Evaluation of Kernel Methods for Change Detection and Segmentation

Application to Audio Onset Detection

Alexandre Lung-Yut-Fong
Abstract

Evaluation of Kernel Methods for Change Detection and Segmentation

Alexandre Lung-Yut-Fong

Finding changes in a signal is a pervasive topic in signal processing. Through the example of audio onset detection to which we apply an online framework, we evaluate the ability of a class of machine learning techniques to solve this task.

The goal of this thesis is to review and evaluate some kernel methods for the two-sample problem (one-class Support Vector Machine, Maximum Mean Discrepancy and Kernel Fisher Discriminant Analysis) on the change detection task, by benchmarking our proposed framework on a set of annotated audio files to which we can compare our results to the ground-truth onset times.
Acknowledgements

First, I would like to thank the TSI department at Telecom ParisTech for welcoming me during this period of six month.

Special thanks to my advisor at Telecom ParisTech, Olivier Cappé, who welcomed me and provided me with wise pieces of advice that allowed me to complete this thesis.

Elisabeth Larsson, for her advice, suggestions and numerous comments concerning this report.

Thanks also to Slim Essid and Cyril Joder, from the AAO team for their help in audio related subjects.

And finally to the members of the STA team, Céline Levy-Leduc, Éric Moulines, François Roueff, Steffen Barebruch, Julien Cornebise, Marine Depecker, Sarah Filippi, Jean-François Germain, Zaid Harchaoui, Olaf Kouamo, Erwan Moreau, Tabea Rebafka, Nataliya Sokolovska, for the moments more or less related to this thesis that we have spent together.
## Contents

1 Introduction 7  
1.1 Background .................................. 7  
1.2 Goals and procedure ......................... 7  
1.3 Outline of this thesis ....................... 8  

2 On kernel methods 9  
2.1 Introduction ................................ 9  
2.2 Kernel theory ................................ 10  
2.2.1 Examples of kernels .................... 11  
2.3 Support Vector Machines .................... 12  
2.4 Computing a dissimilarity measure ......... 14  
2.4.1 Online Kernel Change Detection ........ 14  
2.4.2 Maximum Mean Discrepancy (MMD) ....... 17  
2.4.3 Kernel Fisher Discriminant Analysis (KFDA) 18  

3 Experiments on artificially generated data 23  
3.1 Introduction ................................ 23  
3.2 Detection theory ............................ 23  
3.2.1 Hypothesis testing ..................... 23  
3.2.2 Detection performance .................. 24  
3.3 Simulations ................................. 24  
3.3.1 Test protocol ............................ 25  
3.3.2 Data sets ............................... 26  
3.3.3 Results and interpretation ............. 27  

4 Application to onset detection in audio files 35  
4.1 Introduction and background ................. 35  
4.2 Related work ................................ 35  
4.3 Onset detection ............................. 36  
4.3.1 General scheme ......................... 36
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Embedding into a feature space</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>1-class SVM geometric criterion</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>ROC curve</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Visualisation of the 1-class SVM decision boundaries in the &quot;cross&quot; distributions</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>2D distribution, 50 points, $q_1$</td>
<td>28</td>
</tr>
<tr>
<td>3.4</td>
<td>9D distribution, 150 points, $q_1$</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>2D distribution, 150 points, $q_2, \theta = \pi/8$</td>
<td>30</td>
</tr>
<tr>
<td>3.6</td>
<td>2D distribution, 50 points, $q_2, \theta = \pi/8$</td>
<td>31</td>
</tr>
<tr>
<td>3.7</td>
<td>2D distribution, 50 points, &quot;blobs&quot;</td>
<td>32</td>
</tr>
<tr>
<td>3.8</td>
<td>9D distribution, 150 points, &quot;blobs&quot;</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Flowchart of onset detection algorithm</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>First 16 cosine basis functions</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>First cepstral coefficient of a 6-sec. extract of a piano tune and its associated delta (blue bold line). $m = 3$</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>General abrupt change detection framework based on the time series of descriptors (represented by circles)</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Waveform and spectrogram of a mandolin tune</td>
<td>46</td>
</tr>
<tr>
<td>5.2</td>
<td>Detection functions. $w = 10, \sigma = 0.5$</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>KFDA detection functions with varying $\gamma$</td>
<td>48</td>
</tr>
<tr>
<td>5.4</td>
<td>Precision - recall curves for all algorithms</td>
<td>50</td>
</tr>
<tr>
<td>5.5</td>
<td>F-measure by music category</td>
<td>51</td>
</tr>
<tr>
<td>5.6</td>
<td>Influence of the parameters on MMD</td>
<td>52</td>
</tr>
<tr>
<td>5.7</td>
<td>Influence of $\gamma$</td>
<td>53</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

Finding changes in signals is a topic that is pervasive in signal processing. The number of applications is large [BN93]. It includes for example quality control in industrial processes, seismic data processing, parameter estimation for adaptive algorithms or signal segmentation. This task consists in detecting the time instants of abrupt changes, i.e. sudden changes in the local characteristics of the considered signal. It has mostly been addressed from a parametric point of view, which involves in particular a careful modelling of the behaviour of the signal between changes.

