UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

André Luís Mendes Fakhoury

Time Series Classification Using the Optimum-Path Forest Algorithm

São Carlos

2022

André Luís Mendes Fakhoury

Time Series Classification Using the Optimum-Path Forest Algorithm

Monograph presented to the Undergraduate Program in Computer Science at the Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, as part of the requirements for obtaining a bachelor's degree.

Advisor: Prof. Diego Furtado Silva, Ph.D.

São Carlos

2022

Fakhoury, André Luís Mendes
S856m Time Series Classification Using the Optimum-Path Forest Algorithm / André Luís Mendes Fakhoury ; orientador Diego Furtado Silva. – São Carlos, 2022.
34 p. : il. (algumas color.) ; 30 cm.

1. Time series. 2. Machine learning. I. Diego Furtado Silva. II. Universidade de São Paulo. III. Instituto de Ciências Matemáticas e de Computação. IV. Time Series Classification Using the Optimum-Path Forest Algorithm.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family for helping me through this journey. Second, I express my gratitude to my advisor, Diego Furtado Silva, for guiding me this semester and introducing me to the time series. Third, I thank all the Competitive Programming Study Group members, who certainly made me a better person and developer since my first year of graduation. Last but not least, I express my gratitude to João do Espírito Santo Batista Neto, my previous supervisor, for all his patience and for tutoring me in research over the last few years.

"As long as you are learning, you are not failing."
Bob Ross

ABSTRACT

FAKHOURY, A. L. M. **Time Series Classification Using the Optimum-Path Forest Algorithm**. 2022. 34p. Monografia (Trabalho de Conclusão de Curso) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

The popularization of mobile sensors and other technologies that continuously collect data over time increases the demand for Machine Learning algorithms for time series. One of the most common tasks in this domain is classification, and traditionally the most known algorithms in this field are similarity-based. In addition to reaching excellent results in several datasets, this kind of algorithm is also interpretable, a desirable feature in Machine Learning techniques. In the last decade, alternatively, several algorithms based on classification committees, artificial neural networks, and more approaches were proposed. Some of these methods, although not interpretable, are computationally efficient and reach better results on average than similarity-based ones. In this scenario, this work aims to explore the Optimum-Path Forest (OPF) algorithm in the time series classification domain. The OPF is a similarity-based technique that obtained excellent results in several tasks, overcoming well-known methods such as Support Vector Machine and Gradient Boosting. However, in time series domains, OPF was never explored. One possible explanation is that this algorithm demands the calculation of a distance matrix between all objects in the dataset, which can be unviable for long time series or datasets with large volumes. In particular, the best distance measures between time series reported in the literature are calculated by $O(n^2)$ algorithms in the number of observations of the series, which increases the complexity of the classification algorithm. However, the time series literature presents a diversity of techniques to prune the calculation of distance measures, allowing a significant speedup in several applications. Therefore, this work aims to (a) evaluate the performance of OPF in time series classification and (b) propose runtime-saving techniques to turn it computationally viable, even for large data volumes. Our results from 128 datasets show that the supervised OPF classifier obtains a better accuracy than KNN in some of them. OPF using DTW distance at least draws with 1-NN using Euclidean distance in 71.9% of the datasets, and it at least draws with 1-NN using DTW distance in 39.1% of the datasets.

Keywords: Time series. Machine learning. Classification.

RESUMO

FAKHOURY, A. L. M. **Time Series Classification Using the Optimum-Path Forest Algorithm**. 2022. 34p. Monografia (Trabalho de Conclusão de Curso) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

A popularização de sensores móveis e outras tecnologias que coletam dados continuamente sobre o tempo aumenta a demanda para algoritmos de aprendizado de máquina para séries temporais. Uma das tarefas mais comuns neste domínio é a classificação e, tradicionalmente, os algoritmos mais conhecidos neste campo são baseados em similaridade. Além de atingirem resultados excelentes em vários conjuntos de dados, este tipo de algoritmo também é interpretável, uma característica desejável em técnicas de aprendizado de máquina. Na última década, alternativamente, vários algoritmos baseados em comitês de classificadores, redes neurais artificiais e outras abordagens foram propostos. Alguns desses métodos, mesmo que não interpretáveis, são eficientes computacionalmente e atingem melhores resultados em média que aqueles baseados em similaridade. Nesse cenário, este trabalho objetiva explorar o algoritmo *Optimum-Path Forest* (OPF) para o domínio de classificação de séries temporais. O OPF é uma técnica baseada em similaridade que obtém resultados excelentes em várias tarefas, superando métodos bem conhecidos como *Support Vector Machine* e *Gradient Boosting*. Entretando, o OPF não foi explorado no domínio de séries temporais. Uma das possíveis explicações para isso é que o algoritmo requer o cálculo de uma matriz de distâncias entre todos os objetos do conjunto de dados, o que pode ser inviável para séries temporais longas ou conjuntos de maior volume. Em particular, os melhores métodos de distância entre séries temporais encontrados na literatura são calculados por algoritmos de complexidade $O(n^2)$ no número de observações das séries, aumentando a complexidade do algoritmo de classificação. Porém, a literatura de séries temporais apresenta uma diversidade de técnicas para podar o cálculo da distância, permitindo uma significante melhora no tempo de execução em várias aplicações. Desta forma, este trabalho tem como objetivo (a) analisar a eficiência do OPF para a classificação de séries temporais e (b) propor técnicas de economia de tempo de execução para tornar o algoritmo computacionalmente viável, mesmo para grandes volumes de dados. Os resultados obtidos em 128 conjuntos de dados mostram que o classificador supervisionado OPF obteve uma acurácia melhor que o KNN em alguns conjuntos. Além disso, o OPF utilizando a distância DTW pelo menos empata com 1-NN utilizando distância euclidiana em 71.9% dos conjuntos, e pelo menos empata com 1-NN utilizando a distância DTW em 39.1% dos conjuntos.

Palavras-chave: Séries temporais. Aprendizado e máquina. Classificação.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

OPF	Optimum-Path Forest
MST	Minimum-Spanning Tree
ED	Euclidean Distance
DTW	Dynamic Time Warping
KNN	k-Nearest Neighbors
1NN-ED	1-Nearest Neighbors using the Euclidean distance
1NN-DTW	1-Nearest Neighbors using the Dynamic Time Warping distance
OPF-ED	Optimum-Path Forest Classifier using the Euclidean distance
OPF-DTW	Optimum-Path Forest Classifier using the Dynamic Time Warping distance

CONTENTS

1 INTRODUCTION

The popularization of mobile sensors and other technologies that continuously generate data arranged in time increases the demand for Machine Learning algorithms for time series. Some popular temporal data mining tasks in this domain are classification, clustering, motif discovery, and anomaly detection (SILVA, 2017).

Traditionally, the most known algorithms for time series classification are similarity-based. In addition to reaching excellent results in several datasets, this kind of algorithm is also interpretable, a desirable feature in Machine Learning techniques. In the last decade, alternatively, several algorithms based on classification committees, artificial neural networks, and more approaches were proposed. Some of these methods are computationally efficient and reach better results on average than similarity-based ones. However, they are not interpretable.

In this context, the 1-Nearest Neighbor (1-NN) classifier using dynamic time warping as a distance metric usually gives excellent results and is hard to be beaten (WANG *et al.*, 2013). Although having a quadratic complexity on the number of observations of the time series, there are some well-known techniques to prune the calculation of distance measures, allowing a significant speedup in several applications.

