Geometrical models for time series analysis

Àlex Pardo Fernández
Advisor: Oriol Pujol Vila PhD.
Matemàtica Aplicada i Anàlisi

Master in Artificial Intelligence
GEOMETRICAL MODELS FOR TIME SERIES ANALYSIS

Àlex Pardo Fernández

18 ECTS thesis submitted in partial fulfillment of a Magister Scientiarum degree in Artificial Intelligence

Advisor
Oriol Pujol Vila PhD.

Faculty Representative
Oriol Pujol Vila PhD.

Faculty of Mathematics
University of Barcelona
Barcelona, October 2015
Geometrical models for time series analysis
Geometrical models for T.S.
18 ECTS thesis submitted in partial fulfillment of a M.Sc. degree in Artificial Intelligence

Copyright © 2015 Álex Pardo Fernández
All rights reserved

Faculty of Mathematics
University of Barcelona
Gran Via, 585
08007 Barcelona, Barcelona
Spain

Telephone: 93 402 08 53

Bibliographic information:
Álex Pardo Fernández, 2015, Geometrical models for time series analysis, M.Sc. thesis, Faculty of Mathematics, University of Barcelona.

Keywords: Geometrical, Time series, Kernel, Ensemble, One-class, Multi-class

Barcelona, Spain, October 2015
Special thanks to my advisor Oriol Pujol. Thanks also to Sergio Escalera, Albert Clapés and all my family that contributed in one way or another in the development of this work. Also thanks to the projects that supported this work: MINECO RETOS (RTC-2014-2148-1), RECERCAIXA2011-REMEDII, IMSERSO2012-MEDIMINDER, and VICYLE (TIN2013-43478-P).
Abstract

Geometrical models are intuitive and allow us to easily understand how a data set is structured and which is the best way to classify it. On the other hand, one class problems present a totally different perspective from the classical classification approach to supervised machine learning. This kind of problems are specially challenging due to the fact that only positive data is used in order to train the model.

In this work, a new approach to one class classification is proposed using geometrical models. The proposal is based on the kernelization of a technique of Approximate Polytope Ensemble (APES) [1]. This method allows us to achieve state of the art results using an intuitive, simple and easy to be represented model, named Kernel Approximate Polytope Ensemble (KAPE).

In addition we propose an extension of the original APE, which will be used afterwards in KAPE, that allows to measure the confidence of the membership of a point to the model.

Finally we study the problem of gesture recognition using Inertial Movement Units (IMUs). To this end, a new dataset of actions and interactions is presented, recorded in the wild with elder people. In this work a study of the data given by a wearable sensor in the wrist of the users is presented, exposing the main problems and how can they be mitigated. KAPE is applied in this scenario of variable length gesture sequences with very promising results.
Contents

1 Introduction ................................................. 3
   1.1 Contributions ........................................ 4
   1.2 Publications ........................................ 5
   1.3 Structure ........................................... 5

2 Background ................................................. 6
   2.1 Approximate Polytope Ensembles ....................... 6
      2.1.1 Powerfulness of random projections .............. 6
      2.1.2 Approximate Polytope Ensembles (APE) .......... 6
      2.1.3 Non-convex extension ............................ 8
   2.2 Introduction to kernels ................................ 9
   2.3 Dynamic Time Warping ................................. 10
   2.4 Reference Methods for Data modeling and Classification 10
      2.4.1 Probability Density-based ....................... 11
      2.4.2 Distance-based ................................ 11
      2.4.3 Kernel-based ................................... 11

3 Kernel Approximate Polytope Ensembles ............... 13
   3.1 Core and Membership ................................. 13
   3.2 Kernel APE ........................................ 15
   3.3 Convex Hull enlargement/shrinking .................... 18
   3.4 Outlier selection ................................... 18
   3.5 Computational performance .......................... 19
   3.6 Classifying data with KAPE .......................... 19
   3.7 Working with sequences .............................. 20

4 Experiments and Results ................................. 21
   4.1 Behavior of the method ............................... 21
      4.1.1 Case study: Analysis of Ying-Yang dataset ... 24
   4.2 Datasets ............................................ 27
   4.3 Settings ............................................ 28
   4.4 Results ............................................. 29

5 Gesture recognition ....................................... 37
   5.1 Methodology ......................................... 37
   5.2 Data ................................................ 37
   5.3 Feature selection .................................... 40
   5.4 Settings ............................................ 45
   5.5 Results ............................................. 45

6 Conclusions and future work ......................... 49
   6.1 Conclusions .......................................... 49
   6.2 Future work ......................................... 50
1 Introduction

Pattern recognition is a field of machine learning that tackles the problem of finding the model that better explains a set of data. Essentially, for a given input, the system is able to predict the expected label after a training phase. For doing so, an algorithm or computational procedure is needed.

In this Master Thesis, the proposed method is a geometrical model, which is a branch of the pattern recognition field. Those algorithms form a discipline called computational geometry, which comprehends the problems related to the geometrical properties of a set of points in a $\mathbb{R}^n$ space. Some example problems are: closest pair of points, delaunay triangulation or convex hulls. In [2], authors propose a geometric framework for binary data classification using large-margin classifiers by applying two different geometrical models. The geometric interpretation of SVM classifier is used in [3] in order to propose a geometric algorithm for both separable case (using Convex Hulls) and non-separable cases (using Reduced Convex Hulls). Finally, [4] presents the Nearest Convex Hull classifier, a geometric approach to SVM that uses the distance to the convex hull representing a class as the decision function.

Some other articles make use of geometrical methods in order to improve other techniques. For instance, [5] uses the geometrical interpretation of Support Vector Machines and kernel functions in order to improve some algorithms (e.g. shape reconstruction).

One important property of computational geometry problems is their complexity in time. Most of them have efficient algorithms able to reduce complexity to $O(n \log n)$ [6], but it is a must tot say that these algorithms usually scale exponentially with the dimensionality ($d$) of the space. In our specific case, we explore methods based on the Convex Hull structure. The convex hull is the smallest convex polytope (or polygon in arbitrary dimensionality) that encloses a set of points. The computational complexity of this algorithms is $O(n \log n)$ in $\mathbb{R}^2$ [7].

Among all the problems pattern recognition deals with, we are going to focus on anomaly detection (also referred as one-class modeling or classification). This kind of problems have the particularity that we only have information of one of the classes (the one we have to model). Usually, in binary classification, information of both classes is available so finding the boundary can be defined as a minimization problem in which the error given by a certain classification function should be reduced. In the case of anomaly detection, the information available is related to the objective class. The common approach is to define some of the points as outliers (anomalies) and try to fit a model able to distinguish between inliers and outliers. The number of outliers can be viewed as a regularization factor. One-class problems arise when it is easy to obtain target data, but negative samples are impossible or difficult to collect. Examples of this problems are machine failure prediction (it is not an option to make machines to fail in every possible way) and banknote verification (there are numerous ways of reproduce them). In [8], a survey on one-class problems is presented, in which several methods are detailed. Authors in [9], also compare some methods for outlier detection. The work explained in [10], tackles the problem of high-dimensional spaces in distance-based classifiers and proposes the use of projections in order to find outliers in spaces in which distances are distorted by the curse of dimensionality. Finally, [11] presents a solution for one-class classification in the information retrieval context by using two different One-Class SVM approaches.

This master thesis follows the work of [11] in which the target class is modeled by means of combining Convex Hulls in several random projections. In particular we devise a kernelized version of the method that allows to implicitly and efficiently model non-convex problems using Approximate Polytope Ensembles (APE). Additionally, we explore a new formulation of the basic APE that is able to convey information about the confidence of the membership of a point to the model. Finally, we apply the
proposed method in a real problem of gesture recognition. In this problem, we explore the use of Kernel-APE in a real dataset of gestures in which, each of them is a multi-dimensional sequence with different length. Moreover, it is known to have a large intra and inter-class variability, which increases the complexity of the problem.

In order to be able to tackle this problem, we have to transform our one-class classifier into a multi-class one. An interesting point of the Convex Hull approach for tackling anomaly detection is that it is easy to apply it to multi-class classification. The extension to the multiclass case is done by using the notion of confidences (one of the contributions of this thesis) and selecting the model with maximum confidence. Then, by using confidences given by the proposed improvements, we are able to define a hyperplane separating the classes.

The method used to deal with gesture recognition tasks totally depends on the data input. Basically, there are two trends: images and wearable signal. In the first case, methods such as optical-flow [12] are applied. The second case, the one we deal with, is typically solved by using Hidden Markov Models (HMM) [13], or even with Artificial Neural Networks (ANN) [14]. In this work we use a the proposed kernelized version of APE with a dynamic time warping kernel to deal with sequence alignment and gesture modeling.

1.1 Contributions

This work is based on APE and tries to improve it by adding some features to the method. Next, the most important contributions are detailed.

- In this work we introduce the notion of confidence in the original APE implementation, including the notion of membership to the class. This will be crucial for assessing if a point is member of the class.
- The most important contribution in this work is the application of Kernels to the original APE. This will increase the ability to fit non-convex sets without increasing the computational cost of the method. Also, Kernels will empower the method to use any other types of structured input data.
- As it has been said before, the notion of confidence enables the method to model multi-class problems without any modification of the method. It will be explained how it is approached.
- The one-class classifier will be tested against methods found in literature in order to be able to compare their performances.
- Finally, in order to demonstrate the powerfulness of the proposed modifications, it will be used for a gesture recognition task, using a multi-class approach with inertial movement signals as input.
1.2 Publications
Some of the work presented in this Master thesis has been included in an article presented in the European Conference on Complex Systems and published in Lecture Notes in Computer Science. It also has been presented to the International Journal of Computer Vision and is currently under review. Kernel APE is planned to be published, using the work presented in this Master Thesis.