Recently however, more generic approaches based on the use of kernels, such as the so-called one-class Support Vector Machine [STC04], have shown good performance in several applications. Kernel methods are a class of pattern analysis algorithms, which are a sub-topic of machine learning, through the ability to learn and adapt to new situations, solely from the observation of the presented data.

The purpose of this Master’s project is to evaluate several kernel-based novel approaches for change detection. There are various modes in which the segmentation task can be undertaken. We will study in particular the sequential, or online version, where the changes are to be detected while observing the data. The effort will be focused on audio segmentation tasks, or more precisely on onset (i.e. the beginning of a note in a musical tune) detection algorithms, a problem that can be viewed as one step in a music transcription procedure, for example.

1.2 Goals and procedure

The onset detection task is traditionally tackled with two general approaches, based either on deterministic models, i.e. on the characteristics of the signal or on probabilistic
models, that is trying to find a modelling of the signal and guess the potential onset times.

However, in this work we present an unsupervised machine learning technique to build an onset detector, based on dissimilarity measures, i.e. finding a distance between the support of two sets of descriptors in Reproducing Kernel Hilbert Spaces, allowing the reuse of some known linear algorithms in a non-linear fashion.

Unlike with the deterministic methods mentioned earlier, machine learning techniques do not need any modelling of the signal under study; they however require a careful tuning of some meta-parameters that are critical to the performance of the machine. In this work, we will try to see the effects of these parameters on some artificial examples and then on real world data by benchmarking our methods on an annotated audio corpus.

1.3 Outline of this thesis

First, in chapter 2, we will introduce the concept of kernels, which are a very important part in the different algorithms we are using to generate the detection functions in our onset detection task. Then in chapter 3, we will exhibit the performance of these on some artificially generated data, which will also enable us to visualise the effects of the kernels. In chapter 4, we will detail the onset detection framework, how we intend to find the onset times, as well as the evaluation procedure that will be used to compare our different solutions. And finally we will present and discuss the results of our experiments in chapter 5.
Chapter 2

On kernel methods

2.1 Introduction

The main idea behind machine learning algorithms is to extract information out of raw data. The designer of a learning system, although not exempted from doing careful assumptions and making choices about the problem she is tackling, is working in a model-free environment.

For example, in a classification task, one has to automatically assign a label to some new data. The machine learning is literally inferring the decision function from the data observations it has previously seen, without any preliminary information about the distribution they are drawn from.

Kernel methods are a class of algorithms that enables the re-use of widely studied linear algorithms to solve non-linear problems. To do so, a map $\varphi$ is created to embed the non-linearly related data into a high dimensional space $\mathcal{F}$ where linear relations can be more easily found between the images of the data vectors.

Figure 2.1: Embedding into a feature space
2.2 Kernel theory

Let us first define the mapping from the $d$-dimensional input space $\mathcal{X} \subseteq \mathbb{R}^d$ into a feature space $\mathcal{F}$:

$$\varphi : \mathbb{R}^d \rightarrow \mathcal{F}$$

$$\mathbf{x} \mapsto \varphi(\mathbf{x}) \quad (2.1)$$

So, to each vector $\mathbf{x}$ from the input space, we associate an element $\varphi(\mathbf{x})$ from the feature space.

In $\mathcal{F}$, the dot product is noted $\langle \cdot, \cdot \rangle_\mathcal{F}$.

**Definition 2.1 (Kernel)** A kernel $k$ is a real-valued function of two variables:

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$$

Let $k$ be the kernel over $\mathcal{X} \times \mathcal{X}$ such that:

$$\forall (x, z) \in \mathcal{X} \times \mathcal{X}, k(x, z) = \langle \varphi(x), \varphi(z) \rangle_\mathcal{F}$$

**Definition 2.2** A kernel $k$ is positive if the following propositions hold:

$$\forall (x, z) \in \mathcal{X}^2, k(x, z) = k(z, x) \quad (\text{symmetry})$$

$$\forall n \in \mathbb{N}, \forall (\alpha_i)_{i=1}^n \in \mathbb{R}, \forall (x_i)_{i=1}^n \in \mathcal{X}, \sum_{i,j=1}^n \alpha_i \alpha_j k(x_i, x_j) \geq 0$$

We have just introduced a feature map and a feature space, which were defined *a priori*. But actually, given a positive semi-definite kernel $k$, we can build a feature map $\varphi$ in a Hilbert feature space $\mathcal{F}$.

**Definition 2.3 (Reproducing Kernel Hilbert Space)** Let $k$ be a positive semi-definite kernel in $\mathcal{X}$. Then there exists an Hilbert space $\mathcal{F}$ endowed with an inner product $\langle \cdot, \cdot \rangle_\mathcal{F}$, and a map $\varphi : \mathcal{X} \rightarrow \mathcal{F}$ such that:

$$\forall (x, z) \in \mathcal{X} \times \mathcal{X}, k(x, z) = \langle \varphi(x), \varphi(z) \rangle_\mathcal{F}$$

$\mathcal{F}$ is called a Reproducing Kernel Hilbert Space (RKHS) associated to the kernel $k$, and its existence is assured by the Mercer theorem, shown in [STC04]
\( \mathcal{F} \) is indeed a space function, and (2.1) can be rewritten:

\[
\varphi : \mathbb{R}^d \rightarrow \mathcal{F} \\
x \mapsto k(x, \cdot)
\]  

(2.2)

This enables us to first choose our kernel function instead of defining a feature space and an embedding map \( \varphi \), which may not always be written easily (the feature space can be an infinite dimensional space), as we will show for some examples of kernel functions.