The Optimum-Path Forest (OPF) is a framework used in pattern recognition based on graphs. There are several variations of classifiers using OPF: the supervised classifier (PAPA *et al.*, 2007; PAPA; FALCãO, 2008; PAPA; FALCãO; SUZUKI, 2009), unsupervised (CAPPABIANCO; FALCãO; ROCHA, 2008; ROCHA; CAPPABIANCO; FALCãO, 2009), semi-supervised (AMORIM *et al.*, 2016), with a learning algorithm (PAPA; FALCÃO, 2009), using fuzzy logic (SOUZA *et al.*, 2020), using unsupervised manifold learning (AFONSO; PEDRONETTE; PAPA, 2018), improved for large datasets (PAPA; CAPPABIANCO; FALCãO, 2010; PAPA *et al.*, 2012) and using a combination of disjoint training subsets (PONTI; PAPA, 2011). However, OPF was never used for time series classification.

The supervised OPF classifier is similarity-based and uses each training set sample as vertices of a complete graph, weighted by the distance between their feature vectors. For time series, it can be interesting to compute these edges' weights using the Euclidean distance or DTW distance. However, it can become nonviable for huge time series if we don't use any heuristic to prune or early abandon this calculation.

In these circumstances, in this work, we aim to analyze the supervised Optimum-Path Forest classifier for time series classification and propose runtime-saving techniques to turn it computationally viable, even for large data volumes.

2 MATERIALS AND METHODS

In this section, we discuss the employed datasets and the methodology used in the classification process. In Section 2.1 we discuss the datasets used in our work. In Section 2.2 we present the theory of Minimum-Spanning Trees, that will be useful in the training step of OPF Classifier. Then, in Section 2.3 we present the logic behind the supervised Optimum-Path Forest classifier.

2.1 Datasets

We took the data used in this work from the UCR Time Series Classification Archive repository (DAU *et al.*, 2018). This repository contains 128 datasets, each contemplating a collection of time series, split into train and test sets for the classification process. The training set is used to project and prepare a classifier, and the test set is used for measure the classification error.

The classification error is a decimal number between 0 and 1, obtained by dividing the number of wrong classifications (samples that received the wrong label) over the number of test samples.

All datasets include two files organized in tables containing the corresponding training and test samples. Each row of each table represents a distinct time series, where the first column contains its label, and each next column represents the value for each observation.

Besides presenting the data, the archive also includes, in a different sheet, additional information for each dataset, such as sample type (image, sensor, Spectro), number of training and test samples, length of each time series, and some error rates of basic classification methods (for example, 1-NN using Euclidean distance), as illustrated in Table 1.

We used all datasets with equal time series length and without missing values. The UCR Archive already handles these cases, replacing missing values using linear interpolation and adding low-amplitude random numbers to the end of the time series with different lengths.

2.2 Minimum-Spanning Tree

The minimum-spanning tree (MST) problem aims to find a spanning tree with a minimum total weight of an undirected, connected, and weighted graph (TARJAN, 1983). A spanning tree of some graph G is formed only by the edges of it and connects all its vertices (i.e., it is a subtree of G). Here, the total weight of a graph is the sum of its edges.

Table 1 – A portion of the available information of the first 5 datasets from the *UCR Archive*. Namely, we have the training and test sample size, the number of classes, the length of each time series, and the classification error of 1-NN using the Euclidean distance (1NN-ED).

Type	Name	Train	Test	Class	Length	1NN-ED	Data donor/editor
Image	Adiac	390	391	37	176	0.389	A. Jalba
Image	ArrowHead	36	175	3	251	0.200	L. Ye, E. Keogh
Spectro	Beef	30	30	5	470	0.333	K. Kemsley, A. Bagnall
Image	BeetleFly	20	20	2	512	0.250	J. Hills, A. Bagnall
Image	BirdChicken	20	20	2	512	0.450	J. Hills, A. Bagnall

Source: Dau *et al.* (2018)

Minimum-spanning trees have a notable property that is worth mentioning, called Cut Property:

Property 2.2.1 (Cut property)**.** *For any cut C of the graph G (i.e., for any partition of the nodes V into two nonempty disjoint subsets $(S, V - S)$), if the weight of an edge e in the cut-set S is strictly smaller than the weights of all other edges from S, then this edge belongs to all minimum-spanning trees of G (MINEA, 2018).*

Proof. Assume by ways of contradiction that there is an MST T that does not contain the edge e. If we insert e in T, it will create a cycle inside cut C that contains an edge e' with a weight strictly larger than e. We can delete this edge and get a spanning tree $T' = (T \cup \{e\}) \setminus \{e'\}$ of weight strictly smaller than T, which contradicts the assumption that T was an minimum-spanning tree. ■

Now, we discuss one of the most famous algorithms to solve this problem, named Prim's algorithm. OPF uses this algorithm in the training step (see Section 2.3.2).

2.2.1 Prim's algorithm

In Prim's algorithm, to build the minimum spanning tree, we start at an arbitrary root vertex r and maintain a set S that stores the current building spanning tree. At each step, we find the minimum weight edge e that goes from a vertex u s.t. $u \in S$ to a vertex v s.t. $v \notin S$. Then, we add this edge e to S. Notice that, at each step, the number of vertices reachable from r will grow by one.

We can observe that S is always a tree because, by the definition of the algorithm, we will never add an edge that connects two vertices belonging to S. More specifically, by the Cut Property (2.2.1), S is a minimum-spanning tree (CORMEN *et al.*, 2001).

2.3 Optimum-Path Forest

The Optimum-Path Forest is an efficient framework used in pattern recognition. The first version of a supervised classifier was presented by Papa *et al.* (2007), but later other variations were described, including new supervised, semi-supervised, and unsupervised classification techniques. OPF has applications in various fields, such as medicine, speech recognition, and remote sensing (FALCAO; PAPA, 2022).

In this section, we describe the supervised OPF Classifier, which will be the prime tool used in this work.

2.3.1 Supervised Optimum-Path Forest Classifier

Let Z_1 and Z_2 be the training and test set, respectively. Let $\lambda(s)$ be the function that correctly assigns a label to a given sample s (i.e., the correct class of s). We want to predict labels of $t \in Z_2$ based in what we know about Z_1.

Here we define the supervised classification method described by Papa, Falcão and Suzuki (2009). The samples are converted to a graph, where each class is represented with one or more optimum-path trees rooted in a chosen sample called a prototype. In the prediction step, we choose the test label considering the closest optimum-path tree. We better discuss this process in Sections 2.3.2 and 2.3.3.

