-score normalization the data to have a mean value of 1 and a standard deviation of 0, to preserve the shape while getting rid of the offset, using the equation:

$$x_i^{normalised} = \frac{x_i - \mu(\vec{x})}{\sigma(\vec{x})} \tag{4.1}$$

where μ and σ are the mean and standard deviation respectively.

4.3 Proposed Methods

We propose Uniform Random Selection, DTW-DBA and TADPole as suitable candidates for selection of landmark vectors for approximating the representations learned using the Nystrom's Method. Random Selection was chosen primarily for the minimal computation time required for landmark selection, DTW-DBA was chosen primarily due to its high Rand Index[28] compared to k-Shape and to observe the benefits to accuracy due to the use of DTW-based algorithms for clustering. TADPole, which also uses DTW to cluster the data, was suggested because of the accuracy observed from DBA, as it addresses the problem of the large computational time requirement by pruning the number of DTW comparisons. Therefore, for each

method, following [23], we calculate the landmark-landmark similarity matrix, approximate γ, calculate the landmark-dataset matrix, and obtain representations for the time series data, followed by further approximation using Frequent Directions.

A standard method of classification of the learned representation is required in order to determine the quality of the representations generated by using the landmark selection method for Nystrom's method. For this the 1-Nearest Neighbour classification algorithm was used with Euclidean Distance measured between the learned representations as the distance measure. 1NN was selected as it is a fast and robust algorithm which does not require any sort of tuning without any additional training time, and classifies the data in $O(n^2)$ time complexity. Additionally, 1NN-DTW and Elastic Ensemble, which is a combination of multiple NN-DTW classifiers, are also regarded as State of the art algorithms[6] in terms of accuracy for time series vectors, further reinforcing the choice of 1NN as the classification algorithm. 1NN-ED assigns labels to objects in the test dataset, by choosing the label of its nearest neighbour in the training set. k-NN is more accurate modification of 1NN which chooses the class labels based on the median class from its k Nearest Neighbours, however, as k needs to be tuned here, 1NN was chosen as the classification algorithm.

As the number of landmark sequences selected also defines the classification accuracies, to be fair to all methods of selection, the same number of landmark sequences need to be chosen for each method. Following [23], assuming that each class would form a single cluster and that it can therefore be represented using a centroid of that cluster, the number of landmark features was chosen to be equal to the number of classes present in the dataset.

Chapter 5

Experimental Results

In the chapter, various experiments on the usage of different approaches to selecting landmark vectors for time series representation learning using Nystrom's method and further classification based on the GRAIL framework and their results have been discussed, along with scatter plots comparing the accuracies of the methods. Additionally, the datasets with a significant amount of difference in accuracies have been presented for additional discussion, along with discussion on processing times of these algorithms.

5.1 Experiments

5.1.1 Comparison of Random Selection and k-Shape

In order to analyse randomised methods for selection of landmark sequences, as suggested in the original paper on Nystrom's algorithm, the landmark sequences were selected using a uniform random distribution from the dataset, after z-score normalisation. Then, the value of the parameter γ was estimated to maximize variance in W. This value of γ was then used to calculate the matrices W and E, following which the exact representations(Z_d) of the dataset were obtained using Nystrom's method for both test and train datasets together. Using frequent directions, the approximate representations were obtained, which were then classified using 1NN to obtain classification accuracies. A similar method was followed to obtain the classification accuracy using k-shape clustering as the landmark selection method. In order to account for fluctuations in accuracy due to the randomised approach, both worst case and average case performance for random selection were studied by

repeating the entire process $2k$ times, where k is the number of classes present in the dataset. The accuracies were then plotted as cscatter plots compared with those obtained using k-shape as a landmark selection algorithm, followed by Wilcoxon test with a confidence of 99% to test for statistical significance in the differences in performance. In order to provide a fair comparison in terms of processing time, both the algorithms were implemented in python as per [24], in a CPU environment provided by Google Colab.