**Definition 2.4 (Gram matrix)** Let \( k \) be a positive kernel in \( \mathcal{X} \) and \( x_1 \cdots x_n \) a set of \( n \) observations. The Gram matrix \( K \) is the \( n \)-dimensional matrix with general term \( K_{ij} = k(x_i, x_j) \).

### 2.2.1 Examples of kernels

In this subsection, some examples of commonly used kernels will be showed. We are assuming that \( x \) and \( z \) are elements from the input space \( \mathcal{X} \).

**Polynomial**

\[
k(x, z) = (x^T z + b)^d
\]

Notice that when \( b = 0 \) and \( d = 1 \), then the kernel is the canonical inner product in the input space.

**Laplacian**

\[
k(x, z) = \exp(-\frac{\|x - z\|}{\sigma})
\]

**Gaussian**

\[
k(x, z) = \exp(-\frac{\|x - z\|^2}{2\sigma^2})
\]

In the previous examples, \( \sigma, s \) and \( b \) are some parameters used to change the influence of the kernel, and their tuning is reportedly more important than the choice of the kernel function itself. In the rest of this thesis, it is essentially the Gaussian kernel that is chosen in our experiments, and we will in particular study the influence of the width \( \sigma \) of the kernel.
2.3 Support Vector Machines

We are now introducing the so-called Support Vector Machine (SVM), in its one-class (or single-class) variant. SVMs are the class of algorithms that recently put the kernel methods into the spotlight.

The aim of the one-class SVM is to evaluate the support of the distribution of a set of vectors, i.e. a region of the input space in which vectors drawn from the same distribution as the training set are likely to fall in. The result of the computation is a function that answers to the question whether a new test point lies in that distribution or not (novelty detection). This approach was presented in [SPST+01]. This is an unsupervised learning task.

This problem is solved by finding the hyper-plane that separates the training points from the origin with the greatest distance. Recall that we are using the Gaussian kernel \( k(x, z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2}) \). Then in the RKHS \( \mathcal{F} \), \( \forall x \in \mathcal{X}, \|\varphi(x)\|_F = \langle \varphi(x), \varphi(x) \rangle_F = k(x, x) = 1 \). The mapping of the input points are thus all located on a hyper-sphere of radius 1. Moreover, \( \forall (x, z) \in \mathcal{X}^2, k(x, z) \geq 0 \), and as a consequence, \( \varphi(\mathcal{X}) \) is located in the positive orthant in the feature space \( \mathcal{F} \), guaranteeing the separability of the data points from the origin and thus the existence of the hyper-plane.

In \( \mathcal{F} \), given the vector \( w \) and the parameter \( \rho \), a hyper-plane can be parameterised as the set \( \{ x \in \mathcal{X} | \langle w, x \rangle - \rho = 0 \} \); the distance of that hyper-plane to the origin is \( \frac{\rho}{\|w\|} \).

So, given a set \( X = \{x_i\}_{i=1}^m \) of \( m \) input vectors, the solution to our problem solves the following optimisation problem:

\[
\min_{w \in \mathcal{F}, \rho \in \mathbb{R}, \xi_i \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 - \rho + \frac{1}{\nu m} \sum_i \xi_i
\]

subject to \( \langle w, \varphi(x_i) \rangle_F \geq \rho - \xi_i \)

and \( \xi_i \geq 0 \)

\( i = 1 \cdots m \) (2.3)

The summation term in the objective function and the \( \xi_i \) (which are called the slack variables) are a penalisation term which allow some of the points to lie on the “wrong” side of the separation hyper-plane, allowing some outliers in the training set of data and the algorithm to be resistant to noise. It can be shown that the \( \nu \) factor asymptotically converges to the proportion of outliers in the training data.

The decision function that will be used to test whether a new point \( z \) is distributed under the same distribution as the training points can then be written with the elements \( w^* \) and \( \rho^* \) that solve the optimisation problem:
To solve the constrained optimisation problem (2.3), we are introducing the Lagrangian and Lagrange multipliers \( \{\alpha_i\}_{i=1}^{m} \) and \( \{\beta_i\}_{i=1}^{m} \):

\[
L(w, \rho, \xi_i, \alpha_i, \beta_i) = \frac{1}{2} \|w\|^2 - \rho + \frac{1}{\nu m} \sum_i \xi_i - \sum_i \alpha_i (\langle w, \varphi(x_i) \rangle - \rho + \xi_i) - \sum_i \beta_i \xi_i
\]  

(2.5)

The Karush-Kuhn-Tucker necessary conditions, i.e. derivatives with respect to each variable of the Lagrangian being equal to 0, can be written as:

\[
\alpha_i (\langle w, \varphi(x_i) \rangle - \rho + \xi_i) = 0
\]  

(2.6)

\[
\beta_i \xi_i = 0
\]  

(2.7)

\[
w = \sum_i \alpha_i \varphi(x_i)
\]  

(2.8)

\[
\sum_i \alpha_i = 1
\]  

(2.9)

\[
\beta_i = \frac{1}{\nu m} - \alpha_i
\]  

(2.10)