1.3 Structure
The remainder of the work is structured as follows:

Chapter 2 introduces the background needed for understanding the main contributions of this thesis. First a brief description of the approximate polytope ensembles is given. Then the notion of kernels is explained with special emphasis on kernel methods in the primal. Finally, a brief description of the reference models of the state-of-the-art is given.

In Chapter 3 the proposal is detailed. In this section, the contributions are explained and formulated, giving the reader all the necessary intuitions to understand the method that is being proposed. In particular, the notion of membership is explained, then, once all the prior modifications are introduced, kernel application is detailed. Finally, some additional improvements are explained, including the multi-class extension.

Experiments and results are explained in Chapter 4. First, the behavior of the method is described in terms of its parameters, giving an intuitive overview on how it works. Then, the used datasets are listed. Finally, the results of the performed experiments are presented, including an analysis of the outcomes and some insights about them.

Chapter 5 is devoted to the explanation of SARQuavitae Claret dataset and its use for gesture recognition in the multi-class approach of KAPE.

Finally, in Chapter 6 the work is concluded with a review of the proposed method and some details and considerations. Also the future of this project is analyzed.
2 Background

In this section, all the basic concepts needed to understand the proposed method and the experiments are introduced.

2.1 Approximate Polytope Ensembles

2.1.1 Powerfulness of random projections

Random projections has been proved to be very powerful tools for dimensionality reduction with metric guarantees. These guarantees are stated in the Johnson and Lindestrauss lemma [15]. This lemma states the degree of degradation in pair-wise data distances when these are embedded in a low dimensional space. The intuitive idea behind this lemma is that using random projections, we would obtain a projection of the original points that would nearly preserve the pairwise distances. The lemma, detailed in (2.1.1), states that one can embed a set of \( n \) points into a \( k \)-dimensional space with \( 1 + \epsilon \) distortion.

\[
k = O(\epsilon^{-2} \log n)
\]

\[
f : \mathbb{R}^d \rightarrow \mathbb{R}^k
\]

\[
(1 - \epsilon) \| u - v \|^2 \leq \| f(u) - f(v) \|^2 \leq (1 + \epsilon) \| u - v \|^2
\]

Then, by using random Euclidean projections, we would obtain several representations in a lower-dimensional space, nearly preserving the original distances. This allows to increase the efficiency of some methods.

Approximate Polytope Ensembles use many projections in a two-dimensional space. Even though Johnson-Lindestrauss lemma states that embeddings in two dimensions are prone to large distortions, because APE combine many different projections the effective dimensionality of the methods is much larger, mitigating the distortion effects.

2.1.2 Approximate Polytope Ensembles (APE)

The first implementation of the Approximate Polytope Ensemble [1], proposes a method for modeling data using a geometric model, in this case a polytope.

As explained in the article, APE is based on computing convex hulls in many different 2-D random projections and aggregating the results into a final decision function. Then, the convex hull of this set of points is computed for every projection. Parameter \( \tau \) controls the number of projections used. In Algorithm [\ref{algorithm1}] the training procedure is detailed. In this, data is projected into a 2-dimensional space and finally the convex hull is computed. The model generated is characterized by a set of projections, needed
in the testing phase, and the vertices that define the convex hull.

**Data:** $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d, \tau$

**Result:** A model $M$ composed by $\tau$ projection matrices and their corresponding Convex Hulls

$M = \emptyset$

for $t = 1 \ldots \tau$ do

\[
P_t \sim N(0, 1) \quad /* \text{Generate a random projection from a normal distribution} */
\]

\[
X_t = \{P_t x | x \in X\} \quad /* \text{Project the training points} */
\]

\[
v_t = \text{conv}(X_t) \quad /* \text{Obtain the vertices of the Convex Hull} */
\]

\[
M = M \cup (P_t, v_t) \quad /* \text{Store the projection and the vertices} */
\]

end

**Algorithm 1:** APE training algorithm

In order to test if a point is part of the modeled class, its membership is computed for all the convex hulls. If it is outside one of them, it is not a member of the class. Algorithm 2 details this procedure, in which for a new test point, it is projected using every projection matrix and then, it is tested if it lays inside the polytope defined by the set of vertices. In Figure 1, one can see the case in which a test point that is outside the model in the original space, lays inside two of the three projections. Since it is not inside all the polytopes, it is said to be an outlier for this model. On the contrary, if it lays inside all the polytopes, we can assure that is part of the learned model.

**Data:** Test point $x \in \mathbb{R}^d$, the Model $M$

**Result:** Result $\in \{\text{INSIDE}, \text{OUTSIDE}\}$

Result = INSIDE

for $t = 1 \ldots \tau$ do

\[
x_t = P_t x \quad /* \text{Project the test point} */
\]

if $x_t \notin \text{conv}(v_t)$ then

\[
\text{Result} = \text{OUTSIDE}
\]

Break

end

end

**Algorithm 2:** APE testing algorithm

If we have a large $\tau$, we are able to represent the full convex hull of the original space by using the projections computed. Moreover, the fact of computing the convex hulls on a lower-dimensional space increases the computational performance, since the computation in $\mathbb{R}^2$ has a cost of $O(n \log n)$ in the worst case.

Many algorithms can be used in order to check if a point lays inside a polygon, in this case they use the Ray Casting algorithm [16]. The intuition behind it is the following: for a certain polygon and a sample point, we count the number of times, a ray starting at the point and going in any direction, intersects an edge of the polygon. In this case, if it intersects an odd number of times, the point is inside. If the intersections are even, it is outside.

Because all data points in the training set are modeled, if the set contains outliers, APE can be prone to have less than optimal performance (it is also modeling the outliers). In order to avoid it, they define an enlargement/shrinking parameter $\alpha$ that expands or contracts the convex hull in all of its projections.

This implementation is pretty efficient and fits well to convex data, but it is not able

---

1Figure extracted from [1]
to deal with non-convex points. In the following section an extension to non-convex sets is explained.

2.1.3 Non-convex extension

The method explained in the previous section performs well for a lot of simple problems, specially if they have a convex shape. In the case of non-convex shapes (e.g. a Banana-shaped dataset), the algorithm will model empty spaces given by the non-convexity property of the data.

In order to avoid this situation, an extension to APE had been defined\cite{1}: non-convex APE (N-APE). The key idea is to divide the non-convex shape into a set of convex shapes, modeling the data locally. In Figure 2, the procedure is illustrated: first of all, set all the point as not visited. Then, choose a random point and compute the convex hull of all the point in a radius \( r \) (a). All the points that lay inside the convex hull are marked as visited. The vertices of the computed polytope are set as candidate centers for further steps. Then, a new center is selected from the list of candidate centers and the procedure shown in (a) is repeated. The second iteration of this process is shown in (b). The procedure is repeated until all the points in the set are covered by the polytopes.

In order to check if a point is part of the target class, it is required to lay inside, at least, one of the convex hulls (not in all of them as in APE).

By using this extension any dataset (convex or non-convex) could be modeled. However, the number of convex hulls needed for modeling any dataset is much higher\footnote{Figure extracted from \cite{1}}.
than in the original implementation, adding a large computational cost to the model generation and testing.

2.2 Introduction to kernels

A Kernel $\kappa$ is a transformation function that maps points to a inner-product space. That is, for a given pair of points $x, z$, obtains their inner product in a high-dimensional space, without the need of computing the transformation $\phi$ to that space $F$ as it is shown in Equation \ref{eq2}.

$$\kappa(x, z) = \langle \phi(x), \phi(z) \rangle$$

$$\phi : x \in \mathbb{R}^n \mapsto \phi(x) \in F$$

In literature, kernels are mostly used to tackle with non-linearly separable datasets. The most well known technique using them is Support Vector Machines (SVM), which is a linear classifier that uses kernels to deal with non-linearly separable datasets.

Kernel functions provide the inner product of input data in a certain space. The mathematical properties make them interesting from both a geometrical and a computational point of view. From the first, we have that the kernel of a set of points represents their pairwise distances, which means that every point is characterized by the similarity to all the set, in other words, if we know that a class is defined as a set of points that are similar, we have that those points corresponding to the same class in a kernel space will be close in some of their components (those representing the similarity to the points of the same class). From a computational point of view, the fact of having the inner products of the points in a large dimensional space without the need of projecting the points to that space implies a large efficiency improvement, even enabling us to work with infinite-dimensional spaces.

The Kernel is constructed by defining a convex function that maps points into a Hilbert space $\mathcal{H}$ \[14\].

Usually, Kernels substitute an inner product in a dual form. This is the famous Kernel Trick \[18\]. However other successful approaches, similar to the one presented in this work, have been described in literature, defining, among others, the formulation of SVM in the primal form \[19\].

Kernel functions are commonly used in literature. It is worth mentioning its use for unsupervised tasks in which the target is to reduce the dimensionality of the data. For instance, Kernel-Principal Component Analysis \[20\] takes profit of the non-linearity introduced by the kernel function in order to reduce the dimensions of the dataset while improving the separability of the different classes.

Several types of kernels can be used. Most used kernels are:

- Polynomial: $k(x, y) = (\alpha x^T y + c)^d$
- Gaussian: $k(x, y) = \exp(-\frac{||x-y||^2}{2\sigma^2})$
- Sigmoid: $k(x, y) = \tanh(\alpha x^T y + c)^d$

Other types of kernels can be found in literature, such as Exponential, Laplacian or ANOVA kernels are also RBF-kernels. In general it can be shown that the exponential function of a negative distance is always a kernel. For instance, an exponential-kernel uses a L1 norm instead of a L2 as the Gaussian one does. Taking profit of this property, we would change the distance metric in order to be able to compute a kernel for sequences.
2.3 Dynamic Time Warping

In order to work with sequences, we use Dynamic Time Warping (DTW) as a distance measure.