5.1.2 Comparison of DTW Barycenter Averaging and k-Shape

Then, k-shape and DTW Barycenter Averaging (DBA), a k-means based clustering method which calculates the centroids using Barycenters, were compared by following a similar procedure, while using the centroids as landmark sequences. As DBA primarily uses full DTW for clustering, no additional hyperparameters were required. However, a more fast and accurate variant of DBA using cDTW is also available, which requires the additional parameter of band width for the Sakoe-Chiba band, which was chosen as 10% for all datasets. The implementation of DBA was based on [25]'s implementation. The accuracies for k-shape vs DTW-DBA and for k-shape vs cDTW-DBA were both plotted as scatter plots, followed by the Wilcoxon test with a confidence of 99%. Both the algorthms were implemented in a CPU environment provided by Google Colab.

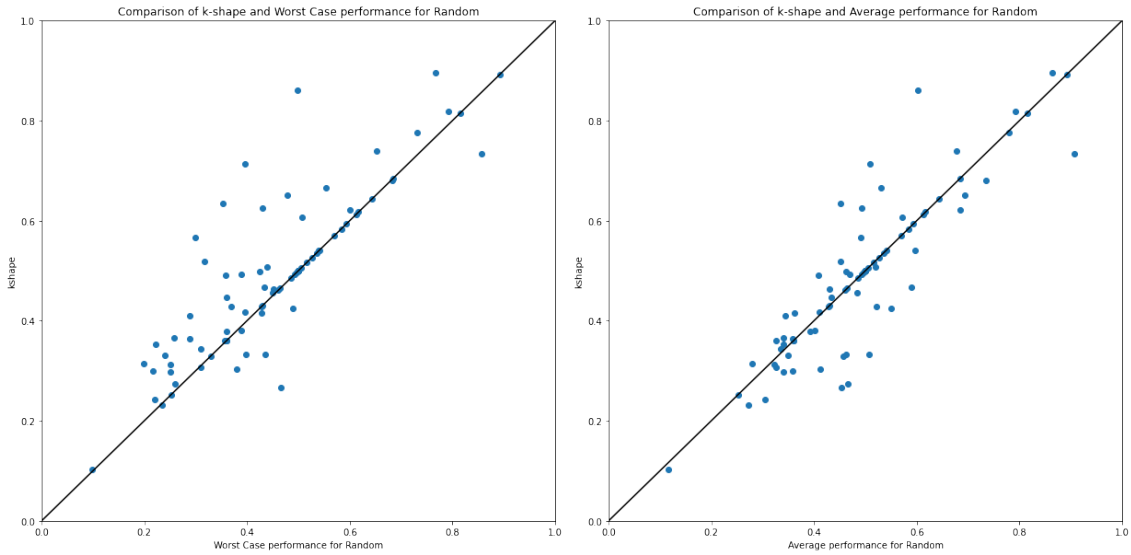
5.1.3 Comparison of TADPole and k-Shape

TADPole, which is a density peaks based clustering algorithm which selects the centers of clusters based on the density and distance of objects in its vicinity, was then compared with k-shape for applications in selection of landmark sequences for Nystrom's method. TADPole, however, requires two hyperparameters, the cut-off distanced_c and the cDTW band width r. Here, for all datasets, the value of d_c was chosen as 1.46 and in order to understand the effect of changing r on accuracy, $r = 5$ and $r = 10$ were used. Additionally, the value of γ was set as 20 for both TADPole and k-Shape. When DTW is used on sequences where the start and end are not alligned, the accuracy has been observed to be lower than expected. As smoothing the data has been suggested for such use cases, the quality of representations obtained using both smoothed and raw data were compared and finally, the best case for both was compared with k-shape to obtain a useful comparison, followed by the

Wilcoxon test with a confidence of 99%. Both the algorithms were implemented in MATLAB and run using the same installation for providing a meaningful runtime comparison.