(2.8) is important, as it states that the solution to the optimisation problem can be written as a linear combination of the images of the training points in the feature space. In particular, (2.4) becomes

\[
f(z) = \text{sign}(\langle \sum_i \alpha_i \varphi(x_i), \varphi(z) \rangle_F - \rho)
\]  

(2.11)

Finally, the optimisation problem (2.3) becomes:

\[
\min_{\alpha \in \mathbb{R}^m} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j)
\]  

subject to \(0 \leq \alpha_i \leq \frac{1}{\nu m}\) 

\[
\sum_i \alpha_i = 1
\]  

(2.12)

(2.12) is a quadratic optimisation problem, and since the kernel is positive, it is a convex problem, so the existence of a solution is assured, and it is easily solvable with a quadratic programming optimisation algorithm.
Support Vectors  Constraints on \( \alpha_i \) and KKT complementary slackness conditions ((2.6), (2.7) and (2.10)) yield the following cases at the optimum:

- \( \alpha_i = 0 \), the constraint is inactive, then according to (2.10) and (2.7), \( \xi_i = 0 \). The corresponding training vector is located inside the distribution support;

- \( 0 < \alpha_i < \frac{1}{\nu_m} \), then \( \beta_i \) is non null while \( \xi_i \) is. Then we have \( \rho = \sum_j \alpha_j k(x_j, x_i) \). The training vector \( x_i \) is called a margin support vector, and is located on the separating hyper-plane (or decision boundary);

- \( \alpha_i = \frac{1}{\nu_m} \), \( x_i \) is a non margin support vector, which correspond to an outlier.

Notice that the only training vectors that are used to compute the decision boundary are the support vectors (hence the name Support Vector Machine), the ones associated with the non-null Lagrangian multipliers. Most of the \( \alpha_i \) are zero, the sparsity of the solution is the main advantage of SVM algorithms: once \( \alpha \) has been computed– after the costly quadratic programming, though – testing the novelty of a new vector is hardly computer-intensive. The SV approach reduces a problem with possibly thousands of training examples into the selection of a few key-features that best describes the underlying distribution.

2.4 Computing a dissimilarity measure

In this section, we will show the algorithms we used in our change detection framework. To be more precise, these methods provide a measure of dissimilarity in the feature space: given two series of observations \( X = \{x_i\}_{i=1}^n \) and \( Y = \{y_i\}_{i=1}^m \) of the considered signal, we are defining a statistic \( T(X, Y) \) that will later be compared to a threshold to make a decision.

The first algorithm is based on the one-class SVM, while the other two are comparing some statistics in the feature space.

2.4.1 Online Kernel Change Detection

A geometric criterion  Based on the one-class SVM, the Kernel Change Detection algorithm was presented by Desobry et al. [DDD05]. They are using a geometric criterion in the feature space to compare the regions supporting the sets \( X \) and \( Y \).

Using the one-class SVM algorithm with a Gaussian kernel, support region \( \mathcal{R}_X \) (resp. \( \mathcal{R}_Y \)) of vector set \( X \) (resp. \( Y \)) is constructed. In the feature space \( \mathcal{F} \), it corresponds to the hyper-plane defined by the couple \( (w_X, \rho_X) \) (resp. \( (w_Y, \rho_Y) \)).
Consider now the two-dimensional plane \( P \) defined by the vectors \( \mathbf{w}_X \) and \( \mathbf{w}_Y \). The intersection of \( P \) and the hyper-sphere \( S \) of radius 1 – where lie all the images of vectors of the input space by the embedding map \( \varphi \) in \( \mathcal{F} \) – is the circle that we will denote by \( C \).

We are also defining the following elements:

- \( c_X \) (resp. \( c_Y \)), as the intersection between \( C \) and the straight line spanned by vector \( \mathbf{w}_X \) (resp. \( \mathbf{w}_Y \));

- \( p_X \) (resp. \( p_Y \)) as one of the two intersections between \( C \) and the hyper-plane defined by \( (\mathbf{w}_X, \rho_X) \) (resp. \( (\mathbf{w}_Y, \rho_Y) \)).

The proposed dissimilarity measure is:

\[
T_{KCD}(X, Y) = \frac{\widehat{c_X c_Y}}{c_X p_X + c_Y p_Y} \tag{2.13}
\]

The numerator is the distance between the centres of the distributions in the feature space, which is scaled in the denominator by the spread of both \( X \) and \( Y \) (a large arc length \( c_X p_X \) means a short margin \( \frac{\rho_X}{\| \mathbf{w}_X \|} \), and thus a wide spread distribution in the feature space). It is a Fisher inspired inter-regions/intra-regions ratio. This statistics will be larger as the sets are more separated and less overlapping in the feature space, thus marking them as more dissimilar.