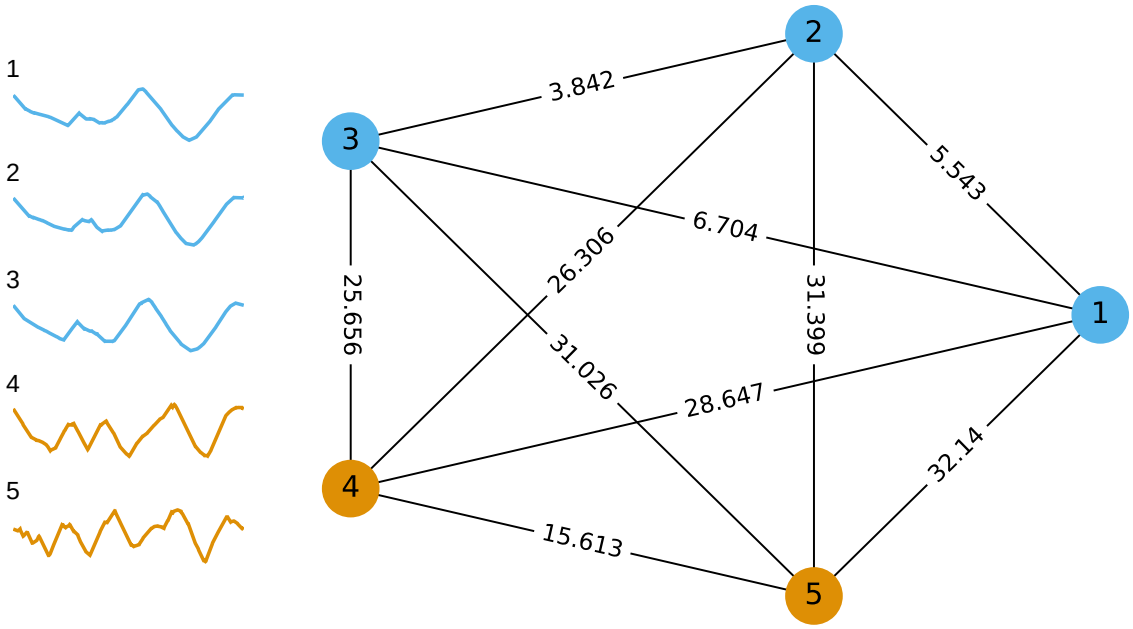
2.3.2 Training process

First, the training sample is converted into a complete graph, whose vertices are samples and edges are weighted by a distance function between their feature vectors. In this work, we define the distance function as the distance between the time series, as can be seen in Section 2.3.4. Figure 1 illustrates a complete graph weighted by the Euclidean distance between the original time series.

Second, the algorithm chooses (at least one) vertex of each class as a prototype, using a minimum-spanning tree T (explained in Section 2.2) of the graph. The prototype set S^* is formed by all the endpoints of edges from T that connect vertices of different labels. For a path cost function f_{max} (maximum of weights along the path), described later in Equations 2.1 and 2.2, this choice of prototypes seeks to minimize the classification errors when their labels are propagated to the vertices of their trees (PAPA *et al.*, 2012). Note that, as the graph is connected, each class is represented by at least one prototype. Figure 2 demonstrates the MST of the complete graph shown in Figure 1 and the chosen prototypes accordingly.

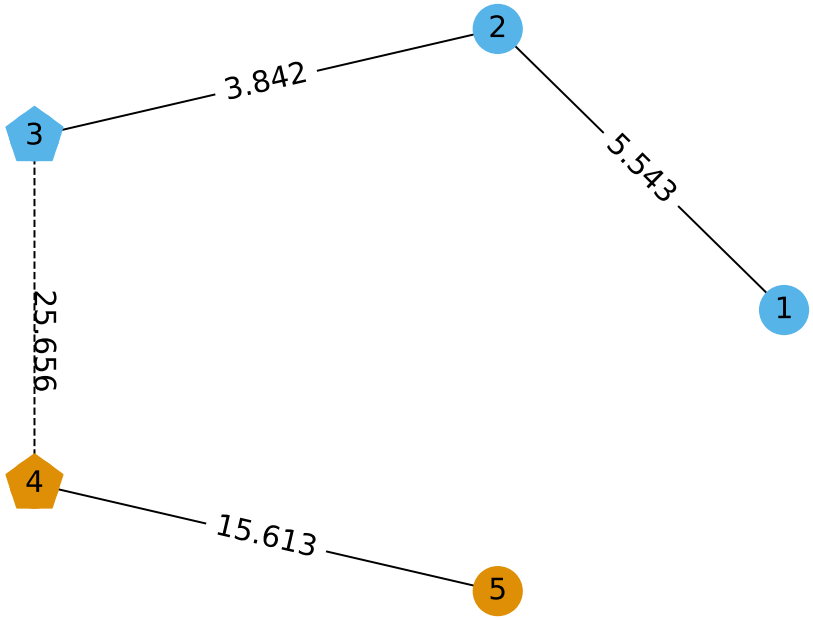
The next step is to define a path cost function for each vertex. The only restriction of the cost function is that it needs to be smooth (FALCãO; STOLFI; LOTUFO, 2004). The cost of each sample s ($s \notin S^*$) is the minimum path cost function from any prototype

Figure 1 – Illustration of the time series (left side) and the complete graph formed by them (right side). Different classes are represented with different colors, and the edges are weighted by the Euclidean distance between distinct series.



Source: The authors.

Figure 2 – Minimum spanning tree obtained from the graph in Figure 1. The dashed edge represents a connection between nodes of different labels, and pentagon-shaped nodes represent the chosen prototypes.



Source: The authors.

$p \in S^*$ to s. The most used are f_{sum} (sum of weights along the path) and f_{max}. In this work, we adopted f_{max}.

Equation 2.1 describes the cost of a single vertex s, and Equation 2.2 describes the cost of a path, concatenating adjacent vertices.

$$f_{max}(< s >) = \begin{cases} 0, \text{ if } s \in S^*, \\ \infty \text{ otherwise} \end{cases} \tag{2.1}$$

$$f_{max}(\pi_s \cdot < s, t >) = max\{f_{max}(\pi_s), d(s,t)\} \tag{2.2}$$

where $d(s,t)$ is the weight of edge (s,t).

Every sample s will have an extra label $L(s)$, which will be used in the classification process. If s is a prototype (i.e., $s \in S^*$), $L(s) = \lambda(s)$, where $\lambda(s)$ is the original class of s; otherwise, $L(s) = L(P(s))$, where $P(s)$ is the predecessor of s in the optimum-path tree. Figure 3 illustrates this process step-by-step.

2.3.3 Classification process

The classification of a test sample $t \in Z_2$ is made by connecting an edge from t to all training samples $s \in Z_1$ in the previously built forest of optimum paths, with weight equal to the distance between their feature vectors. We want to find the best route from a prototype $p \in S^*$ to t. The optimum f_{max} path cost of it is described by Equation 2.3.

$$C(t) = \min_{\forall s \in Z_1} \{\max\{C(s), d(s,t)\}\} \tag{2.3}$$

Let s^* be the sample that satisfies Equation 2.3, expressly the predecessor of t in the optimum-path tree of some root prototype p. We classify t as the same class of p by choosing the label of $L(s^*)$, described in Section 2.3.2.