5.2 Results

5.2.1 Comparison of Random Selection and k-Shape



(a) Worst Case

(b) Average case

Figure 5.1: Comparison of k-shape and random sampling for landmark selection

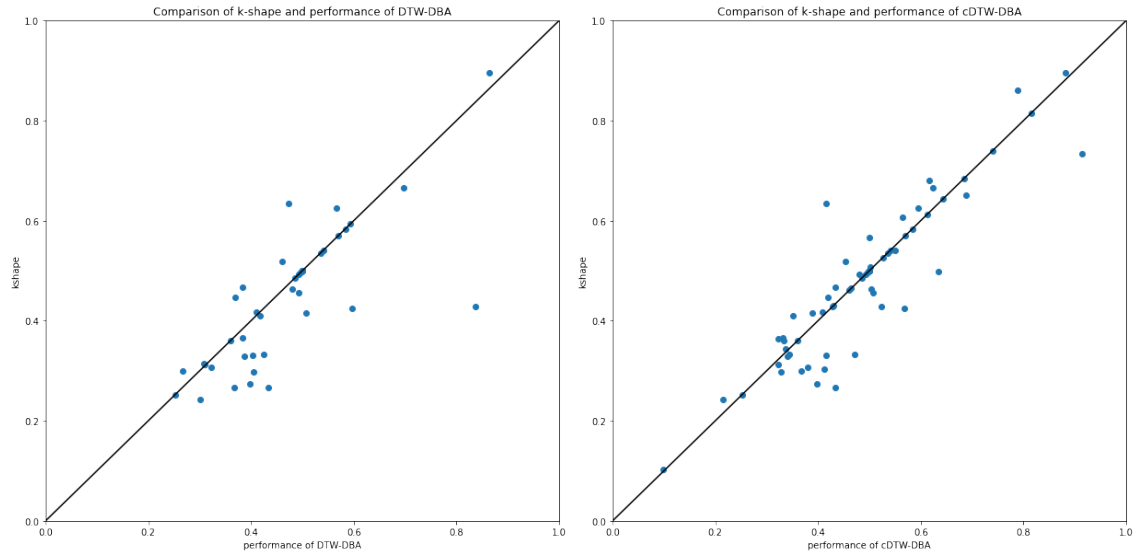
Method	< k-Shape	= k-Shape	> k-Shape
Random (Worst Case)	39	11	33
Random (Average Case)	25	27	31

Table 5.1: Performance of random selection in worst and average case compared to k-shape for landmark selection

When comparing k-shape and random selection for Nystrom's method, it was observed that k-shape outperformed the worst case of the random selection algorithm according to Figure 5.1. However, k-shape performed better on less than half the datasets as per 5.1 as compared to the average case performance of random selection. Wilcoxon test further shows that the difference in performance is significant and therefore, as the mean for the average case of random selection is higher than that of k-shape, we conclude that random selection outperforms k-shape in the average

case. Additionally, random selection is much faster in terms of processing time as the clustering operation is completely avoided. However, due to the disparity between average and minimum performance we can say that despite the better performance in average case, random selection fails to improve upon k-shape as in the wild, there is no way to check whether the current performance of the algorithm is optimal without repetition, and therefore random selection will have a lower precision, making it unsuitable for use in an online setting. k-Shape out performing the worst case of random selection, however, is expected as the best case for this scenario would be selecting completely dissimilar time series, while the worst case would be selecting the same time series. Therefore, we expect k-shape, which tries its best to select dissimilar landmarks to perform better.

5.2.2 Comparison of DTW Barycenter Averaging and k-Shape



(a) DTW-DBA

(b) cDTW-DBA with r = 0.1

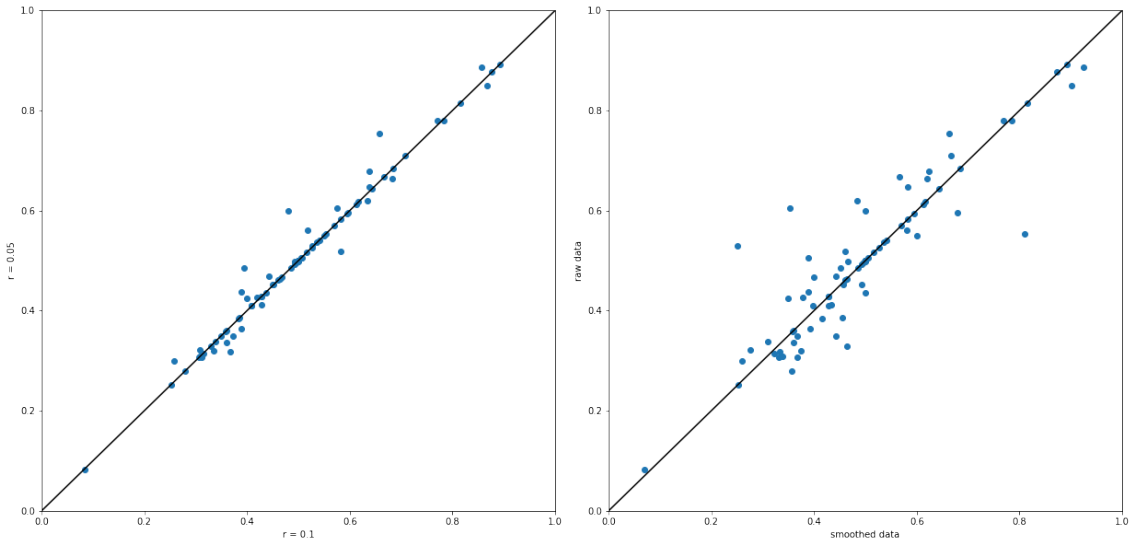
Figure 5.2: Comparison of k-shape and DTW Barycenter Averaging sampling for landmark selection