---

1 note that if these are colinear, there is an infinity of possible planes, any of them is suitable
2 both are suitable, since what will matter is the length of the arc \( c_X p_X \)
Computation using Gram matrices  Now, let us see how the terms of (2.13) can be calculated in the feature space, using only the Gram matrices relative to the sets $X$ and $Y$. Since the points $c_X$ and $c_Y$ are on a circle of radius one, we can write that $c_X^T c_Y = c_X^T c_Y$, where the latter notation designs the angle corresponding to the arc. This angle measure appears in the following expression:

$$\langle w_X, w_Y \rangle_F = \|w_X\|_F \cdot \|w_Y\|_F \cdot \cos(c_X^T c_Y)$$ (2.14)

The angle here is in $[0, \pi/2]$, since all the points lie in the same quadrant of the circle $C$. We can then safely write that

$$c_X^T c_Y = \arccos\left(\frac{\langle w_X, w_Y \rangle_F}{\|w_X\|_F \cdot \|w_Y\|_F}\right)$$ (2.15)

Using (2.8), and noting $(\alpha_{X,i})$ (resp. $(\alpha_{Y,i})$) the weights of $w_X$ (resp. $w_Y$), we have:

$$\langle w_X, w_Y \rangle_F = \left\langle \sum_{i=1}^{m} \alpha_{X,i} \varphi(x_i), \sum_{j=1}^{m} \alpha_{Y,j} \varphi(y_j) \right\rangle_F$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_{X,i} \alpha_{Y,j} \langle \varphi(x_i), \varphi(y_j) \rangle_F$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_{X,i} \alpha_{Y,j} k(x_i, y_j)$$

$$= \alpha_X^T K_{XY} \alpha_Y$$ (2.16)

where $(K_{XY})_{i,j} = k(x_i, y_j)$. In the same manner, we also have:

$$\|w_X\|_F = \sqrt{\alpha_X^T K_X \alpha_X}$$ (2.17)

Similarly,

$$\langle w_X, \overrightarrow{p_X} \rangle_F = \|w_X\|_F \cdot \|\overrightarrow{p_X}\|_F \cdot \cos(c_X \overrightarrow{p_X})$$ (2.18)

yielding

$$c_X^T p_X = c_X^T \overrightarrow{p_X} = \arccos\left(\frac{\rho_X}{\sqrt{\alpha_X^T K_X \alpha_X}}\right)$$ (2.19)

We finally obtain our dissimilarity measure by putting all things together in (2.13).
Summary To compute the dissimilarity measure $T_{KCD}(X, Y)$ between the sets $X$ and $Y$:

- Compute Gram matrices $K_X$ and $K_Y$;
- run the quadratic program (2.11) on sets $X$ and $Y$, yielding the weights and margins $(\alpha_X, \rho_X)$ and $(\alpha_Y, \rho_Y)$;
- calculate $\|w_X\|_F$, $\|w_Y\|_F$ and $\langle w_x, w_y \rangle_F$ using (2.17) and (2.16);
- calculate $c_X p_X$, $c_Y p_Y$, $c_X c_Y$ and finally $T_{KCD}(X, Y) = \frac{c_X c_Y}{c_X p_X + c_Y p_Y}$

2.4.2 Maximum Mean Discrepancy (MMD)

Let $\mathcal{F}$ be a RKHS; the Maximum Mean Discrepancy \cite{Gretton07}, or MMD, is a test statistic whose aim is to check whether two sample of observations are taken from the same distribution.

It is defined as following:

**Definition 2.5 (mean element and MMD)** Let $p$ and $q$ be Borel probability distributions, and $X$ and $Y$ sets of samples drawn from $p$ and $q$. The mean element $\mu \in \mathcal{F}$ is such as:

$$
\mu_p(x) = \langle \mu, k(\cdot, x) \rangle = \mathbb{E}_{z \sim p}[k(z, x)]
$$

(2.20)

The MMD is then defined as:

$$
\text{MMD}[\mathcal{F}, p, q] = \|\mu_p - \mu_q\|^2
$$

(2.21)

It is intuitively the difference between the means of a function over $p$ and $q$, which will be considered as different if it is a function that has different expectation values on the two distributions.

**MMD in feature space** In a RKHS, the evaluation of functions $f$ of $\mathcal{F}$ can be written $f(x) = \langle f, \varphi(x) \rangle$. The empirical estimate of the expectation of $\varphi(x)$ is:

$$
\hat{\mu}[X] = \frac{1}{n} \sum_{i=1}^{n} \varphi(x_i)
$$
So, an empirical estimate of MMD can be calculated in the following:

\[
\|\hat{\mu}[X] - \hat{\mu}[Y]\|^2_F = \frac{1}{n^2} \left\langle \sum_{i=1}^{n} \varphi(x_i), \sum_{j=1}^{n} \varphi(x_j) \right\rangle_F - \frac{2}{nm} \left\langle \sum_{i=1}^{n} \varphi(x_i), \sum_{j=1}^{m} \varphi(y_j) \right\rangle_F + \frac{1}{m^2} \left\langle \sum_{i=1}^{m} \varphi(y_i), \sum_{j=1}^{m} \varphi(y_j) \right\rangle_F
\]

\[
= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \varphi(x_i), \varphi(x_j) \rangle_F - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \langle \varphi(x_i), \varphi(y_j) \rangle_F + \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \langle \varphi(y_i), \varphi(y_j) \rangle_F
\]

\[
= \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(x_i, x_j) - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, y_j) + \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} k(y_i, y_j)
\]

which can be rewritten with the help of the Gram matrices (\(1\) being the all-ones vector):

\[
T_{MMD}(X, Y) = \left[ \frac{1}{n^2} 1^T K_X 1 - \frac{2}{nm} 1^T K_{XY} 1 + \frac{1}{m^2} 1^T K_Y 1 \right]^\frac{1}{2}
\]

which gives us our test statistic.