DTW is a dynamic programming algorithm that computes the cost of the alignment between two sequences, this is, which will be the cost of transforming one sequence into the other. This is a good indicator of their separation. In the case were two sequences are very similar, it would be easy to transform one into the other.

As a dynamic programming method, it uses a matrix to reach the final result. Let $D \in \mathbb{R}^{m \times n}$ be the dynamic programming matrix, where $m$ is the length of sequences $x$ and $n$ the length of sequence $y$. then, in order to compute each cell of the matrix, we need the three upper-left neighbors (North, West and North-West) to have a value. Essentially, the value of a cell follows the following formulation:

$$D(i,j) = \min \{D(i - 1, j - 1), D(i - 1, j), D(i, j - 1)\} + \|x_i - y_j\|_{\delta}$$

In Figure 3, one can see an example of the alignment of two words (SPEECH and SsPEEhH). In this case, if we use a binary distance being $d = 0$ if letters match and $d = 1$ if not, the alignment will have a cost of 3, which match with the words, SsPEEhH has two different letters and an extra one so in order to transform it into SPEECH, we have to perform two substitutions and one deletion.

![Figure 3: Dynamic Time Warping example on word alignment](image)

Then, the cost of aligning both gestures is the value at $D(m, n)$, normalized by the length of the sequence (in order to avoid a weighting due to sequence lengths).

However, DTW is not a metric since it does not fulfills the symmetry property $d(x, y) \neq d(y, x)$. Note that this fact will have implications on its application to Kernel-APE.

2.4 Reference Methods for Data modeling and Classification

In the State of the Art in One-class problems (also known as Novelty Detection) and also in Classification, there is a large collection of well performing algorithms. An small set of representative methods, chosen by its proven usefulness, will be explained and used in further comparisons against the presented method in order to prove its efficiency.

The taxonomy has been extracted from the review presented in [8].
2.4.1 Probability Density-based

First, we introduce Probability Density-based methods which are a generative approach that tries to model the data by means of Probability Density Functions (PDF), representing the target data as a statistical distribution. We focus on parametric methods.

The density function learned might be thresholded in order to define a boundary which acts as a frontier between those points that are part of the underlaying probability distribution and those which are not.

Both Gaussian Models and Gaussian Mixture Models can be used in One-class and in Multi-class approaches. The latter implies treating each of the classes separately and then fusing the models.

Gaussian Models The simplest approach are Gaussian Models (GM). In this case, we suppose a Normal distribution in our data. Then, we model the class space with a single Gaussian by learning the parameter $\sigma$.

It is obvious that for in non-normally distributed sets, the model will not be representative enough and then, will lead us to misclassifications and low performance rates. Another disadvantage is that is not suitable for on-line learning since the $\sigma$ needs to be learned for each new sample in the system.

Even though this technique is not very common, it has been used [21] until the formulation of Gaussian Mixture Models.

Gaussian Mixture Models Gaussian Mixture Models (GMM) are an extension of GM in which the data is represented by a weighted mixture of Normal distributions. Basically, we set the number of Gaussians we want to represent our data and set one center (mean) for each of them, then, using an iterative process we tune the parameters until we reach a (sub) optimal point.

Then, we are able to set a threshold in order to define a boundary for our class. This models are widely used [22] because of its simplicity and usefulness.

2.4.2 Distance-based

This type of algorithms use distance metrics in order to classify each of the points.

K-Nearest Neighbors K-Nearest Neighbors (K-NN) is based in the assumption that the difference between outlier and non-outlier points is that the former are far from the latter. So, if we sample a point and it is far from the labeled ones, we can say it is an outlier. To do so we have to define a minimum radius at which we can say points laying there are outliers.

Setting the minimum distance and the percentage of nearest neighbors are critical parameters with a difficult computation.

This simple approach is used [23] because of the lack of learning in all the process and the good results it could achieve.

In the case of multi-class classification, the classical K-NN is used, in which for a new sample, we get the label of the K nearest neighbors and set the new label using a certain policy (e.g. voting)

2.4.3 Kernel-based

Kernel-based methods can be interpreted as a non-linear projection (see Section 2.2) in order to make data linearly separable.
**Support Vector Machines**  Support Vector Machines (SVM) is a linear classifier that takes profit of a Kernel in order to separate linearly problems that are not linearly separable in the original space. In the case of One-class problems, a boundary is computed by considering some of the training points as outliers, in such a way that most of the inliers will be well classified.

This technique is well known for giving competitive results and is applied to several areas. In the case of novelty detection its use [24] is based on its usefulness is highlighted by the fact of maximizing the margin between classes and the use of kernels.

On the other hand we have multi-class (i.e. two classes) approach in which a soft margin is maximized in order to have the minimum amount of misclassified points.
3 Kernel Approximate Polytope Ensembles

Approximate Polytope Ensembles as described in the original implementation is a useful tool for modeling some one-class problems. However, it does not work with non-convex sets. The N-APE extension is able to deal with this problem but highly increasing the computational cost of the method. In our proposal, the kernelized version of APE will be able to deal with non-convexities while preserving the computational costs closer to the APE implementation.

A CH is the part of the space formed by a convex combination of points, which is a positive weighted linear combination of points where the sum of the weights is one:

\[ CH(X) = \sum_{i}^{N} \alpha_i x_i ; \ x_i \in X \]
\[ \alpha_i \geq 0 \ , \ \forall i \in [0,N] \]
\[ \sum_{i}^{N} \alpha_i = 1 \]

As it has been explained in Section 2, the set of convex hulls from randomly-projected points forms an APE. By using them, we create a region in which the points of the target class lay. Next, we define some new concepts related to APEs that will lead us to the final formulation of the Kernel-APE.

3.1 Core and Membership

Defining the concept of Core. In APE, the membership of a certain point to the class is binary, so it is part or not, but there is no degree of membership. In this original implementation, all the points of the set are used in order to construct the polytopes, for this reason it makes sense to define this type of membership: if the target class is modeled using all the data available, it is sensible to suppose that any new point will behave the same way. However, this approach does not take into account the case in which an actual member point is out of some projections due to some distortions generated by the projection. Moreover, by considering all the points in the training set we can end with a bad model in the case the positive set contains outliers.

In order to increase the robustness of methods in front of outliers, it is common in literature to use some re-sampling techniques, such as bagging, in which training data is a randomly chosen (uniformly and with replacement) subset of the input points. This kind of techniques are used, for instance, in decision trees and are known to control overfitting while improving stability and accuracy.

In our case, we would implement a similar technique for each of the polytopes. By doing so we expect outliers to be selected in few projections.

It is important to notice that in the original bagging algorithm, samples are taken uniformly and with replacement. In our case, since it can generate strange situations (when computing the polytope of a set with repeated points), replacement is not allowed. Thus, instead of using a bootstrapping technique we will use a resampling technique.

By including this modifications in the construction of the polytopes, we get that all the convex hulls will be built on a different set of points, in other words, every point will be found inside a certain percentage of polytopes. With this consideration in mind, emerges the concept of Core.

Then, given a set of points and a set of convex hulls constructed from those, we define the core as the space in which all the convex hulls, intersect.
Let $X$ be the set of point we want to model. We consider $\eta$ random subsets of $X$, $\Pi = \{\Pi_0, \Pi_1, ..., \Pi_\eta\}$, where $\Pi_i \subset X$. The core under the set of random samplings $\Pi$, $\Theta_\Pi$ is defined as the intersection of all the convex hulls of the sets, i.e.

$$\Theta = \bigcap_{i=1}^{\eta} CH(\Pi_i)$$

The concept of core, defines the region of the space where all convex hulls agree. Complementary to this definition we may define the concept of membership of a point to the model as the proportion of $CH$ where the point lies inside the hull. Let $z$ be an unseen point, we define the confidence of it of being part of the class of $X$ as:

$$\nu(z) = \frac{\sum_{i=0}^{\eta} (1 \mid z \in CH(\Pi_i))}{\eta}$$

In Figure 4 one can observe how the core is constructed, the cores is only the part inside the membership= 1 contour. Notice the fact that it is a region in the original space that is represented by its different projections. Given that the intersection of the convex hulls is difficult to compute since are defined in different spaces, the intersection is computed by assessing the membership of each of the points and generating the polytopes using the subset with full membership. The core is denoted in each of the polytopes by a red polygon inside it.

**Figure 4:** Figure showing the membership of the space and the core (membership = 1)

It is important to understand that this changes the basic decision function of the APE. In the original formulation of APE a point is considered to be an outlier if it is outside of at least one convex hull, and a member of the model if it can not be rejected. In our new formulation this definition no longer holds. The membership to the model is defined by an additional parameter, a threshold on the confidence of belonging to the model. This parameter has to be tuned.
3.2 Kernel APE

The non-convex extension explained on Section 2.1.3 is a great improvement over APE, but it will only work with vectorial data. Moreover, for each space modeled a full APE is computed and the number of APEs needed increase when reducing the radius parameter; these APEs will have large intersections and this implies a low efficiency of the algorithm. In order to avoid these limitations, we propose the use of a kernelized version of the APE.

A convex combination of a set of points \( X \) is defined as

\[
CS(X) = \sum_{i}^{N} \alpha_i x_i
\]  

(3)

with:

\[
\alpha_i \geq 0 \quad \forall i
\]

\[
\sum_{i}^{N} \alpha_i = 1
\]

The convex hull for \( CS \) is the smallest polytope formed by the points in the vertices that bounds the convex set.