Method	< k-Shape	= k-Shape	> k-Shape
DTW-DBA	10	15	17
cDTW-DBA	22	25	20

Table 5.2: Performance of DTW and cDTW based Barycenter Averaging compared to k-shape for landmark selection

When we compare DTW based Barycenter Averaging methods and k-Shape, we can observe that DTW-DBA outperforms k-shape. However, due to multiple calculations of DTW, in terms of speed, DTW-DBA is definitely worse. Additionally, it can be observed that cDTW-DBA with a warping width of 10% is much faster than the DTW based method, but still slower than k-Shape. According to Table 5.2 and Figure 5.2, in comparison to k-Shape, cDTW does much better than k-Shape on datasets it is better on, while performs slightly worse on datasets where it is worse, giving a better performance overall. It also performs much better than DTW-DBA on the same datasets. However, it was observed that for some datasets, both cDTW-DBA are extremely slow and memory intensive, leading to memory issues in the environment. Therefore, despite a great performance in terms of classification accuracy, DTW-DBA and cDTW-DBA are not suitable for application on the edge and as online algorithms due to their slow processing speed and large memory requirement.

5.2.3 Comparison of TADPole and k-Shape



(a) $r = 0.05$ compared with $r = 0.10$ (b) analysis on smoothed data vs raw data

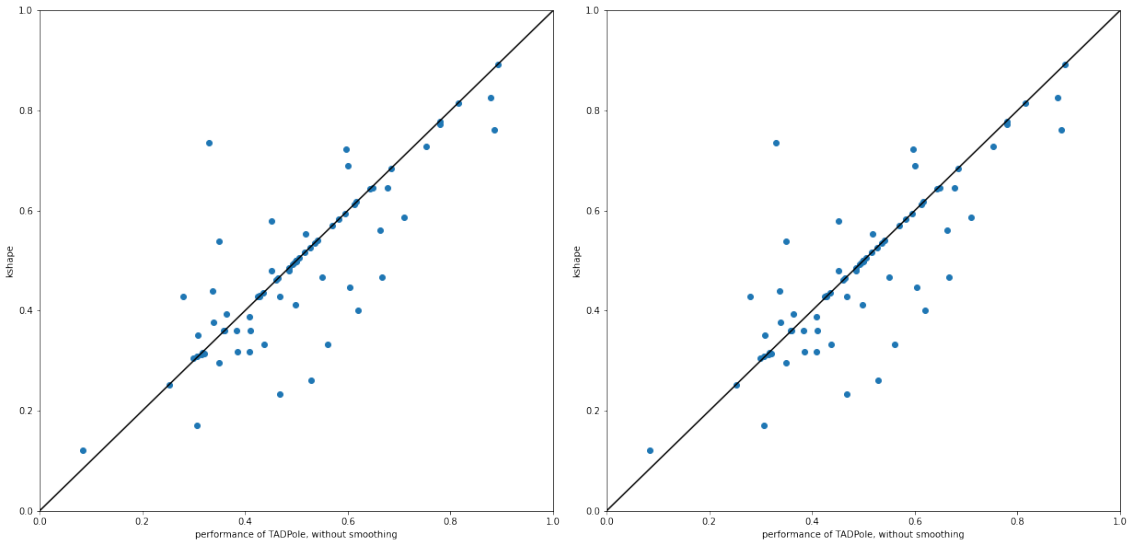
Figure 5.3: Comparison of different settings for TADPole