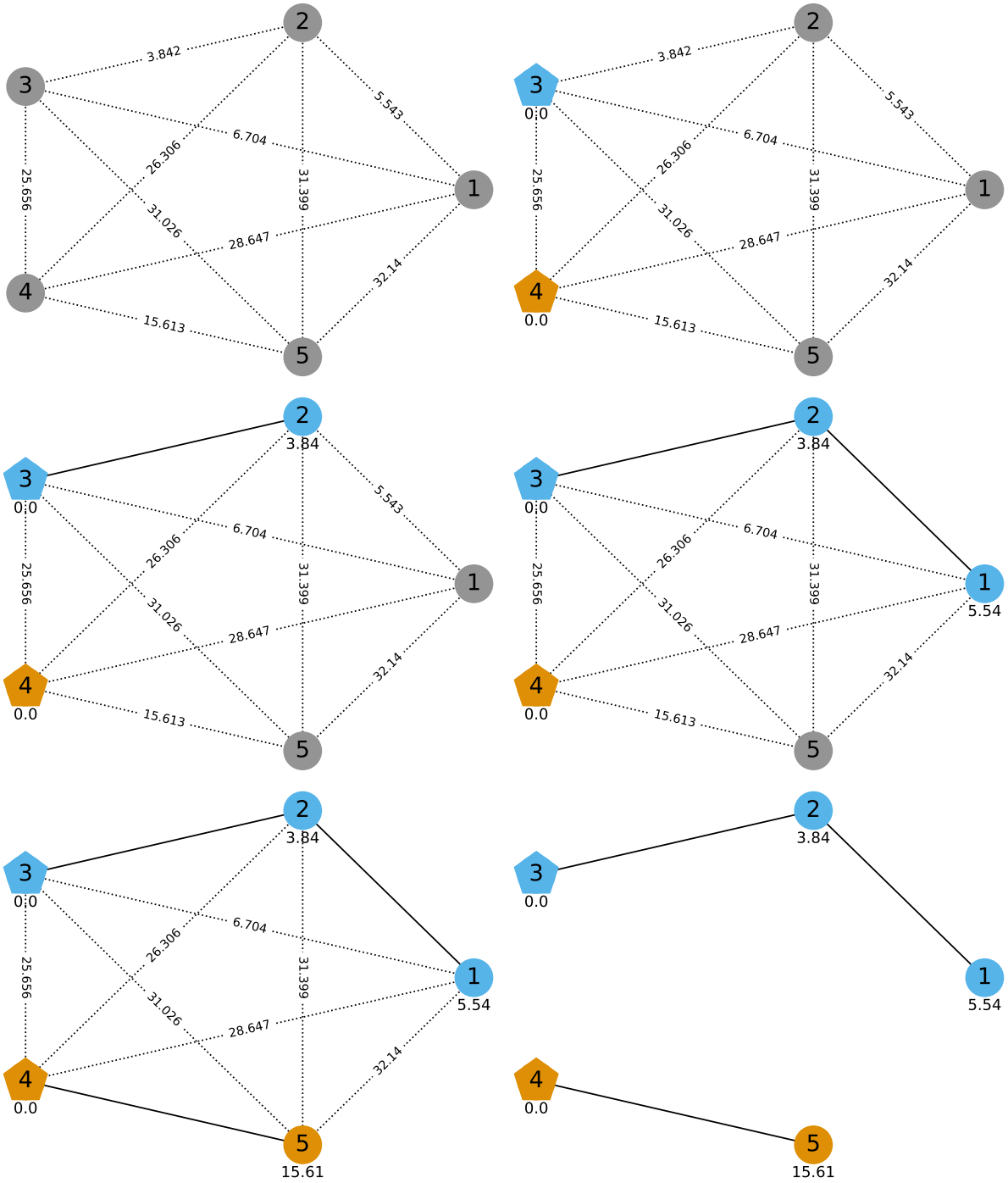
2.3.4 Time series implementation

For the time series classification problem, we can define a complete graph G that contains the time series training samples as vertices, and the distance between them can be described by a distance function F. In our case, we used two distance functions: *Euclidean distance* (ED) and *dynamic time warping* (DTW) distance.

According to Holder, Middlehurst and Bagnall (2022), given two univariate time series t and s of the same length l, the Euclidean distance is the $L2$ norm between them, given by Equation 2.4.

$$ED(t,s) = \sqrt{(t_1 - s_1)^2 + (t_2 - s_2)^2 + \cdots + (t_l - s_l)^2} \tag{2.4}$$

Figure 3 – Step-by-step definition of path-cost functions for each vertex, and creation of optimum-path trees. Unknown labels are colored gray, and other labels are colored as defined in Figure 1. The costs are positioned below each vertex (when defined). Initially, the prototypes are chosen with a cost of 0. Then, the cost of each vertex is defined by the distance to the closest prototype.



Source: The authors.

And the DTW distance of s and t is usually calculated by a dynamic programming approach, achieving an optimal elastic alignment under boundary, monotonicity, and continuity constraints (SILVA; BATISTA, 2016). Equation 2.5 presents the base cases of the function.

$$D(i,j) = \begin{cases} \infty, \text{ if } i = 0 \text{ or } j = 0, \\ 0, \text{ if } i = j = 0 \end{cases} \tag{2.5}$$

And Equation 2.6 describes the recurrence relation of the algorithm.

$$D(i,j) = c(s_i, t_j) + \min \left\{ \begin{matrix} D(i-1,j), \\ D(i,j-1), \\ D(i-1,j-1) \end{matrix} \right\}, 1 \leq i,j \leq n \tag{2.6}$$

where $c(s_i, t_j)$ is the cost of pairing the observations s_i and t_j, usually calculated by the Euclidean distance. The DTW distance is the value of $D(l,l)$. One can note that DTW can also be used in time series with different lengths without loss of generality, but our work focuses on time series of the same length.

We can also use a fixed window length of the difference between indices i and j in Equation 2.6. Equation 2.7 demonstrates the recurrence relation of the DTW if we fix a window length w (where $w \leq n$), for all indices $1 \leq i,j \leq n$.

$$D(i,j) = \begin{cases} c(s_i, t_j) + \min\{D(i-1,j), D(i,j-1), D(i-1,j-1)\}, \text{ if } |i-j| \leq w \\ \infty, \text{ otherwise} \end{cases} \tag{2.7}$$

where $c(s_i, t_j)$ is also the cost of pairing two observations s_i and t_j.

To improve the efficiency of calculating DTW, we used the optimizations described in Rakthanmanon *et al.* (2012), known as the *UCR Suite*. Before calculating the DTW, we normalize the time series to make comparisons (KEOGH; KASETTY, 2003).

- *Using the Squared Distance:* omit the calculation of squared root on the code to save computing time and enable some optimizations, just taking it afterward;
- *Lower Bounding:* prune off unpromising candidates with a cheap-to-compute lower bound, as *Euclidean distance*, *LB_Kim*, and *LB_Keogh*;
- *Early Abandoning:* current calculations can be abandoned if they get greater than the best answer so far. It can be applied when calculating the *ED*, *LB_Keogh*, *DTW* and even the *Z-Normalization*;

- *Reordering Early Abandoning:* sort the indices based on absolute values of the query series Q after Z-normalization to improve the speedup of early abandoning;
- *Cascading Lower Bounds:* use all lower bounds to prune off unpromising candidates in cascade, starting with the ones with less complexity to calculate.

3 EXPERIMENTS

In this section, we describe the experiments made, first by testing the OPF Classifier on the dataset (Sections 3.1 and 3.2) and later by optimizing and pruning the distance calculations (Section 3.3), making it possible to batch-test all the datasets (Section 3.4).

All experiments were performed on a 64-bit operating system with Processor Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz and 16GB of RAM.

3.1 Initial tests

We started by testing OPF Classifier with a few datasets to validate how it would perform with time series as samples. We selected 6 datasets from the UCR Archive (DAU *et al.*, 2018) of different sizes (considering training set sizes and time series lengths) to use as input to the classifier, first using Euclidean distance as the distance function. Table 2 describes these datasets.

Table 2 – Description of the 6 datasets chosen to represent data, containing the name, type of origin, training and test number of samples, number of labels, length of each series, and error of classification obtained via the 1-Nearest Neighbor using Euclidean Distance (1NN-ED) as a metric.

Name	Type	Train	Test	Class	Length	1NN-ED
WordSynonyms	Image	267	638	25	270	0.3824
SemgHandSubjectCh2	Spectrum	450	450	5	1500	0.5956
PLAID	Device	537	537	11	1344	0.4767
MelbournePedestrian	Traffic	1194	2439	10	24	0.1525
ChlorineConcentration	Sensor	467	3840	3	166	0.3500
ShapesAll	Image	600	600	60	512	0.2483

Source: Dau *et al.* (2018)

Table 3 shows the classification errors of OPF, using the Euclidean distance to build the complete graph, and the 1-NN Euclidean distance classification error present in UCR Archive.

These preliminary results show that OPF produces a similar error on those datasets, so we continue further analyzing it.

Table 3 – Classification errors for each dataset using the Euclidean distance as metric in the OPF Classifier (OPF-ED) and 1-NN (1NN-ED), and Runtime ($\mu \pm \sigma$ of 10 interactions) of OPF training and classification.