### 2.4.3 Kernel Fisher Discriminant Analysis (KFDA)

The KFDA algorithm [HBM08] is extending the approach of the MMD for the same problem, by normalising the difference between the means of the tested sets by the covariance information, a technique that proved to yield good results for algorithms in input space. The KFDA is a kernelized version of the linear Fisher discriminant.

**Linear Fisher discriminant** The linear Fisher discriminant’s aim is to solve a classification problem between two sets \(X^{(1)}\) and \(X^{(2)}\). The idea behind the Fisher discriminant is to find the direction \(w\) that maximise the following quantity:

\[
J(w) = \frac{w^T S_B w}{w^T S_W w}
\]

(2.24)
where

\[
\hat{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T
\]
\[
\hat{S}_W = \sum_{i=1,2} \sum_{x \in X^{(i)}} (x - \mathbf{m}_i)(x - \mathbf{m}_i)^T
\]
\[
\hat{\mathbf{m}}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} x_j^{(i)}
\]

(2.25)

\(J\), which is called a Rayleigh coefficient, is the ratio of the between-class \((\hat{S}_B)\) and within-class \((\hat{S}_W)\) scatter matrices. Hence, we want to maximise the separation of the class means of the projection of the input vectors, while minimising the variance around their means to get the overlap between the classes as small as possible.

**Mean and covariance in reproducing kernel Hilbert spaces** Similarly, we want to find a non-linear discriminant by mapping our input vectors into a feature space in which we will compute the Fisher discriminant, and again, we want to find a solution as a linear combination of the images of the input vectors: \(\mathbf{w} = \sum_{j=1}^{m} \alpha_j \varphi(x_i)\).

We must first introduce the mean element and covariance operator in a RKHS that are needed in the Fisher discriminant based algorithm. By widely spread ([SS02] for example) abuse of notation (writing \(\varphi(x)\varphi(x)^T\) is slightly incorrect since \(\mathcal{F}\) may be infinite dimensional, especially in the Gaussian case), given \(X = (x_i)_{i=1...n}\) the mean \(\mu\) and covariance \(\Sigma\) in \(\mathcal{F}\) can be written empirically as:

\[
\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \varphi(x_i)
\]
\[
\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (\varphi(x_i) - \mu)(\varphi(x_i) - \mu)^T
\]

(2.26)

Between-class covariance becomes:

\[
\Sigma_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T
\]

(2.27)

And we can write our Rayleigh coefficient, with \(\Sigma_W = \Sigma_1 + \Sigma_2\):

\[
J(\mathbf{w}) = \frac{\mathbf{w}^T \Sigma_B \mathbf{w}}{\mathbf{w}^T \Sigma_W \mathbf{w}}
\]

(2.28)

Our test statistic for the two-sample problem is the value of \(J\), associated with the optimal direction \(\mathbf{w}^*\).
Solving the optimisation problem  Maximising (2.28) could be solved as a general eigenvector problem. By defining the Lagrangian and differentiating it with respect to $w$, we obtain:

$$\Sigma_B w - \lambda^\top \Sigma_W w = 0$$  \hspace{1cm} (2.29)

Assuming that $\Sigma_W$ is positive definite, premultiplying by $\Sigma_W^{-1}$, (2.29) becomes:

$$\Sigma_W^{-1} \Sigma_B w = \lambda^\top w$$  \hspace{1cm} (2.30)

By definition, $\Sigma_B$ is a rank one matrix, as a consequence, so is $\Sigma_W^{-1} \Sigma_B$, which then has only one non-null eigenvalue, which can be calculated as its trace. So,

$$T_{KFDA} = \text{Trace}(\Sigma_W^{-1} \Sigma_B) = \langle \mu_1 - \mu_2, \Sigma_W^{-1} (\mu_1 - \mu_2) \rangle$$  \hspace{1cm} (2.31)

**Regularisation**  We previously assumed that $\Sigma_W$ was invertible, which might actually not be true. A workaround to this problem is to add a multiple of the matrix identity to $\Sigma_W$, the statistic becoming:

$$T_{KFDA} = \langle \mu_1 - \mu_2, (\Sigma_W + \gamma I)^{-1} (\mu_1 - \mu_2) \rangle$$  \hspace{1cm} (2.32)

The regularisation factor $\gamma$ also controls the influence of the intra-matrix scatter factor. Depending on its value, it is equivalent to thresholding the eigenvalues of $\Sigma_W$. Hence, for a large $\gamma$, the identity matrix is dominating, and the KFDA statistic becomes equivalent to an MMD statistic. [HBM08] shows the evolution of the performance of the test with varying values of $\gamma$. The KFDA hence needs a careful tuning of that parameter (in addition to the choice of the bandwidth of the kernel in the Gaussian case for example).