Method 1(a)	Method 2(b)	$a < b$	$a = b$	$a > b$
$TADPole_{0.05}$	$TADPole_{0.10}$	14	49	20
$TADPole_{raw}$	$TADPole_{smoothed}$	26	31	26

Table 5.3: Comparison of different settings for TADPole

When we analyse the classification accuracy for representations obtained from $r = 0.05$ and $r = 0.1$, as in Figure 5.3 we can clearly see that the more datasets prefer $r = 0.05$. However, because as many as 14 datasets show better accuracies with $r = 0.1$ 5.3. we can observe that the parameter r needs to be tuned based on information about data. This is generally done by using Leave One Out Classifiers which optimise the band to which the warping path is restricted using algorithms like Grid Search. As the performance for $r = 0.05$ is better, it is used for further analysis of TADPole.

On analysing the classification accuracy for TADPole over smooth and unsmoothed data, we observe that the datasets are equally split between both the methods. However, for a few of the datasets, the difference in accuracies is too high. Example, which are know to contain time series vectors which are not alligned at the start and end positions. Therefore, we conclude that for datasets where time series vectors are not alligned, we see a significant increase in accuracy, while for datasets which rely on data contained in higher frequencies and are properly alligned, a significant drop in accuracy is observed after smoothing.



(a) Data Agnostic TADPole (b) TADPole with selective smoothing

Figure 5.4: Comparison of k-shape and TADPole for landmark selection

Method	< k-Shape	= k-Shape	> k-Shape
Data Agnostic TADPole	29	6	43
TADPole with selective smoothing	26	6	46

Table 5.4: Performance of k-shape and TADPole sampling for landmark selection

It was observed that for the same setting of $\lambda = 20$, TADPole outperformed

k-Shape by a huge margin, as shown in Figure 5.4 and Table 5.4. Timing results also show that because of the pruning algorithm, TADPole performs significantly faster than k-Shape, and is therefore better at learning representations. Judging by the results from checking different settings for TADPole, we can observe that further optimisation in accuracy for landmark selection by TADPole is possible. However, TADPole requires a space complexity of $O(n^2)$, where n is the length of time series, which poses problems in scaling to very large datasets. We also observe that additional smoothing helps a few of the datasets to outperform k-Shape, however, barring few of the datasets, TADPole shows promising results.

Chapter 6

Conclusion and Future Work

In this project, we have addressed the problem of selecting an algorithm for landmark vector selection for Nystrom's method for time series datasets, in order to obtain a classification algorithm which can work in an online manner in resource constrained settings on the edge. From our experiments, we observe that TADPole, while having an $O(n^2)$ space complexity, can be implemented with a high degree of classification accuracy for shorter computation times, as compared to k-Shape. We also observe that due to its iterative one-shot clustering method, it is possible to implement it in an online setting. We also observe that selecting the landmark sequences using a uniform random selection, while extremely fast in terms of computation time, fails to guarantee a good precision during implementation, while DTW Barycenter Averaging is too slow and memory consuming, making it unusable in resource constrained environments despite its relatively high accuracy. Additionally, as only 1-Nearest Neighbours has been used for classifying time series vectors, more complex methods like k-NN and SVM based classifiers can surely show a much better performance.

As observed before, TADPole can be implemented in an online setting, primarily due to the Density Peaks algorithm it is based on, making clustering, which is the costliest operation in the GRAIL framework cheaper. However, exhaustive analysis of the online algorithm in terms of time and memory consumption. Additionally, a modification to Density Peaks to avoid the $O(n^2)$ memory complexity would ensure that this algorithm remains scalable for larger datasets. Another possible avenue for optimising this technique would be to study the effect of using the SBD distance measure to cluster time series data using a modification of TADPole.

Bibliography

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. pages 69–84. Springer Verlag, 1993.