Name	OPF-ED	1NN-ED	OPF-ED Runtime (s)
WordSynonyms	0.3871	0.3824	1.0942 ± 0.0412
SemgHandSubjectCh2	0.2111	0.5956	6.1674 ± 0.1357
PLAID	0.3277	0.4767	10.5680 ± 0.3786
MelbournePedestrian	0.0558	0.1525	13.8631 ± 0.1259
ChlorineConcentration	0.3555	0.3500	6.9485 ± 0.1232
ShapesAll	0.2517	0.2483	3.9778 ± 0.0416

Source: The authors and Dau *et al.* (2018)

3.2 DTW distance

The next step was to test the algorithm using the DTW distance to weigh the edges. Visualizing the results from Dau *et al.* (2018), one can see that it achieves better results than Euclidean distance, considering the 1-NN classification. Table 4 shows the classification errors of the 1-NN and OPF Classifier using DTW Distance, with a fixed window length of $w = 10\%$ of the series length.

Table 4 – Classification errors for each dataset using the Dynamic Time Warping (DTW) distance, with fixed window size of $w = 10\%$ of series length, as metric in the OPF Classifier (OPF-DTW) and 1-NN (1NN-DTW), and Runtime ($\mu \pm \sigma$ of 10 interactions) of OPF training and classification

Name	OPF-DTW	1NN-DTW	OPF-DTW Runtime (s)
WordSynonyms	0.2680	0.2649	13.1987 ± 0.1140
SemgHandSubjectCh2	0.1244	0.1200	614.2726 ± 4.6539
PLAID	0.0987	0.0931	747.8448 ± 1.3472
MelbournePedestrian	0.0836	0.0791	17.8932 ± 0.1285
ChlorineConcentration	0.3607	0.3518	41.1680 ± 0.0875
ShapesAll	0.2217	0.2117	141.4626 ± 0.3572

Source: The authors.

We can see that it leads us to similar results of 1-NN: DTW gives us proportionally better results than Euclidean distance in both OPF and 1-NN. However, the runtime needed for this classification increased almost 100 times in some datasets, as the DTW has quadratic complexity on the time series' length. Next, we perform some prunings and

efficiency improvements made on the implementation to make DTW computations viable on larger datasets.

3.3 Pruning and improvements

Our results with a few datasets show that OPF at least draws in some cases with 1-NN but is not scalable to larger datasets. In that way, we performed some efficiency improvements aiming to reduce runtime.

The first enhancement was to code the classifier and distance calculations in C++ instead of Python. It led to a big time-saver in Euclidean distance, as shown in Table 5.

Table 5 – Runtime comparison of Python and C++ implementations of classification using OPF Classifier with Euclidean distance. Each runtime is represented as $\mu \pm \sigma$ of 10 interactions. Speedup was calculated by dividing Python by C++ runtime.

Name	Python Runtime (s)	C++ Runtime (s)	Speedup
WordSynonyms	1.0942 ± 0.04118	0.0511 ± 0.0009	21.4129 ± 0.8898
SemgHandSubjectCh2	6.1674 ± 0.1357	0.3866 ± 0.0114	15.9529 ± 0.5869
PLAID	10.5680 ± 0.3786	0.5514 ± 0.0240	19.1658 ± 0.1080
MelbournePedestrian	13.8631 ± 0.1259	0.0693 ± 0.0065	200.0447 ± 0.1885
ChlorineConcentration	6.9485 ± 0.1232	0.2545 ± 0.0034	27.3026 ± 0.6061
ShapesAll	3.9778 ± 0.0416	0.2611 ± 0.0128	15.2348 ± 0.7637

Source: The authors.

The next pitfall to deal with was the DTW complexity. We used the pruning techniques and heuristics previously discussed in Section 2.3.4. Table 6 shows the time-saving of this method.

3.4 Batch-tests

The optimizations described in Section 3.3 enabled the possibility of testing the OPF classifier in all datasets from the UCR Archive using the Euclidean distance as an edge-cost function in under an hour of runtime.

Table 7 presents the 10 datasets with the greatest absolute differences from OPF and 1-NN errors, considering the Euclidean distance.

We also tested the datasets using the DTW distance to model the graph. Table 8 presents those that obtained errors in the OPF classification that was farther from the 1-NN classification errors, considering the DTW distance.

Table 6 – Runtime comparison of OPF classification using DTW distance with and without pruning. Runtimes are represented as $\mu \pm \sigma$ of 10 interactions. The speedup was calculated by dividing the previous by pruned runtimes.

Name	Previous Runtime (s)	Pruned Runtime (s)	Speedup
WordSynonyms	13.1987 ± 0.1140	11.8134 ± 0.0407	1.1173 ± 0.0104
SemgHandSubjectCh2	614.2726 ± 4.6539	472.2561 ± 1.2915	1.3007 ± 0.0105
PLAID	747.8448 ± 1.3472	524.9052 ± 4.0287	1.4247 ± 0.0112
MelbournePedestrian	17.8932 ± 0.1285	5.7252 ± 0.0878	3.1253 ± 0.0529
ChlorineConcentration	41.1680 ± 0.0875	43.8011 ± 0.3000	0.9399 ± 0.0067
ShapesAll	141.4626 ± 0.3572	105.9866 ± 0.1777	1.3347 ± 0.0040

Source: The authors.

Table 7 – Name, classification error rates of OPF-ED and 1NN-ED and respective relative difference of classifiers on the top 10 datasets with the greatest absolute difference between 1-NN and OPF classifications, using the Euclidean distance.

Name	1NN-ED	OPF-ED	Difference
SemgHandSubjectCh2	0.5956	0.2111	0.3845
InsectEPGSmallTrain	0.3373	0.0000	0.3373
InsectEPGRegularTrain	0.3213	0.0000	0.3213
Rock	0.1600	0.4000	-0.2400
SemgHandMovementCh2	0.6311	0.4067	0.2244
PigArtPressure	0.8750	0.7067	0.1683
PLAID	0.4767	0.3277	0.1490
SemgHandGenderCh2	0.2383	0.1017	0.1366
MelbournePedestrian	0.1525	0.0558	0.0967
GestureMidAirD1	0.4231	0.5154	-0.0923

Source: The authors.

Appendix A presents the classification errors obtained on all datasets, and Section 3.5 analyzes them.

3.5 Results

In this section, we discuss the results of the OPF Classifier on all 128 datasets from the UCR Archive (Appendix A shows the raw data).

Figure 4 illustrates the results obtained from OPF Classifier (with both Euclidean

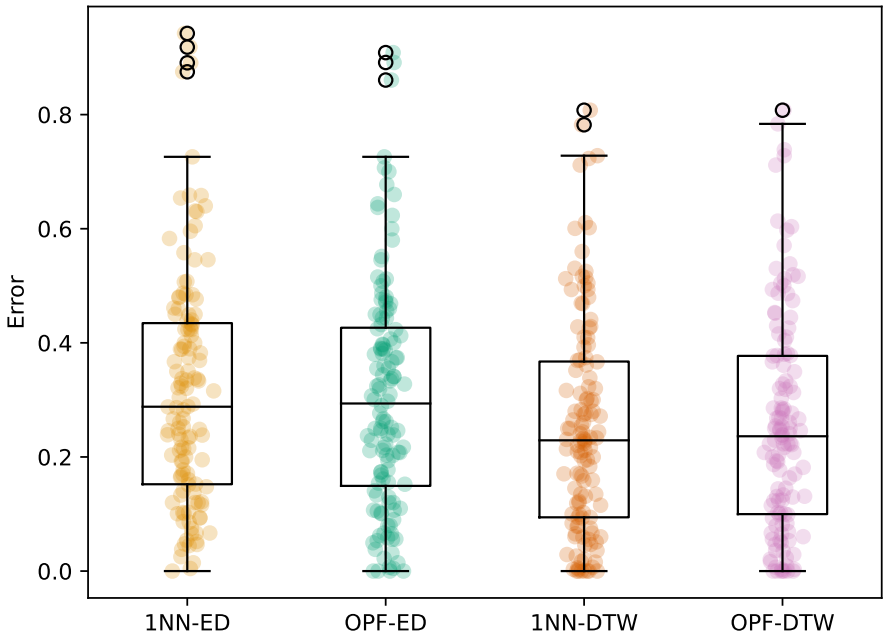
Table 8 – Name, classification error rates of OPF-DTW and 1NN-DTW and respective relative difference of classifiers on the top 10 datasets with the greatest absolute difference between 1-NN and OPF classifications, using the DTW distance.