**Computation of the test statistics**  Let us consider the two samples $\{x^{(1)}_1, \ldots, x^{(1)}_{n_1}\}$ and $\{x^{(2)}_1, \ldots, x^{(2)}_{n_2}\}$, with $n_1 + n_2 = n$. Denote by $G^{(u)}_n : \mathbb{R}^{n_u} \to \mathcal{H}$, $u = 1, 2$, the linear operators which associates to a vector $\alpha^{(u)} = [\alpha^{(u)}_1, \ldots, \alpha^{(u)}_{n_u}]^T$ the vector in $\mathcal{H}$ given by $G^{(u)}_n \alpha^{(u)} = \sum_{j=1}^{n_u} \alpha_j^{(u)} k(x_j^{(u)}, \cdot)$. This operator may be presented in a matrix form

$$G^{(u)}_n = \begin{bmatrix} k(x_1^{(u)}, \cdot), \ldots, k(x_{n_u}^{(u)}, \cdot) \end{bmatrix}.$$  \hspace{1cm} (2.33)

We denote by $G_n = \begin{bmatrix} G^{(1)}_n & G^{(2)}_n \end{bmatrix}$. We denote by $K^{(u,v)}_n = [G^{(u)}_n]^T G^{(v)}_n$, $u, v \in \{0, 1\}$, the Gram matrix given by $K^{(u,v)}_n(i, j) = k(x_i^{(u)}, x_j^{(v)})$ for $i \in \{1, \ldots, n_u\}$, $j \in \{1, \ldots, n_v\}$. Define, for any integer $\ell$, $P_\ell = I_\ell - \ell^{-1} 1_\ell 1_\ell^T$ where $1_\ell$ is the $(\ell \times 1)$ vector whose components are all equal to one and $I_\ell$ is the $(\ell \times \ell)$ identity matrix and let $N_n$ be given by

$$N_n \overset{\text{def}}{=} \begin{pmatrix} P_{n_1} & 0 \\ 0 & P_{n_2} \end{pmatrix}$$  \hspace{1cm} (2.34)
Finally, define the vector $m_n = (m_{n,i})_{1 \leq i \leq n}$ with $m_{n,i} = -n_1^{-1}$ for $i = 1, \ldots, n_1$ and $m_{n,i} = n_2^{-1}$ for $i = n_1 + 1, \ldots, n_1 + n_2$. With the notations introduced above,

$$
\hat{\mu}_2 - \hat{\mu}_1 = G_n m_n, \quad \hat{\Sigma}_u = n_u^{-1} G_n^{(u)} P_{n_u} P_{n_u}^\top (G_n^{(u)})^\top, \quad u = 1, 2, \quad \hat{\Sigma}_W = n^{-1} G_n N_n N_n^\top G_n^\top,
$$

which implies that

$$
\langle \hat{\mu}_2 - \hat{\mu}_1, (\hat{\Sigma}_W + \gamma I)^{-1}(\hat{\mu}_2 - \hat{\mu}_1) \rangle_H = m_n^\top G_n^\top (n^{-1} G_n N_n N_n^\top G_n^\top + \gamma I)^{-1} G_n m_n.
$$

Recall the matrix inversion lemma, which gives

$$
(\gamma I + UV)^{-1} = \gamma^{-1} I - \gamma^{-2} U (I + \gamma^{-1} V U)^{-1} V
\quad = \gamma^{-1} I - \gamma^{-1} U (\gamma I + V U)^{-1} V
\quad = \gamma^{-1} \left\{ I - U (\gamma I + V U)^{-1} V \right\}.
$$

Then, we get

$$
m_n^\top G_n^\top (n^{-1} G_n N_n N_n^\top G_n^\top + \gamma I)^{-1} G_n m_n
\quad = \gamma^{-1} m_n^\top G_n^\top \left\{ I - n^{-1} G_n N_n \left( \gamma I + n^{-1} N_n^\top G_n^\top G_n N_n \right)^{-1} N_n^\top G_n^\top \right\} G_n m_n
\quad = \gamma^{-1} \left\{ m_n^\top K_n m_n - n^{-1} m_n^\top K_n N_n \left( \gamma I + n^{-1} N_n^\top K_n N_n \right)^{-1} N_n K_n m_n \right\}.
$$

which is our test statistics.
Chapter 3

Experiments on artificially generated data

3.1 Introduction

In this chapter the behaviour of our algorithms is tested on some artificially generated data. We first introduce some elements on detection theory that allow us to have a good performance criterion with the so-called Receiver Operating Characteristics curve. Then we make some first evaluation on some Gaussian mixture data and try to see the resistance of the algorithms to some simple transformations, so as to have a glimpse of their expected performance for future real data experiments.

3.2 Detection theory

3.2.1 Hypothesis testing

Supposing we have two observations $X$ and $Y$, defined by $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ assumed to be distributed according to probability density functions $p$ and $q$ respectively. We want to determine whether $p = q$ or not, that is testing the hypotheses:

\[ H_0 : p = q \]
\[ H_1 : p \neq q \]

$H_0$ is called the null hypothesis, while $H_1$ is the alternative hypothesis.

The general approach to this problem is to define a test statistic $T(X, Y)$ that will be used as a similarity measure between the two distributions, and a threshold $\eta$ to which that statistic will be compared to.
Then the following decisions are made:

\[ H_0 \text{ if } T(X, Y) \leq \eta \]
\[ H_1 \text{ if } T(X, Y) > \eta \]

### 3.2.2 Detection performance

One way to evaluate the performance of a detector and its test statistic is to build a Receiver Operating Characteristics, or ROC curve [Kay98]. You construct the ROC curve by plotting the probability of detection \( P_D \) of the test against the probability of false alarm \( P_{FA} \), for different values of the threshold \( \gamma \). For that matter, consider an experiment with sample points drawn over two distributions. Then the outcome of the tests should become as in table 3.1.