[2] Shadab Alam, Franco D. Albareti, Carlos Allende Prieto, F. Anders, Scott F. Anderson, Timothy Anderton, Brett H. Andrews, Eric Armengaud, Eric Aubourg, and Stephen Bailey et al. THE ELEVENTH AND TWELFTH DATA RELEASES OF THE SLOAN DIGITAL SKY SURVEY: FINAL DATA FROM SDSS-III. *The Astrophysical Journal Supplement Series*, 219(1):12, jul 2015.

[3] O. T. at Twitter. Observability at twitter: technical overview, part i, 2016.

[4] Francis Bach and Michael Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 03 2003.

[5] Francis R. Bach and Michael I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 33–40, New York, NY, USA, 2005. Association for Computing Machinery.

[6] Anthony Bagnall, Aaron Bostrom, James Large, and Jason Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version, 2016.

[7] Gustavo Batista, Bilson Campana, and Eamonn Keogh. Classification of live moths combining texture, color and shape primitives. pages 903–906, 12 2010.

[8] Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn Keogh. Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 49–58, New York, NY, USA, 2015. Association for Computing Machinery.

[9] Bharat B. Biswal, Maarten Mennes, Xi-Nian Zuo, Suril Gohel, Clare Kelly, Steve M. Smith, Christian F. Beckmann, Jonathan S. Adelstein, Randy L.

Buckner, Stan Colcombe, Anne-Marie Dogonowski, and Monique Ernst et al. Toward discovery science of human brain function. *Proceedings of the National Academy of Sciences*, 107(10):4734–4739, 2010.

[10] Yuhan Cai and Raymond Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, page 599–610, New York, NY, USA, 2004. Association for Computing Machinery.

[11] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Yanping Ratanamahatana, Chotirat Ann, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. The ucr time series classification archive, October 2018.

[12] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[13] Ahmed Farahat, Ali Ghodsi, and Mohamed Kamel. A novel greedy algorithm for Nyström approximation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 269–277, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[14] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[15] Zoltan Geler, Vladimir Kurbalija, Mirjana Ivanović, Miloš Radovanović, and Weihui Dai. Dynamic time warping: Itakura vs sakoe-chiba. In *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6, July 2019.

[16] Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions : Simple and deterministic matrix sketching, 2015.

[17] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[18] A. K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

[19] Charles Loboz, Slawek Smyl, and Suman Nath. Datagarage: Warehousing massive performance data on commodity servers. *Proc. VLDB Endow.*, 3(1–2):1447–1458, sep 2010.

[20] Cameron Musco and Christopher Musco. Recursive sampling for the nyström method, 2016.

[21] P. Nemenyi. Distribution-free multiple comparisons, 1963.

[22] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, 3rd edition, 2009.

[23] John Paparrizos and Michael J. Franklin. Grail: Efficient time-series representation learning. *Proc. VLDB Endow.*, 12(11):1762–1777, jul 2019.

[24] John Paparrizos and Luis Gravano. K-shape: Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 1855–1870, New York, NY, USA, 2015. Association for Computing Machinery.

[25] François Petitjean, Alain Ketterlin, and Pierre Gançarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3):678–693, 2011.

[26] Kin pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings of 15th International Conference on Data Engineering (ICDE '99)*, pages 126–133, 1999.

[27] Thanawin Rakthanmanon, Eamonn J. Keogh, Stefano Lonardi, and Scott Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *2011 IEEE 11th International Conference on Data Mining*, pages 547–556, Dec 2011.

[28] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[29] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[30] Pongsakorn Ruengronghirunya, Vit Niennattrakul, and Chotirat Ann Ratanamahatana. Speeding up similarity search on a large time series dataset under time warping distance. In *Advances in Knowledge Discovery and Data Mining*, pages 981–988, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[31] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech, Lang. Process.*, 26:43–50, 1978.

[32] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, oct 2007.

[33] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING*, pages 327–352. MIT Press, 1999.

[34] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[35] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[36] Renjie Wu and Eamonn J. Keogh. Fastdtw is approximate and generally slower than the algorithm it approximates. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[37] Tianbao Yang, Yu Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in Neural Information Processing Systems 25*, Advances in Neural Information Processing Systems, pages 476–484, December 2012.

[38] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1232–1239, New York, NY, USA, 2008. Association for Computing Machinery.