Name	1NN-DTW	OPF-DTW	Difference
BeetleFly	0.3000	0.2500	0.0500
ToeSegmentation1	0.2281	0.2675	-0.0394
OliveOil	0.1667	0.1333	0.0334
MiddlePhalanxOutlineCorrect	0.3024	0.3333	-0.0309
SonyAIBORobotSurface2	0.1595	0.1868	-0.0273
MiddlePhalanxOutlineAgeGroup	0.5000	0.4740	0.0260
Computers	0.4280	0.4520	-0.0240
OSULeaf	0.4091	0.4298	-0.0207
Rock	0.4800	0.5000	-0.0200
AllGestureWiimoteX	0.2343	0.2543	-0.0200

Source: The authors.

and DTW distance variations), besides 1-NN classification errors.

Figure 4 – Boxplot of classification errors for each classifier algorithm: 1NN-ED, OPF-ED, 1NN-DTW, and OPF-DTW. The illustration also shows scatter plots of each classification error, presented with a jitter for better visualization.



Source: The authors.

Table 9 details those OPF results describing mean and standard deviation, along

with the number of wins, draws, and losses against 1-NN using Euclidean and DTW distances.

Table 9 – Mean and standard variation of classification errors for each algorithm, and also the number of times they achieved a better, equal, and worst error than 1NN-ED and 1NN-DTW.

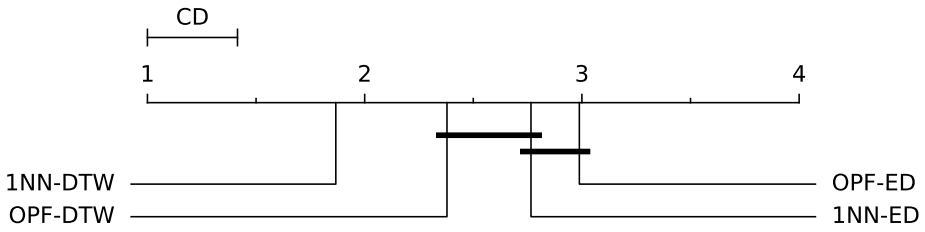
Algorithm	Mean (μ)	Stdev (σ)	Wins/Draws/Losses against 1NN-ED	Wins/Draws/Losses against 1NN-DTW
OPF-ED	0.3045	0.2023	37/25/66	35/7/86
OPF-DTW	0.2545	0.1908	85/7/36	14/36/78
1NN-ED	0.3138	0.2031	0/128/0	38/4/86
1NN-DTW	0.2508	0.1904	86/4/38	0/128/0

Source: The authors.

In Figure 4 and Table 9, we can see that 1NN and OPF obtained similar distribution and mean error rates in both Euclidean and DTW distances. However, 1NN-DTW still outperforms the OPF classifiers, considering the error rates. OPF-ED at least draws with 1NN-DTW in only 32.8% of the dataset. Furthermore, OPF-DTW at least draws with 1NN-DTW in only 39.1% of the datasets.

To help analyze and not do any arbitrary conclusions, we performed a Nemenyi test with these algorithms, which makes pairwise tests of the performances of each classifier for comparing them over multiple data sets (GARCIA; HERRERA, 2008). Figure 5 presents the critical difference diagram obtained.

Figure 5 – Critical distance diagram formed by classification errors. Algorithms that are not statistically different from others are connected. The classifiers are positioned on the axis according to their average rank, from the lowest to the highest rank (most important to least important).



Source: The authors.

Even though 1-NN with DTW distance is better in most cases, the OPF Classifier performs better in some datasets, being statistically comparable to 1-NN with Euclidean distance.

As OPF requires more space than 1-NN in the training step (the adjacency matrix used is quadratic in the number of series), those results are insufficient to justify its use for time series classification in a general sense. In Section 3.6, we mention the main limitations found in this work, and we suggest possible future approaches.

3.6 Difficulties, limitations and future works

The main difficulties seen in this project were the time needed to run the tests using the DTW distance and some limitations of the OPF in its traditional supervised form. During the time given to complete this work (of around 3 months), we focused on developing the algorithms and improving the distance metrics but didn't have time to do more improvements on the OPF itself.

Another possible limitation was that we used error rates as a guideline for analyzing the results because the ground truth from Dau *et al.* (2018) was based on error rates. However, as some datasets are not balanced, the accuracy paradox (ZHU; DAVIDSON, 2007) shows that other scores may be more adequate for this task, for example, the F1 Score.

Future works may focus on trying time series classification using already studied variations of the OPF, such as using a k-NN graph instead of a complete graph in the training stage or combining OPF-based classifiers trained with disjoint training subsets (PONTI; PAPA, 2011). Another idea is to use OPF inside embeddings of classifiers, as it achieved better results than 1-NN-DTW in some cases. The best-accuracy algorithm known in literature is the HIVE-COTE 2.0 (MIDDLEHURST *et al.*, 2021), a heterogeneous meta-ensemble formed by different classifiers from multiple domains, including distance-based algorithms. Middlehurst *et al.* (2021) show that combining various distance-based classifiers achieves better results than any of them individually. Therefore, it can be interesting to analyze how OPF contributes to this ensemble. Besides that, parallel and distributed processing can also be explored to save execution time.

4 CONCLUSION

4.1 Project contributions to the student

The project contributed to me because it was an excellent way to reinforce part of the knowledge passed on in the undergraduate course as algorithms and Graph applications. Furthermore, it was a great way to start studying *time series* and become aware of its techniques and methodologies because I had the opportunity to do lots of experiments and coding.

4.2 Relationship between the undergraduate course and the project

The undergraduate course is heavily connected with the developed project. In particular, one can highlight the subjects of Algorithms and Data Structures, Computational Modeling in Graphs, and Advanced Algorithms and Applications. These subjects present a wide range of knowledge and learning related to data structures and algorithm development. Overall, the undergraduate course provides a solid theoretical foundation in Computer Science.

Another fundamental point of the course was the opportunities for teaching assistance and research development, which improved my ability to communicate ideas and solutions. In addition, the presence of study and extension groups further complements the contents of the bachelor's degree.

4.3 Considerations about the undergraduate course

Overall, the Bachelor of Computer Science at the Instituto de Ciências Matemáticas e de Computação of the Universidade de São Paulo was impressive and exceeded my initial expectations. The course presents positive and negative elements. Next, we discuss the main aspects of it.

4.3.1 Positive aspects

There are plenty of positive aspects to present. The main ones are, first and foremost, the outstanding infrastructure of the Institute and the knowledge and kindness of most professors and employees. Second, the wide availability of social projects and extension groups for different areas, where students develop various personal skills and gain a broader view of research and work possibilities, establishing valuable connections.

4.3.2 Negative aspects

There are also a few negative points to consider. Beyond developing a system to review the courses and professors (with the *"Sistema de Avaliação de Disciplinas"*), those reviews also need to be considered. Also, one can point out the recurrent lack of interest by some professors regarding teaching and a few unpleasant in-class situations, such as rudeness.