We want to use a certain space of large-dimensionality in order to better represent the space. Let us consider a transformation \( \phi(x) : R^d \rightarrow R^{d'} \) of the original space into an arbitrary space. The convex combination of the points in this new space is defined as follows,

\[
CS(X) = \sum_{i}^{N} \alpha_i \phi(x_i)
\]  

(4)

subject to \( \alpha \geq 0 \sum \alpha_i = 1 \)

In the transformed space any point inside the convex hull can be represented by the convex combination of the vertices. Let us define the reconstruction error of a point by a convex combination of points as

\[
\epsilon = (\phi(x) - \sum_{i}^{N} \alpha_i \phi(x_i))^2
\]

subject to \( \alpha_i \geq 0; \sum \alpha_i = 1 \)  

(5)

For any point inside the convex hull the reconstruction error \( \epsilon = 0 \). We can rewrite Equation (6) as:

\[
\phi(x)\phi(x) - 2 \sum_{i}^{N} \alpha_i \phi(x_i)\phi(x) + \sum_{i}^{N} \alpha_i \phi(x_i) \sum_{j}^{N} \alpha_j \phi(x_j) = \epsilon
\]

(6)

By definition, the kernel \( k(x,y) \) can be written as the inner product \( <\phi(x), \phi(y)> \). This yields the following equation,

\[
k(x, x) - 2 \sum_{i}^{N} \alpha_i k(x_i, x) + \sum_{i,j}^{N} \alpha_i \alpha_j k(x_i, x_j) = \epsilon
\]  

(7)
Given that \( k(x, x) = 1 \) by definition and changing the last term into a vector-matrix form as follows,

\[
\sum_{i,j}^N \alpha_i \alpha_j k(x_i, x_j) = \alpha^T K \alpha
\]

Equation (7) can be rewritten as:

\[
1 + \alpha^T K \alpha - 2 \sum_i^N \alpha_i k(x_i, x) \quad (8)
\]

subject to \( \alpha_i \geq 0 \); \( \sum \alpha_i = 1 \)

Notice the only part of the equation that depends on the input data is \( \sum_i^N \alpha_i k(x_i, x) \).

Analogous to the standard convex hull, we can define the convex hull in the kernel space in order to assess the membership of a point to the interior of the convex hull. Note that although the computed hull is convex in the kernel space, it represents an arbitrary transformation of the input space and thus can represent non-convex shapes in the original space.

\[
CH(X) = \sum_i^N \alpha_i k(x_i, x)
\]

\[
\alpha_i \geq 0 \ , \ \forall i
\]

\[
\sum_i^N \alpha_i = 1
\]

Since we are working in the RKHS, K matrix is positive semidefinite; given that, K can be associated to a metric. We must assume that a class is defined as those points that are close to each other (at least in some dimensions), so using the kernel matrix K introduces the notion of distance between points into the working space. Recall the projection strategy used by the APE algorithm and shown in the following equation:

\[
X_t = \{ P_t x | x \in X \} \quad (10)
\]

In order to apply a kernel function \( K \) to the training points, we reformulate the projection process from the original APE shown in Equation (10) to the kernel version as in Equation (11) in which \( X_s \) is uniformly drawn from \( X \) with a probability \( \tau \) and \( K \) is applied to every point in the training set before projecting it into the lower dimensional space.

\[
X_t = \{ P_t K(x_i, x) | x_i \in X_s \}
\]

K \((x, y) = \exp(\gamma \|x - y\|^p) \quad (11)
\]

The kernel used is an exponential of a distance as stated in Equation (11). The parameter \( \gamma \) will control the coverage of the function, cutting off the value at a maximum distance. Choosing the optimal \( \gamma \) ensures a local view of the dataset, as N-APE does with the radius. Having a low coverage will result on \( K \approx I \), so the algorithm will over-fit on train data since it will not be able to generalize. On the contrary, if we set a large coverage, the kernel matrix will be uniform (all the points will be considered close due to the large range) and will under-fit data due to its incapability to represent the pattern.
An optimal $\gamma$ will lead us to a situation in which those points that are far from the population (i.e. outliers) will have a nearly-zero norm (the computation of the kernel will be zero if points are far from each other). This means that, when projecting into the random space in $\mathbb{R}^2$, we will have a point close to $(0,0)$. Following the same logic, a point that is close to a large number of points will have a norm close to 1. This means that when applying the random projection, it will lay in the borders of the convex hull computed in this space. The Kernel will generate a hypersphere (a convex set) in which, those points with zero norm will be the ones far from the population in the original set, and those with maximum norm, will tend to be centers of mass. The fact that every point with lower norm is mapped close to 0,0 is, in fact, a good property, since we only have to model the singularity generated by the definition of the kernel in order to get rid of this points. We define an exclusion zone in the center of the hull (close to 0,0) in which every point that lays inside is considered to be out of the convex hull.

However, in the One-class problem, only outliers will lay into this space. In the case of classification, it will be different. This problem will be tackled later.

By using the kernel approach, we reduce the complexity of the N-APE algorithm to the original APE plus a Kernel computation. Additionally, we add the confidence level defined in Section 3.1.

Data: $X = \{x_1, \ldots, x_n\} \in \mathbb{R}^d, \eta, p, \gamma, \tau$

Result: A model $M$ composed by $\tau$ projection matrices and their corresponding Convex Hulls and the kernel matrix $K$ for the given input data

$$M = \emptyset$$

$$K(x_i, x_j) = \exp(\gamma \|x_i - x_j\|^p) \forall x_i, x_j \in X; i \neq j$$

for $n = 1 \ldots \tau$

$K_s = \text{subsample}(K, \eta)$ /* Get the subset of points from the kernel */

$P_t \sim U(0, 1)$ /* Generate a random projection from a uniform distribution */

$X_p = \{P_t K_s(x_j, x) \mid x_j \in K_s\}$ /* Project the training points in the kernel */

$v = \text{conv}(X_p)$ /* Obtain the vertices of the Convex Hull */

$M = M \cup (P_t, v)$ /* Store the projection and the vertices */

end

Algorithm 3: K-APE training algorithm

As it is detailed in Algorithm 3, the procedure is not very different from the original APE formulation, but only the kernel function and the sub-sampling technique have been introduced.

The testing Algorithm 4 goes as follows: for a new point, we project it into the kernel space and count in how many hulls it lays inside. The greater this number is,
the larger will be the confidence of being part of the class.

**Data:** Test point \( x \in \mathbb{R}^d \), the Model \( M \), the kernel matrix \( K \)

**Result:** The membership of the point

\[
\text{Count} = 0
\]

\[
x_k = Kx /* Project the test point over the kernel */
\]

for \( t = 1 \ldots \tau \) do

\[
x_t = P_t x_k /* Project the test point over the kernel */
\]

if \( x_t \in \text{conv}(v_t) \) then

\[
\text{Count} += 1
\]

end

end

Membership = Count / \( \tau \)

**Algorithm 4:** K-APE testing algorithm

This Kernel approach enables us to use any input data (vectors, text, sequences, images, etc.) since the data is modeled from the kernel, increasing the usefulness of the method. The lower-dimensionality projections maintains the expressiveness of the model, as it has been demonstrated before, while reducing its complexity.

### 3.3 Convex Hull enlargement/shrinking

Even that in the Kernel-APE algorithm, it is possible to enlarge or shrink the polytopes as in the original APE, the introduction of the notion of core and consequently the membership, makes the enlargement/shrinking redundant. In the exposed approach, the membership acts as the shrinking parameter does. By selecting a certain threshold, we are reducing or increasing the coverage of the convex hull. The variation of this membership threshold, goes from the polytope given by the union of all the convex hulls, where points only have to lay in one of the polygons, until the intersection of all of them, where points have to lay in every convex hull. The proposed approach has a further benefit when compared with the original APE. The shrinking procedure is done by selecting a center of the hull, e.g. analytic center, center of mass, etc. The selection of the best center is still an open problem. The membership proposal avoids this problem.

### 3.4 Outlier selection

In the One-Class problem, we model a single class, so it is easy to overfit our model. In order to avoid this, we can say that a certain amount of points (say 10%) are outliers or do not represent our class. One first approach to do that is by taking the Kernel matrix and computing the average similarity of each row:

\[
S_i = \sum_{j=0}^{n} K(i, j) \quad \forall i \in 0, n
\]

Then, we order the values and exclude a certain percentage of them. This approach will work as expected if the samples of the model are in a single cluster. In data -sets with more than one cluster, we will be removing random points (the average of each row will be almost the same due to the large number of zeros).

The concept of reducing the training samples by eliminating those that are in the boundaries is almost the same as the shrinking of the convex hulls. In fact, we can join both concepts in an efficient fashion that will led us to better results in multi-clustered sets.
This approach consists on fixing a certain amount of points we want to consider as outliers, then we reduce the convex hull until we have the desired number of samples outside our model. By doing so, we are able to solve the case of multiple clusters and simplify the training phase.

### 3.5 Computational performance

In our case, in the training phase we only have to include the cost of computing the kernel matrix $O(n^2)$ and the cost of projecting a new point in the kernel space $O(\eta n d)$, giving a global complexity of $O(\tau (\eta n \log \eta n) + n^2)$. The cost of testing every point using a Ray-Casting algorithm is $O(\tau e)$ where $\tau$ is the number of projections and $e$ the number of edges of the polytope, setting the testing complexity at $O(\tau(n d + e))$. However, the most expensive algorithm, the kernel computation, can be computed for each of the projections instead for all the input set. In this case, the cost will be $O(\tau (\eta n)^2)$. Setting a low $\eta$ will compensate the cost of computing it for each of the projections, thus, this approach will be more efficient.

Notice that the cost of the N-APE algorithm is $O(C(\tau ( m \log m)n^2))$, where $C$ is the number of centers needed to model the shape and $m$ the number of points in the radius $r$ from each center and $n^2$ is the cost of obtaining the $m$ points within the radius. Compared to the cost of the K-APE, this second is more expensive considering that a large $C$ will be needed. It is exactly $C$ times slower than KAPE.

### 3.6 Classifying data with KAPE

One of the best well-known linear classifiers that take advantage by using kernels is SVM. Basically, it is a linear classifier that, by means of applying a kernel is able to linearly separate data that is not linearly separable.