REFERENCES

AFONSO, L.; PEDRONETTE, D.; PAPA, J. Improving optimum-path forest classification using unsupervised manifold learning. *In*: **International Conference on Pattern Recognition**. Beijing, China: IEEE, 2018. p. 560–565.

AMORIM, W. P. *et al.* Improving semi-supervised learning through optimum connectivity. **Pattern Recognition**, v. 60, p. 72–85, 2016. ISSN 0031-3203. Available at: https://www.sciencedirect.com/science/article/pii/S0031320316300668.

CAPPABIANCO, F.; FALCãO, A.; ROCHA, L. Clustering by optimum path forest and its application to automatic gm/wm classification in mr-t1 images of the brain. *In*: **2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Proceedings, ISBI**. Paris, France: IEEE, 2008. p. 428–431.

CORMEN, T. H. *et al.* **Introduction to algorithms**. 2. ed. Cambridge, Massachusetts: The MIT Press, 2001.

DAU, H. A. *et al.* **The UCR Time Series Classification Archive**. 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

FALCAO, A. X.; PAPA, J. **Optimum-Path Forest**. 1. ed. Cambridge, Massachusetts: Academic Press, 2022.

FALCãO, A.; STOLFI, J.; LOTUFO, R. The image foresting transform: Theory, algorithms, and applications. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, v. 26, p. 19– 29, 02 2004.

GARCIA, S.; HERRERA, F. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. **Journal of Machine Learning Research - JMLR**, v. 9, 12 2008.

HOLDER, C.; MIDDLEHURST, M.; BAGNALL, A. A review and evaluation of elastic distance functions for time series clustering. 2022.

KEOGH, E.; KASETTY, S. On the need for time series data mining benchmarks. **Data Mining and Knowledge Discovery**, 2003.

MIDDLEHURST, M. *et al.* Hive-cote 2.0: a new meta ensemble for time series classification. **Machine Learning**, v. 110, n. 11, p. 3211–3243, Dec 2021. ISSN 1573-0565. Available at: https://doi.org/10.1007/s10994-021-06057-9.

MINEA, M. **Lecture 8: Minimum Spanning Trees**. 2018. https://people.cs.umass.edu/~marius/class/cs311-fa18/lec8-nup.pdf. Accessed: 2022–10-20.

PAPA, J.; CAPPABIANCO, F.; FALCãO, A. Optimizing optimum-path forest classification for huge datasets. *In*: **2010 20th International Conference on Pattern Recognition**. Istanbul, Turkey: IEEE, 2010. p. 4162–4165.

PAPA, J.; FALCãO, A. A new variant of the optimum-path forest classifier. *In*: **Lecture Notes in Computer Science (LNIP)**. Berlin, Heidelberg: Springer, 2008. p. 935–944. ISBN 978-3-540-89638-8.

PAPA, J. *et al.* Efficient supervised optimum-path forest classification for large datasets. **Pattern Recognition**, v. 45, p. 512–520, 01 2012.

PAPA, J. *et al.* Design of robust pattern classifiers based on optimum-path forests. p. 10–13, 11 2007.

PAPA, J.; FALCãO, A.; SUZUKI, C. Supervised pattern classification based on optimum-path forest. **International Journal of Imaging Systems and Technology**, v. 19, p. 120 – 131, 06 2009.

PAPA, J. P.; FALCÃO, A. X. A learning algorithm for the optimum-path forest classifier. *In*: TORSELLO, A.; ESCOLANO, F.; BRUN, L. (ed.). **Graph-Based Representations in Pattern Recognition**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 195–204. ISBN 978-3-642-02124-4.

PONTI, M. P.; PAPA, J. P. Improving accuracy and speed of optimum-path forest classifier using combination of disjoint training subsets. *In*: SANSONE, C.; KITTLER, J.; ROLI, F. (ed.). **Multiple Classifier Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 237–248. ISBN 978-3-642-21557-5.

RAKTHANMANON, T. *et al.* Searching and mining trillions of time series subsequences under dynamic time warping. *In*: . New York, NY, USA: Association for Computing Machinery, 2012. (KDD '12), p. 262–270. ISBN 9781450314626. Available at: https://doi.org/10.1145/2339530.2339576.

ROCHA, L.; CAPPABIANCO, F.; FALCãO, A. Data clustering as an optimum-path forest problem with applications in image analysis. **International Journal of Imaging Systems and Technology**, v. 19, p. 50 – 68, 06 2009.

SILVA, D.; BATISTA, G. Speeding up all-pairwise dynamic time warping matrix calculation. p. 837–845, 06 2016.

SILVA, D. F. **Large scale similarity-based time series mining**. 2017. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, São Carlos, 2017.

SOUZA, R. W. R. de *et al.* A novel approach for optimum-path forest classification using fuzzy logic. **IEEE Transactions on Fuzzy Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 28, n. 12, p. 3076–3086, dec 2020. Available at: https://doi.org/10.1109%2Ftfuzz.2019.2949771.

TARJAN, R. **Data Structures and Network Algorithms**. Philadelphia, Pa: Society for Industrial and Applied Mathematics, 1983. (CBMS-NSF Regional Conference Series in Applied Mathematics). ISBN 9780898711875.

WANG, X. *et al.* Experimental comparison of representation methods and distance measures for time series data. **Data Mining and Knowledge Discovery**, v. 26, 1 2013.

ZHU, X.; DAVIDSON, I. Knowledge discovery and data mining: Challenges and realities. 01 2007.

APPENDIX

APPENDIX A – CLASSIFICATION RESULTS

In this appendix, in Table 10 (split in 3 pages), we present the results obtained by the OPF classifier using Euclidean distance and DTW distance. We also show the 1-NN errors for comparison purposes.

Table 10 – Classification error of each algorithm (1NN-ED, OPF-ED, 1NN-DTW, and OPF-DTW) for all datasets from UCR Archive (DAU *et al.*, 2018), sorted by name.

Name	1NN-ED	OPF-ED	1NN-DTW	OPF-DTW
ACSF1	0.4600	0.4600	0.3700	0.3800
Adiac	0.3887	0.3964	0.3964	0.4041
AllGestureWiimoteX	0.4843	0.4757	0.2343	0.2543
AllGestureWiimoteY	0.4314	0.4743	0.2800	0.2857
AllGestureWiimoteZ	0.5457	0.5514	0.3114	0.3186
ArrowHead	0.2000	0.2114	0.2800	0.2857
Beef	0.3333	0.3333	0.3667	0.3667
BeetleFly	0.2500	0.2500	0.3000	0.2500
BirdChicken	0.4500	0.4500	0.2500	0.2500
BME	0.1667	0.1733	0.0067	0.0067
Car	0.2667	0.2667	0.2500	0.2667
CBF	0.1478	0.2167	0.0056	0.0089
Chinatown	0.0466	0.0583	0.0262	0.0350
ChlorineConcentration	0.3500	0.3555	0.3518	0.3607
CinCECGTorso	0.1029	0.1051	0.2768	0.2768
Coffee	0.0000	0.0000	0.0000	0.0000
Computers	0.4240	0.4320	0.4280	0.4520
CricketX	0.4231	0.4436	0.2205	0.2231
CricketY	0.4333	0.4436	0.2333	0.2487
CricketZ	0.4128	0.4231	0.2308	0.2385
Crop	0.2883	0.2766	0.2979	0.3027
DiatomSizeReduction	0.0654	0.0654	0.0359	0.0359
DistalPhalanxOutlineAgeGroup	0.3741	0.3741	0.2302	0.2230
DistalPhalanxOutlineCorrect	0.2826	0.2899	0.2826	0.2826
DistalPhalanxTW	0.3669	0.3813	0.4101	0.4101
DodgerLoopDay	0.4500	0.4125	0.5125	0.5125
DodgerLoopGame	0.1159	0.1522	0.0942	0.1014
DodgerLoopWeekend	0.0145	0.0145	0.0290	0.0290
Earthquakes	0.2878	0.3094	0.3165	0.3237
ECG200	0.1200	0.1100	0.1700	0.1700
ECG5000	0.0751	0.0796	0.0751	0.0793
ECGFiveDays	0.2033	0.2033	0.2079	0.2079
ElectricDevices	0.4492	0.4557	0.3691	0.3779
EOGHorizontalSignal	0.5829	0.5801	0.4696	0.4862
EOGVerticalSignal	0.5580	0.6436	0.5166	0.5166
EthanolLevel	0.7260	0.7260	0.7280	0.7280