As [25] demonstrates, computing an SVM classifier is the same as computing the convex hull for each of the sets and then, finding the hyperplane that maximizes the distance to them. The same procedure can be followed when working with non-separable cases, by applying a soft margin.

In our case, we have a method that is able to model data in a convex and non-parametric way. If we want to classify data by means of geometrical models, we have to, as [25] suggests, compute the separation between the sets. For the sake of simplicity, we are going to consider that we only have two classes in our problem. All the arguments are valid for higher numbers of classes.

- **The separable case** in which classes do not intersect, implies a first step of computing the KAPE for each of the classes, so we have a clear idea of its coverage in the space. Since we have that classes do not intersect and that there is a kernel that convexifies the data, we can guarantee that we would be able to separate both classes. In this case, since a confidence has been defined, we would be able to asses the proximity or the membership to any of the classes. At the end, for classifying as a class or another, the largest confidence is taken as the predicted class.

- **The non-separable case** in which classes are intersecting, is a much more difficult task. We can not guarantee that all the points will be well classified. In some cases, this intersection is due to uninformative features, so classes only intersect in some of the dimensions. In this case, we can take advantage by using a kernel and random projections in order to solve this problem.

In the case of sets that intersect in more dimensions, we will never be able to solve this situation, due to the fact that the same point is part of both classes so it is present in both convex sets. But, since we have a confidence on each of the models, we are able to assess a soft margin in which misclassifications
appear because of the shape of the sets. The soft margin is created when the two classes intersect. In the places in where the confidence is different, we are able to choose one class or another, but it might be a space in which both confidences are the same so we are not able to assess the final label (the decision will be taken randomly). Having a soft margin implies that for this area, there is an uncertainty because of the mixing of both classes. However, since we are modeling the classes independently from the other, it would not affect to the confidences. That is, even if we have an intersection, the confidence in this place will be the same as if there is no other class.

As it has been said before, the approach followed works as a SVM would do, so the behavior in separable/non-separable cases is the same. The only difference in our implementation is that we compute a confidence so we are able to deal easily with uncertainties, reducing the area in where we do not know the labels to a reduced one (i.e. the place in where both confidences are the same).

### 3.7 Working with sequences

Most of the state of the art on data modeling is about numerical datasets. However, there are several applications that are less explored and that have an additional degree of difficulty. We are talking about sequences, in particular, sequence classification.

In this particular case, we are going to deal with a gestural dataset (deeply explained in Section 4.2), acquired in the wild using Inertial Movement Units (IMUs). The gestures are from four classes: taking a pill, drinking from a glass, eating from a dish and reading a book. The subjects were 13 elder people. The recording has been made with minimal external intervention, so they were free to perform as they wanted. The result is a challenging dataset with large intra and inter-class variability and a small number of samples.

All the gestures have been previously segmented by using the ground-truth. Each of the sequences has a different length.

In order to adapt our method to a new input, we only have to modify the preprocessing step, that is the Kernel computation. Since an exponential kernel is basically the exponential of the euclidean distance between pair of points, we can apply any other distance metric. We selected Dynamic Time Warping algorithm (see Section 2) to compute alignment costs and use them as distances between sequences. We redefine the RBF kernel in order to use DTW instead of L2 norm and be able to deal with gestures. The kernel will be the following:

\[
K(x_i, x_j) = \exp\left(\frac{DTW(x_i, x_j) + DTW(x_j, x_i)}{-4\sigma^2}\right)
\]

We average both DTW distances in order to make the kernel matrix symmetric, since DTW has not the symmetry property.

Then, we can apply any algorithm that uses a kernel in the learning process, getting rid of the sequences and their different lengths.

Notice that the same procedure could be applied to images, text, categorical features or even structures, enabling the algorithm to deal with any kind of input data.

The procedure for modeling or classifying the data will be the same explained before.
4 Experiments and Results

4.1 Behavior of the method

In this section, the main parameters of the method are analyzed and explained, using results from different datasets. The aim is to give enough intuitions on how the method behaves when tuning them.

Notice that in the following figures, the accuracy value is not analyzed but the behavior of it. We have set some default parameters, able to work with any of the datasets. Accuracy values will be studied later.

Random subset length $\eta$.

The parameter that controls the percentage of points used to fit each Convex Hull is $\eta$. Next, we analyze the expected behavior of the algorithm when varying this value from (close to) 0 to 1. For lower values, the classifier will underfit on the data due to the low number of examples available for learning so the training accuracy will be low. However, negative test accuracy will be larger than training one since almost all the points will be classified as negative. Test accuracy will be proportional to the number of examples until a certain value in which it will saturate, after reaching its maximum. Train accuracy will increase as $\eta$ increases. $\eta$ also controls the probability of selecting outliers in the training phase, increasing the chances if it is large. This intuitions have been confirmed in empirical experiments as it is shown in Figure 5. One can observe, in (a), the performance of the method in the training phase when varying $\eta$; reaching the maximum value and then, saturating. In (b), the test set (positive class) is used to study the behavior, here one can observe the same performance as in the training phase. Finally, (c) represents the behavior on negative class, observe that the performance of the “moons” dataset decreases when $\eta$ does it. This dataset has a 30% of noise, so adding more training points, increases the probability of selecting outliers. Even that, the performance of the other two datasets is maintained along the different $\eta$ values.

![Figure 5: $\eta$ behavior on train and test stages](image)

0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
$\eta$

0.0 0.2 0.4 0.6 0.8 1.0
Accuracy

Train

0.0 0.2 0.4 0.6 0.8 1.0
Accuracy

Test (+)

0.0 0.2 0.4 0.6 0.8 1.0
Accuracy

Test (-)
Number of Convex Hulls $\tau$.

The number of Convex Hulls has a similar functioning than $\eta$. This is because when using more CH for describing a class, we are approximating it better. However, the rate of increase in accuracy is much larger because every new CH will increase the expressivity of the model by adding a new random projection and using $\eta$ percentage of points. In the case of the length of the subset, we are only adding more examples to the APE, making it to learn better. However, as it happens with $\eta$, large values of $\tau$ will prone to introduce outliers into the model and reducing its fit. Figure 6 illustrates the intuitions explained before. In (a), one can see that increasing the number of convex hulls, also increases the accuracy of the method, as it happens in (b), until a certain point at which, the model has its maximum accuracy. In (c), on can notice that accuracy values fluctuate over a narrow range. This means that $\tau$ is not very related to outliers accuracy and given a minimum number of projections, it is maintained.

![Figure 6: $\tau$ behavior on train and test stages](image)

Kernel parameter $\sigma$.

The $\sigma$ parameter of the Kernel controls the size of the gaussian bell. That is, for larger values, each data point will be prone to influence the rest, while for lower values, the influence of each data point is more local. Working in both ends (too large or too low) will lead us to bad situations. The first will imply a very restrictive influence area, where each data point will only see itself (Kernel matrix will be $I$) and will have perfect overfit. That means that the classifier is only able to predict as positive those values used in the training stage. A larger $\sigma$ implies an underfit on data. All the points will be similar to each other so the classifier sees the whole dataset as the same point. A good analogy to this behavior is a zoom in an image: if it is too big, we will only see a certain pixel, but if, on the contrary, we set it too small, we will not be able to distinguish the details on it. The behavior in the testing set is a bit different. In the case of small values, since the classifier is overfitted, accuracies will be low and on the other hand, for larger $\sigma$, when the method underfits the data, it will predict everything as positive (not able to distinguish the details) and test accuracy will also be low.

It is clear that an intermediate value of $\sigma$ will give the best results.
As Figure 7 shows, at a certain point, the performance of the model has a substantial change and increases rapidly, reaching its maximum. The underfit area is not represented in these plots since the required value is too large. In (c), one can see that the behavior of sigma on the negative class is the opposite. First, for lower values, the accuracy is high (every point is classified as negative). Then, at a certain value, it decreases until its minimum. For this $\sigma$ value, all the data points are classified as positive.

![Figure 7: $\sigma$ behavior on train and test stages](image)

**Membership threshold $\nu$.**

The membership threshold controls the boundary of the model and increases or reduces its coverage of the space. The membership of a point is computed as the percentage of convex hulls it lays inside. By increasing this value, being part of the class will be more restrictive, and only those points appearing in a large number of polytopes will be classified as members of the positive class. On the contrary, when reducing this value, those points belonging to a small set of convex hulls will be considered part of the class. Intuitively, this means that low values will include outliers or negative points and large ones, will exclude actual members of the class. The best value, as it is shown in Figure 8, will be that which minimizes the number of false negatives and the amount of false positives. Ideally, for positive data points, a lower $\nu$ value will give high accuracies. On the contrary, for negative ones, larger values of $\nu$ will maximize accuracies. Finding the optimal trade-off is crucial.

**Behavior conclusions**

After reviewing the behavior of the parameters, we can conclude that, the critical parameter of the method is $\sigma$. We have to find one that allows to “see” any point of the objective class but not able to cover the negative one. However, if we set a $\sigma$ higher than the optimal one, we can correct it by reducing the coverage of the CH (decreasing $\nu$). On the contrary, if we set a lower value of $\sigma$, the model will be underfitted and unseen points will lay in the singularity.

Both $\tau$ and $\eta$ are directly proportional to the complexity of the method so the optimal value should be chosen. This value is placed at the beginning of the saturation of the accuracy. Higher values will not be critical to the performance but for the complexity of the solution.
To sum up, $\sigma$ is a critical parameter. Incorrect settings of it could be improved by tuning the threshold $\nu$. Finally, $\tau$ and $\eta$ are related to the complexity and the expressiveness of the model.

4.1.1 Case study: Analysis of Ying-Yang dataset

In this section, a bi-dimensional dataset will be analyzed. The purpose of this chapter is to detail and compare the behavior of three of the methods: KAPE, OCSVM and GMM. By using this dataset we would be able to visualize the results and, furthermore, understand the intuitions behind them.