Name	1NN-ED	OPF-ED	1NN-DTW	OPF-DTW
FaceAll	0.2864	0.3006	0.2142	0.2166
FaceFour	0.2159	0.2386	0.1705	0.1818
FacesUCR	0.2307	0.2463	0.0839	0.0927
FiftyWords	0.3692	0.3736	0.2440	0.2462
Fish	0.2171	0.2171	0.1714	0.1771
FordA	0.3348	0.3409	0.4409	0.4447
FordB	0.3938	0.3975	0.3741	0.3815
FreezerRegularTrain	0.1951	0.1933	0.0993	0.1028
FreezerSmallTrain	0.3242	0.3242	0.2407	0.2407
Fungi	0.1774	0.1613	0.1452	0.1452
GestureMidAirD1	0.4231	0.5154	0.3923	0.3769
GestureMidAirD2	0.5077	0.5077	0.5308	0.5385
GestureMidAirD3	0.6538	0.7000	0.7231	0.7385
GesturePebbleZ1	0.2674	0.3372	0.2326	0.2384
GesturePebbleZ2	0.3291	0.3671	0.2278	0.2342
GunPoint	0.0867	0.0800	0.0600	0.0533
GunPointAgeSpan	0.1013	0.0316	0.0032	0.0032
GunPointMaleVersusFemale	0.0253	0.0063	0.0032	0.0032
GunPointOldVersusYoung	0.0476	0.0000	0.0000	0.0000
Ham	0.4000	0.3905	0.5048	0.5048
HandOutlines	0.1378	0.1541	0.1189	0.1297
Haptics	0.6299	0.6234	0.6006	0.5974
Herring	0.4844	0.4688	0.4688	0.4531
HouseTwenty	0.3361	0.3193	0.1933	0.1933
InlineSkate	0.6582	0.6600	0.6018	0.6036
InsectEPGRegularTrain	0.3213	0.0000	0.0000	0.0000
InsectEPGSmallTrain	0.3373	0.0000	0.0000	0.0000
InsectWingbeatSound	0.4384	0.4495	0.5253	0.5303
ItalyPowerDemand	0.0447	0.0496	0.0457	0.0466
LargeKitchenAppliances	0.5067	0.5120	0.2747	0.2693
Lightning2	0.2459	0.2623	0.1311	0.1311
Lightning7	0.4247	0.4247	0.2603	0.2466
Mallat	0.0857	0.0900	0.0652	0.0691
Meat	0.0667	0.0667	0.0667	0.0667
MedicalImages	0.3158	0.3263	0.2553	0.2605
MelbournePedestrian	0.1525	0.0558	0.0791	0.0836
MiddlePhalanxOutlineAgeGroup	0.4805	0.4870	0.5000	0.4740
MiddlePhalanxOutlineCorrect	0.2337	0.2371	0.3024	0.3333
MiddlePhalanxTW	0.4870	0.4805	0.4935	0.4870
MixedShapesRegularTrain	0.1027	0.1068	0.1225	0.1307
MixedShapesSmallTrain	0.1645	0.1682	0.1835	0.1918
MoteStrain	0.1214	0.1198	0.1350	0.1238
NonInvasiveFetalECGThorax1	0.1710	0.1740	0.2097	0.2229
NonInvasiveFetalECGThorax2	0.1201	0.1293	0.1349	0.1440
OliveOil	0.1333	0.1333	0.1667	0.1333
OSULeaf	0.4793	0.4711	0.4091	0.4298
PhalangesOutlinesCorrect	0.2389	0.2459	0.2716	0.2844
Phoneme	0.8908	0.8914	0.7822	0.7838
PickupGestureWiimoteZ	0.4400	0.3800	0.3200	0.3200

Name	1NN-ED	OPF-ED	1NN-DTW	OPF-DTW
PigAirwayPressure	0.9423	0.9087	0.8077	0.8077
PigArtPressure	0.8750	0.7067	0.5144	0.5192
PigCVP	0.9183	0.8606	0.7115	0.7115
PLAID	0.4767	0.3277	0.0931	0.0987
Plane	0.0381	0.0381	0.0000	0.0000
PowerCons	0.0667	0.0222	0.0556	0.0500
ProximalPhalanxOutlineAgeGroup	0.2146	0.2293	0.1951	0.2000
ProximalPhalanxOutlineCorrect	0.1924	0.2027	0.2165	0.2234
ProximalPhalanxTW	0.2927	0.2976	0.2439	0.2537
RefrigerationDevices	0.6053	0.6000	0.5600	0.5707
Rock	0.1600	0.4000	0.4800	0.5000
ScreenType	0.6400	0.6373	0.6107	0.6133
SemgHandGenderCh2	0.2383	0.1017	0.0783	0.0817
SemgHandMovementCh2	0.6311	0.4067	0.2022	0.2044
SemgHandSubjectCh2	0.5956	0.2111	0.1200	0.1244
ShakeGestureWiimoteZ	0.4000	0.3600	0.1000	0.1000
ShapeletSim	0.4611	0.5000	0.3611	0.3778
ShapesAll	0.2483	0.2517	0.2117	0.2217
SmallKitchenAppliances	0.6587	0.6773	0.3387	0.3493
SmoothSubspace	0.0933	0.0533	0.0467	0.0533
SonyAIBORobotSurface1	0.3045	0.3411	0.2745	0.2795
SonyAIBORobotSurface2	0.1406	0.1406	0.1595	0.1868
StarLightCurves	0.1512	0.1554	0.0942	0.0992
Strawberry	0.0541	0.0622	0.0595	0.0649
SwedishLeaf	0.2112	0.2224	0.2000	0.2080
Symbols	0.1005	0.1005	0.0603	0.0603
SyntheticControl	0.1200	0.1200	0.0167	0.0233
ToeSegmentation1	0.3202	0.3070	0.2281	0.2675
ToeSegmentation2	0.1923	0.2077	0.1154	0.1231
Trace	0.2400	0.2400	0.0000	0.0000
TwoLeadECG	0.2529	0.2502	0.1027	0.1150
TwoPatterns	0.0932	0.1068	0.0002	0.0002
UMD	0.2361	0.2083	0.0208	0.0208
UWaveGestureLibraryAll	0.0519	0.0561	0.0511	0.0558
UWaveGestureLibraryX	0.2607	0.2621	0.2239	0.2298
UWaveGestureLibraryY	0.3384	0.3439	0.3004	0.3130
UWaveGestureLibraryZ	0.3504	0.3548	0.3230	0.3277
Wafer	0.0045	0.0047	0.0157	0.0165
Wine	0.3889	0.3889	0.4259	0.4259
WordSynonyms	0.3824	0.3871	0.2649	0.2680
Worms	0.5455	0.5455	0.4935	0.4935
WormsTwoClass	0.3896	0.3896	0.4286	0.4156
Yoga	0.1697	0.1777	0.1583	0.1633

Source: The authors and Dau *et al.* (2018)