This dataset was artificially generated from a binary image. Random points were obtained using a uniform distribution over a part of the $\mathbb{R}^2$ space. Then, the label of each of the points was assigned by using the color of a binary image. In this case, the image represents the Chinese symbol of the Ying and Yang. Furthermore than the philosophical meaning, its shape is useful for the problem we want to show here. As is can be appreciated in Figure 9, the fact of having two islands in the middle of each of the classes, makes this problem difficult for some of the methods.
Next, we would analyze how OCSVM, GMM and KAPE behave on this dataset and, also, we would explain the main reasons of this behavior. The settings of the experiments will be the same for each of them, a 50% of the positive class will be used for train and the negative class and the remaining positive points, for test. Parameters are selected using a nested cross validation approach.

**Figure 10**: Once Class SVM performance in Ying-Yang dataset

**OCSVM.** This method, as it has been explained before, uses a kernel function in order to work in a multidimensional space, making any problem linearly separable. However, since only a fourth of the points (train set is the half of the positive class, which is the half of the set) is being used for learning the distribution, the geometrical shape is not well modeled by the method and thus, almost a third of the test set is not well classified. As it can be seen in Figure 10 the exterior boundary of the data is pretty well defined, which means that outlier points (the ones of the negative class) are correctly classified.

From this analysis, we can conclude that OCSVM is not a good choice when the training set is too small to understand the underlaying distribution of the data.

**GMM.** As it has been explained in Chapter 2, Gaussian Mixture Models use a set of Gaussian distributions in order to model a non-gaussian one. In this case, it is obvious that the underlaying distribution is not Gaussian. As before, a fourth of the points is used for training the method. In this case, GMM is achieving good results. However, it is using 18 gaussian models in order to represent the dataset. Observe, in Figure 11 that the Gaussians are small and covering a tiny area, so the probabilities are low in almost all the space. This means that the threshold should be fine-tunned in order to achieve a good representation. This causes the threshold to be a crucial value in the system, able to decide whether the method is modeling well the set or not, even if the number of components is correctly selected.

**KAPE.** The Kernel-APE uses some properties of each of the previous models in order to represent a dataset. KAPE is an ensemble method, using a set of convex hulls in order to represent an unknown distribution. However, unlike GMM, each of the simple model do not depend on any parameter, they are simply the convex polygon
that limits a set of points, easing the training and the parameter selection. Moreover, it uses a kernel function in order to map the points to an inner product space, as the One Class SVM does. As one can see, KAPE uses the idea of representing the points by using their inner products in a multi-dimensional space and constructs a model by ensembling a set of simpler representations of a complex space.

As it is shown in Figure 12, KAPE is modeling the shape of the class and giving a correct distribution of probabilities. In Figure 13, one can see the difference of distribution of the probabilities of each set of points. Notice that while all GMM membership values are centered in the lower part of the interval, KAPE is distributing them, situating the ones corresponding to the positive points in the upper part and the ones of the negative points in the lower part, being able to clearly distinguish between them.
**Conclusions.** From this experiment the reader may observe that, while OCSVM is not behaving well with a small number of training samples and GMM is able to model complex shapes at the cost of using a large amount of components, KAPE is using key concepts of each of them and using it to geometrically represent the underlaying distribution of the data, using a non-parametric approach and behaving well when using a small set of training points.

### 4.2 Datasets

**State-of-the-art datasets.** In order to compare the behavior of the method against other one-class approaches, we are going to use several datasets commonly used in literature. In Table 1, the details of each of the dataset are shown. Observe
that these data sets are meant for classification. In the one-class setting we will model one of the classes disregarding all the information from the other class. This information will only be used later in the experiments for assessing the performance of the methods.

The number of features use to be an important specification of the input data. Here we have a range of values from 2 to 60. In our case it is not so important because of the use of Kernels.

Finally, we decided to choose an small number of samples in order to be able to increase the combinations of parameters for each of the methods. However, we cover the range that goes from about 200 to more than 7000.

Additionally, those datasets with more than two classes will be split using a one-versus-all strategy, increasing the number of datasets.

<table>
<thead>
<tr>
<th>Name</th>
<th>Features</th>
<th>Classes</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>13</td>
<td>2</td>
<td>270</td>
</tr>
<tr>
<td>mg</td>
<td>6</td>
<td>2</td>
<td>1385</td>
</tr>
<tr>
<td>Ying-Yang</td>
<td>2</td>
<td>2</td>
<td>787</td>
</tr>
<tr>
<td>fourclass</td>
<td>2</td>
<td>2</td>
<td>862</td>
</tr>
<tr>
<td>australian</td>
<td>14</td>
<td>2</td>
<td>690</td>
</tr>
<tr>
<td>segment</td>
<td>19</td>
<td>7</td>
<td>2310</td>
</tr>
<tr>
<td>diabetes_scale</td>
<td>8</td>
<td>2</td>
<td>768</td>
</tr>
</tbody>
</table>

Table 1: Details of the used datasets

4.3 Settings

One class problems have some particularities in terms of how training is performed. The main problem when training a classifier using a single class is the parameter selection. It is easy to overcome with a trivial solution that would not be able to generalize well. This will result in poor test accuracy rates. This is due to the fact of not having negative samples in the training phase.

Almost all the methods have two parameters we have to tune:

- KNN: Number of neighbors and Distance threshold
- K-APE: Sigma and Threshold
- GMM: Number of components and Threshold
- OCSVM: $\nu$ and $\gamma$
- Gauss: Threshold

We train and test using all the combinations. Then we define a percentage of training points we want to reject (i.e. we consider them outliers). With the obtained results, we find the best combination that gives the closest rate of outlier rejection. For doing so we used a nested cross validation strategy to select the best parameter and finally assess an accuracy value.

Threshold parameter is tuned again in the testing phase selecting the one that corresponds to the crossing of test and outlier curves. This is known as the Equal Error Rate.

**Fixed parameters.** Some of the parameters of the K-APE method have been fixed. After analyzing the behavior of the method, we decided to fix the percentage of subsampling ($\eta$) to two thirds and the number of projections ($\tau$) to 300, since we consider their behavior is not critical when setting non-limit values.
**Parameters range.** The parameters that require a validation are: threshold $\nu$ and RBF-Kernel $\sigma$. The resolution of the threshold is directly related to the number of projections. That is, since the threshold $\nu$ shows the proportion of polytopes in which a point lays inside, the step size of this parameter is $1/\eta$ and bounded in the interval $(0, 1]$. Since every dataset is standarized, $\sigma$ is bounded in the interval $(0, 3]$ and we will use 20 different values.

### 4.4 Results

In this section, the results obtained by the proposed method are compared to the ones given by the state of the art presented in Chapter 2. Two comparisons will be made, first of all, both kernel methods (OCSVM and KAPE) will be compared. Next, KAPE will compete against other non-kernel methods. In order to compare kernel methods, one may look at Tables 2 and 3 first and last columns. These results show that KAPE outperforms OCSVM in almost all the tested datasets.

Comparing KAPE with the rest of the methods, it is found to be be in third position. Observe that the difference between the three first methods, both in EAR and in AUC, is not statistically significant. If we look at the EAR, KNN is the best method, followed by GMM and KAPE. On the other hand when looking at the AUC, GMM is the best of the five followed by KNN and KAPE.

If we take a closer look at the results, we find different behaviors of KAPE. If we check the plots of the EAR Figures [14][16] one can observe that KAPE converges in all the datasets except for `segment_0`. Notice also that KNN and GAUSS fail to converge in several datasets.

In Figure 16, one may observe that KAPE is not able to achieve good results on `diabetes_scale` dataset. Moreover, in Figure 19 it is evident that the classifier is not working properly in this dataset. This is due to the number of convex hulls used. Augmenting this value, will improve the expressivity of the model, yielding a better result.

One may observe the difference between Figure [15] and [18]. The model is achieving good results in this dataset but, the ROC curve shows that more parameters should be tested in order to achieve a correct result. Observe also in Table 2 that this dataset is achieving large performances in other methods and actually, one of the lowest values is the one achieved by KAPE.

It is important to mention the case of `mg` (Figure 17) in which the difference between KAPE and OCSVM is notable.

As one may observe in Table 4 KNN is the best of the methods, and KAPE performs five out of seven times better than OCSVM. However, KAPE is not performing better than GMM and draws against Gauss.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>KAPE</th>
<th>KNN</th>
<th>GMM</th>
<th>GAUSS</th>
<th>OCSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>0.68</td>
<td>0.66</td>
<td>0.68</td>
<td>n/a</td>
<td>0.63</td>
</tr>
<tr>
<td>mg</td>
<td>0.73</td>
<td>0.75</td>
<td>0.75</td>
<td>0.79</td>
<td>0.61</td>
</tr>
<tr>
<td>Ying-Yang</td>
<td>0.91</td>
<td>0.91</td>
<td>0.82</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td>fourclass</td>
<td>0.98</td>
<td>1.00</td>
<td>0.90</td>
<td>0.66</td>
<td>0.76</td>
</tr>
<tr>
<td>australian</td>
<td>0.69</td>
<td>0.68</td>
<td>0.74</td>
<td>0.76</td>
<td>0.63</td>
</tr>
<tr>
<td>segment_0</td>
<td>0.93</td>
<td>0.97</td>
<td>0.98</td>
<td>0.93</td>
<td>0.95</td>
</tr>
<tr>
<td>diabetes_scale</td>
<td>0.48</td>
<td>0.58</td>
<td>0.57</td>
<td>0.61</td>
<td>0.57</td>
</tr>
<tr>
<td>Average</td>
<td>0.77 ±0.17</td>
<td>0.79 ±0.15</td>
<td>0.78 ±0.13</td>
<td>0.65 ±0.28</td>
<td>0.70 ±0.12</td>
</tr>
</tbody>
</table>

Table 2: Results showing Equal Accuracy Rate

29
Figure 14: Equal Accuracy Rates (in red) for positive (blue) and negative (green) data points of the test set
Figure 15: Equal Accuracy Rates (in red) for positive (blue) and negative (green) data points of the test set
Figure 16: Equal Accuracy Rates (in red) for positive (blue) and negative (green) data points of the test set
Figure 17: Receiver Operating Characteristic (ROC) curve for each of the datasets and each of the models and the computed Area Under the Curve (AUC)
Figure 18: Receiver Operating Characteristic (ROC) curve for each of the datasets and each of the models and the computed Area Under the Curve (AUC)
Figure 19: Receiver Operating Characteristic (ROC) curve for each of the datasets and each of the models and the computed Area Under the Curve (AUC)
<table>
<thead>
<tr>
<th>Dataset</th>
<th>KAPE</th>
<th>KNN</th>
<th>GMM</th>
<th>GAUSS</th>
<th>OCSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>heart</td>
<td>0.740</td>
<td>0.727</td>
<td>0.735</td>
<td><strong>0.761</strong></td>
<td>0.686</td>
</tr>
<tr>
<td>mg</td>
<td>0.811</td>
<td>0.832</td>
<td>0.842</td>
<td><strong>0.862</strong></td>
<td>0.678</td>
</tr>
<tr>
<td>Ying-Yang</td>
<td><strong>0.971</strong></td>
<td>0.950</td>
<td>0.894</td>
<td>0.827</td>
<td>0.812</td>
</tr>
<tr>
<td>fourclass</td>
<td>0.983</td>
<td><strong>0.999</strong></td>
<td>0.966</td>
<td>0.737</td>
<td>0.832</td>
</tr>
<tr>
<td>australian</td>
<td>0.749</td>
<td>0.724</td>
<td><strong>0.810</strong></td>
<td>0.747</td>
<td>0.678</td>
</tr>
<tr>
<td>segment_0</td>
<td>0.933</td>
<td>0.969</td>
<td><strong>0.984</strong></td>
<td>0.979</td>
<td>0.978</td>
</tr>
<tr>
<td>diabetes_scale</td>
<td>0.483</td>
<td>0.628</td>
<td>0.607</td>
<td><strong>0.644</strong></td>
<td>0.590</td>
</tr>
<tr>
<td>Average</td>
<td>0.810</td>
<td>0.833</td>
<td><strong>0.834</strong></td>
<td>0.798</td>
<td>0.751</td>
</tr>
</tbody>
</table>

Table 3: Results showing Area Under the Curve

<table>
<thead>
<tr>
<th></th>
<th>KAPE</th>
<th>KNN</th>
<th>GMM</th>
<th>GAUSS</th>
<th>OCSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAPE</td>
<td>-</td>
<td>2/1/4</td>
<td>2/1/4</td>
<td>3/1/3</td>
<td>5/0/2</td>
</tr>
<tr>
<td>KNN</td>
<td>4/1/2</td>
<td>-</td>
<td>3/1/3</td>
<td>4/0/3</td>
<td>7/0/0</td>
</tr>
<tr>
<td>GMM</td>
<td>4/1/2</td>
<td>3/1/3</td>
<td>-</td>
<td>4/0/3</td>
<td>6/1/0</td>
</tr>
<tr>
<td>GAUSS</td>
<td>3/1/3</td>
<td>3/0/4</td>
<td>3/0/4</td>
<td>-</td>
<td>4/0/3</td>
</tr>
<tr>
<td>OCSVM</td>
<td>2/0/5</td>
<td>0/0/7</td>
<td>0/1/6</td>
<td>3/0/4</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: Results showing Wins/Draws/Loses in EAR values
5 Gesture recognition

In the recent years, a lot of wearable devices have appeared. Most of them are worn in the wrists, which means that they could (in the case of having the proper sensors) recognize gestures. Gesture detection and recognition are challenging tasks with low accuracy rates in the wild. Most of the datasets found in literature consist on sequences of single gestures. Moreover, this gestures use to be artificial. This implies that the confusion between classes will be reduced since they were crafted to be so.

We decided to record a dataset in the wild with real gestures and in a multi-modal setting, combining accelerometer, gyroscope, RGB and Depth images.

5.1 Methodology

The recorded dataset consists on 31 sequences of 13 different elder people performing daily activities with a sensor in the right arm (their dominant arm). Subjects participating in the recording have some health impairments that will make the recognition even more challenging. The gestures labeled in the sequences are: take a pill, drink from a glass, eat with a spoon and read a book. The recordings were made with minimal external intervention, so subjects were free to interact with the objects.

The users were situated in a chair in front of a table with a set of objects to interact with: a pillbox, a dish, a book, a glass and a Tetrabrik of juice. The sequences were also recorded using two Kinect™ RGB-Depth cameras faced one to the other, so occlusions are reduced. In Figure 20 one can see some sample frames of the recorded sequences. The first row shows the RGB data obtained by the cameras and in the second row, the depth values recorded. One can see that the recording environment have been set in order to recreate real conditions. Table 5 shows the details about the recorded dataset. It is important to notice the difference of frames between wearable and vision due to the different recording rates.

![Figure 20: Recorded images in RGB (first row) and Depth (second row)](image)

5.2 Data

As it has been said before, we obtained accelerometer and gyroscope from the wearable module and RGB and Depth from the cameras. Both modalities can be synchronized a posteriori using the recorded timestamps.

We used a Shimmer® sensor (shown in Figure 21) attached to the wrist, connected via Bluetooth® to an Android phone that will label the sequences with a timestamp and send them to the server that will store them. On the other side, on the vision module, Kinect™ cameras use a ping-pong protocol to record even frames due to interferences (when both cameras are turned on, the IR emitters intersection makes...
Table 5: Summary of the SARquavitae Claret dataset

<table>
<thead>
<tr>
<th>Modules</th>
<th>Vision</th>
<th>Wearable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
<td>2x Kinect</td>
<td>1x IMU (Shimmer)</td>
</tr>
<tr>
<td><strong>Type of data</strong></td>
<td>RGB</td>
<td>Accelerometer</td>
</tr>
<tr>
<td></td>
<td>Depth</td>
<td>Gyroscope</td>
</tr>
<tr>
<td><strong>No. sequences</strong></td>
<td>31</td>
<td></td>
</tr>
<tr>
<td><strong>No. subjects</strong></td>
<td>14</td>
<td></td>
</tr>
<tr>
<td><strong>No. frames</strong></td>
<td>3,747 + 3,701</td>
<td>36,858</td>
</tr>
<tr>
<td><strong>No. actions (gestures)</strong></td>
<td>86 (162)</td>
<td></td>
</tr>
<tr>
<td><strong>General challenges</strong></td>
<td>Elderly subjects, uncontrolled behavior</td>
<td></td>
</tr>
<tr>
<td><strong>Specific challenges</strong></td>
<td>Ambient light, Reflexes and shadows, Small objects, Low framerate, Depth noise</td>
<td>Gesture intra-inter variability, Device noise</td>
</tr>
</tbody>
</table>

the sensors to record corrupted data). Due to this problem, the sampling rate is about 2 frames per second. The frames are also labeled with a timestamp.

![Shimmer® sensor](image)

Figure 21: Shimmer® sensor used in the dataset

The wearable module, the one that we are going to use in this work, includes accelerometer (m/s²) and gyroscope (rad/s) sensors. Samples are obtained at approximately 25Hz which means 25 measurements per second.

We are going to use only pre-segmented wearable data. This means that we are not working with full sequences but with gesture sequences instead. This is because we do not want to implement gesture detection but only a gesture classifier and be able to prove that K-APE is able to work with multiple types of data, in this case sequences of different length. In Figure 22 one can see an example of a full sequence recorded using the sensor. The segmented gestures used in this work are shown in Figure 23 in which the colored area limits the gesture in each of the sequences. Observe the similarity between turn-page and drink gestures.

Some of the activities/gestures we are intended to recognize are quite similar one to another: the arm movement is very similar in “drinking” and “taking the pill” from the perspective of the vision module. In addition, in the inertial cue, there is also a certain degree of similarity between gestures of different classes compared to the “no-
After this analysis, we have concluded that the before-mentioned difficulties are large enough to encourage us to find other derived features. In order to empower the learning of the classifiers, we selected a set of feature candidates:
Raw accelerometer data. Data recorded on the scenario regarding only to accelerometer.

Sorted accelerometer. A set of discrete features which account for a relative rank among the three axis of the accelerometer is defined. For each sample we assign a value (-1, 0 or 1) depending according to the ranking of its value compared to the other axis, i.e. the axis with the lowest value is set to -1, the axis with the largest value is set to 1, and the remaining one to 0.

Complementary filter. The complementary filter mixes gyroscope and accelerometer values in order to get a smoothest signal with less noise and transforming acceleration into rotations. In essence, we transform the acceleration vector \( \mathbf{a} \) of each sample into the rotation vector, then we apply a low-pass filter to the accelerometer in order to remove noise, and a high-pass filter to the gyroscope for removing the drift (an almost constant component). Then we merge both measures in order to get the orientation of the sensor. The rotation vector from the accelerometer is computed as follows:

\[
\alpha = \cos^{-1} \frac{x}{|\mathbf{a}|}, \quad \beta = \cos^{-1} \frac{y}{|\mathbf{a}|}, \quad \gamma = \cos^{-1} \frac{z}{|\mathbf{a}|}
\]

Where \( x, y, z \) are the components of the acceleration vector. Then, for each sample \( i \) we are able to apply the complementary filter defined on the next equation:

\[
C_{x_i} := \sigma \cdot \theta_{x_i} + (1 - \sigma) \cdot \alpha_i
\]

\[
C_{y_i} := \sigma \cdot \theta_{y_i} + (1 - \sigma) \cdot \beta_i
\]

\[
C_{z_i} := \sigma \cdot \theta_{z_i} + (1 - \sigma) \cdot \gamma_i
\]

The value of \( \theta \) represents the gyroscope values, while the value of \( \sigma \) controls the response of the filter; the highest the value, the biggest the response.

Jerk. We use Jerk, which is the derivative of the acceleration \( \mathbf{j}(t) = \frac{d\mathbf{a}(t)}{dt} \). It shows the transitions of the acceleration and is numerically computed using centered differences.

5.3 Feature selection

Given the set of features described in the former section, all their possible combinations have been generated. Using this information we generated a set of distance matrices (using DTW as metric) in order to test which is the combination that minimizes intra-class confusion and increases inter-class similarity.

We include the distance matrices given for several feature combinations (Figures 24(a)-33(b)). Observe that most of them get the lower distances in the diagonal for the intra-subject case. However, the inter-subject matrices suffer from confusion and this is determinant in order to choose the best set of features. Observe the case of, for example, Figures 25(a) and 25(b). In this case, Intra-subject distances are acceptable (we are not able to differentiate Take-pill only by distance) but the inter-subject distances are behaving in a different way. It is evident that the fact of having close distances in each row will hinder the ability to learn of the classifiers applied.

In Figure 33(a) and 33(b) one can see that the best results are the ones given by the combination of raw accelerometer, sorted accelerometer, jerk and complementary filter.
Even that this work is only centered in gesture recognition (or classification) tasks, gesture spotting or activity recognition can also be applied to this dataset. Indeed, other works are being developed, centered in activity recognition, a task that involves gesture spotting and recognition, tracking and human-object interaction.
Figure 27: Sorted accelerometer + Complementary filter

Figure 28: Sorted accelerometer + Jerk

Figure 29: Complementary filter + Jerk
Figure 30: Raw accelerometer + Sorted accelerometer + Complementary filter

Figure 31: Raw accelerometer + Sorted accelerometer + Jerk

Figure 32: Sorted accelerometer + Complementary filter + Jerk
Figure 33: Raw accelerometer + Sorted accelerometer + Complementary filter + Jerk
5.4 Settings

In the case of the SARQuavitae Claret dataset, we would use a multi-class approach. In this case, we are only going to test K-APE and SVM due to the need of using a custom kernel (input sequences have different lengths). We are going to use a One-vs-ALL strategy. In the case of K-APE, one K-APE will be modeled per class. Then, for each test sample, the membership to each of the models will be assessed and the label will be assigned taking into account its maximum value. In the case of SVM, the classic One-vs-ALL approach will be used.

5.5 Results

In this section we would analyze the results obtained on SARQuavitae Claret using SVM and KAPE methods, in a multi-class approach.

In Table 6, one can observe the values obtained for each of the classes. It is important to mention that, even that KAPE achieves lower average accuracies rates, it improves the results obtained by SVM in two of the classes. Even that, it is patent that KAPE is not able to differentiate some of the classes. In particular, Drink class works specially bad. In Figures 34, 35 and 36 the confusion matrix for each of the methods is shown. Notice that, in those cases in which KAPE is below SVM accuracies, there is a confusion against one of the other classes. It is important to mention that this confusion is only with one class, which suggests that the problem might be related with the volume of data available. Notice also that SVM is behaving as it was described for KAPE in those cases in which it has low accuracy rates.

Notice the difference between SVM (multi-class approach) and OC-SVM (one-class approach). It its important to highlight that even that KAPE is designed for one-class problems, is able to generalize to multi-class problems, achieving competitive performance rates. Moreover, the difference between KAPE and OCSVM proves the versatility of KAPE in contrast to the behavior of OC-SVM.

<table>
<thead>
<tr>
<th></th>
<th>SVM</th>
<th>KAPE</th>
<th>OCSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take-pill</td>
<td>6/12  (0.33%)</td>
<td>11/7  (0.61%)</td>
<td>0/18  (0%)</td>
</tr>
<tr>
<td>Drink</td>
<td>26/13 (0.67%)</td>
<td>12/27 (0.33%)</td>
<td>17/22 (0.43%)</td>
</tr>
<tr>
<td>Turn-page</td>
<td>23/10 (0.69%)</td>
<td>26/7  (0.79%)</td>
<td>1/32  (0.03%)</td>
</tr>
<tr>
<td>Spoonful</td>
<td>57/10 (0.85%)</td>
<td>44/23 (0.52%)</td>
<td>1/66  (0.01%)</td>
</tr>
<tr>
<td>Total</td>
<td>112/45 (0.71%)</td>
<td>93/64 (0.59%)</td>
<td>19/138 (0.12%)</td>
</tr>
</tbody>
</table>

Table 6: Gesture dataset results, correct classifications in green, misclassifications in red and accuracy is indicated in brackets
**Figure 34**: Gesture dataset confusion matrix for SVM
Figure 35: Gesture dataset confusion matrix for KAPE
Figure 36: Gesture dataset confusion matrix for OCSVM
6 Conclusions and future work

6.1 Conclusions

In this work, we first present a study of the geometrical method Approximate Polytope Ensemble (APE) and the derived Non-Convex-APE (NAPE). In these methods, a one-class problem is modeled by means of random projections in 2-dimensions. We introduce the concept of core (the region at which all the convex hulls intersect) and thus, the concept of membership to the class, which accounts for the proportion of convex hulls in which a test data point, lays in. Following the approach applied in Support Vector Machines, in which a kernel function is used in order to enable a linear classifier to model non-linearly separable data and, in order to improve the original implementations of APE and NAPE, a kernel function has been introduced to the original APE formulation. Moreover, the addition of a kernel function in the formulation of our method, enables us to use any input data, increasing the utility of the model. This leads us to the implementation of Kernel-Approximate Polytope Ensembles, which is one of the main proposals of this work. In order to compare the performance of KAPE and the one of the classifiers used in literature, we have chosen a bunch of commonly used datasets. In an effort to show the utility of the kernel function by enabling us to use any type of input data, we designed and generated a dataset of actions and interactions in the wild, with elder people. This dataset has been recorded giving maximum freedom to the subjects that were collaborating, yielding a challenging dataset. With the application of the proposed method to the recorded data, we prove the generalization of KAPE to multi-class problems and to any kind of input data.

With the results obtained in the one-class approach, the reader may observe that the outcomes are not statistically different from the ones given by the state of the art methods in most of the datasets. This leads us to the conclusion that the method is working as expected. The case of the Ying-Yang dataset explained in this work, is very illustrative and enables us to understand the behavior of KAPE compared to GMM and OCSVM. Intuitively, it is easy to notice that our method is approximating a complex shape by means of the intersection of polygons, yielding a polytope that bound the target class. Given some complex shapes (as the one proposed), GMM is not able to approximate correctly the shape of the target class. Moreover, OCSVM tries to model the boundary but it is not able to finally achieve the target shape.

The main weak point of the proposed method is the fact of needing to store the Gramm matrix in order to test any test data point, that decreases the efficiency of the method in terms of memory storage. However, if the proposed thesis, that states that only the vertices will need to be stored in order to project new points to the kernel space, holds, this problem will be solved, improving the memory allocation of the model.

Even that the proposed method has some weak points, advantages overcome them. First of all, as any geometrical model, its functioning is intuitive and easy to understand. The fact of using low-dimensional projections (2D), makes it suitable for plotting and enables us to trace any point and visually understand what is happening. Moreover, it is computationally efficient and improves the performance of NAPE and is far more efficient than SVM ($O(n^3)$) and is close to K-NN ($O(n^2)$). However, it is far from GMM which complexity is $O(nk^3)$ and Gaussian Model, $O(n)$. Another advantage is that KAPE is non-parametric because the convex hull (the base method) only needs a set of points, which makes it easier to validate. It has been demonstrated in Chapter 4 the robustness of the method when varying the parameters. Thus, most of them can be fixed and the validation process is reduced even more. Finally, as it has been explained before, any input data could be given to the method.

We can conclude that Kernel-APE is a method with a large potential and capac-
ity of improvement, given the points detailed before and the proposals described in Chapter 6.2. The study of One-class problems has been useful for accounting for different types of problems, with certain particularities and very interesting challenges, not common in literature. Even that, One-class approaches are often needed in actual problems. Moreover, the creation of a gesture dataset in the wild, enabled us to understand the particularities of this type of data. The fact of participating in all the process, from the very beginning, in the initial study, until the end, in the recordings and the labeling, gives us a wide view of the problem we are working with, helping to understand the key reasons of any problem we can found in the use of this dataset.

Finally, I must say that the experience of developing this work explained here, makes me feel like I have become enriched as a junior researcher and helped me to discover other fields and methods that were unknown for me. I have also improved my expertise in writing scientific documents with the help of the related articles I have written and this particular work.

6.2 Future work

The work presented in this Master Thesis is a first approach to the kernelized version of the Approximate Polytope Ensemble method. Even that lots of aspects have been studied and analyzed, there are some others that, due to a lack of time and an interest in focusing in other interesting points, have not been tackled. For instance, we could, in order to improve test performance, project new data points in the subspace formed by the vertices of the convex hull. This approach will help us to improve the efficiency of the method and, probably, achieve the same results. The intuition behind this approach is that, any point inside a convex hull, is a convex combination of the vertices.

The multi-class approach proposed shows the multi-class behavior of Kernel-APE. In contraposition to other one-class targeted methods, KAPE is able to model a multi-class problem, achieving reasonably good performance rates and even improving SVM. For this reason, this approach should be studied and tested against other kind of datasets.

Moreover, different kernels could be used in order to take profit of the intrinsic properties of the datasets being tested.
References