<table>
<thead>
<tr>
<th>Predicted negative</th>
<th>Predicted positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual negative</td>
<td>True Negative (TN)</td>
</tr>
<tr>
<td>Actual positive</td>
<td>False Negative (FN)</td>
</tr>
</tbody>
</table>

Table 3.1: Confusion matrix

Then, on the data set you’re evaluating your test, an estimate of the probability of detection is \( P_D \) and of the probability of false alarm \( P_{FA} \) are the True positive rate, and False positive rate respectively.

\[ TPR = \frac{TP}{TP + FN} \]
\[ FPR = \frac{FP}{FP + TN} \]

The ROC curve is formed as these values are plotted as coordinate pairs across the range of possible values of \( \eta \). An example of a ROC curve is showed in figure 3.1.

The area under the curve (AUC) is a useful overall indicator of the performance of a test statistic. The closer it gets to 1.0, the better the test is, while a value of 0.5 indicates the performance of a random decision (AUC of the diagonal).

### 3.3 Simulations

This section is dedicated to simulations of the algorithms previously presented on some artificial data sets.
3.3.1 Test protocol

The general framework for the series of tests is as following:

- draw 25 observations of $N$ points distributed according to a distribution $p$;
- draw another 25 sets of $N$ points, distributed according to $q$, $q \neq p$;
- for each pair $(X_i, Y_j)$ of observations taken from the 50 sets just generated, compute the test statistic $T(X_i, Y_j)$;
- for $\eta$ varying in the range $[\min_{i,j} T(X_i, Y_j), \max_{i,j} T(X_i, Y_j)]$, decide $\mathcal{H}_0$ or $\mathcal{H}_1$;
- compute true positive and false positive rates for each $\gamma$ and plot TPR against FPR to obtain the ROC curve. The area under curve is computed using the trapezoidal numerical integration method.

This process was applied on the different kernel methods we presented. Some meta-parameters ($\sigma$, the width of the Gaussian kernel for all the algorithms, $\nu$, rate of outliers for SVM-based algorithms, $\gamma$, regularisation factor for KFDA) have to be set, and there is no clear general rule on how to tune them. So, grid search, a common method for parameter selection was used over the parameter space on a selected range:

- $\sigma$ in $d \ast [0.5 0.8 1 1.5 2 5]$, where $d = E(||x - x'||)$, $(x, x') \in X$ is the average distance between all pairs of points in the observation; using such a $\sigma$ is a general rule of thumb used in [DDD05] for example.
- $\nu$ in $[0.001 0.01 0.05 0.1 0.2 0.4]$
• \( \gamma \) in \([0.1 1 2 5 10]\)

The top five results for each algorithm were then kept and the associated ROC plotted.

### 3.3.2 Data sets

For all the toy sets we generated, we used Gaussian mixture model distributions using the \textit{H2M} [Cap01] Matlab toolbox.

\begin{figure}[h]
\centering
\begin{subfigure}{0.45\textwidth}
\includegraphics[width=\textwidth]{fig1.png}
\caption{\(p \) vs \(q_1\)}
\end{subfigure}\hspace{1cm}
\begin{subfigure}{0.45\textwidth}
\includegraphics[width=\textwidth]{fig2.png}
\caption{\(p \) vs \(q_2\)}
\end{subfigure}
\caption{Visualisation of the 1-class SVM decision boundaries in the “cross” distributions}
\end{figure}

**“Cross” distribution** The first distribution was a mixture of two 2-dimensional centred normal distributions:

\[
p \sim \frac{1}{2} \mathcal{N}(0, \Sigma_1) + \frac{1}{2} \mathcal{N}(0, \Sigma_2)
\]

such that \( \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 0.2 \end{bmatrix} \) and \( \Sigma_2 = \begin{bmatrix} 0.2 & 0 \\ 0 & 4 \end{bmatrix} \)  

As an alternative distribution, \( q_i \) is the same as above with some small modifications applied to it:

- \( q_1 \sim \frac{1}{2} \mathcal{N}(\mu, \Sigma_1) + \frac{1}{2} \mathcal{N}(\mu, \Sigma_2) \), with \( \mu = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \), i.e. a shift in the NE direction,

- \( q_2 \sim \frac{1}{2} \mathcal{N}(0, \Omega \Sigma_1 \Omega') + \frac{1}{2} \mathcal{N}(0, \Omega \Sigma_2 \Omega') \), \( \Omega = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \) being a rotation matrix with angle \( \theta \).
These distributions were used as they are in 2-dimensional spaces, and also in 9-dimensional spaces, where the seven remaining components were padded with compatible white Gaussian noise.

Figure 3.2 shows these distribution and the decision boundaries computed by the 1-class SVM.

**“Blobs distribution”** The “blob” is also a Gaussian mixture, the Gaussians being quite well separated.

\[
\rho_b \sim \frac{1}{2} \mathcal{N}(\mu_1, \Sigma_1) + \frac{1}{2} \mathcal{N}(\mu_2, \Sigma_2)
\]

\[
\Sigma_1 = \begin{bmatrix} 2 & 0.3 \\ 0.3 & 0.5 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}, \mu_1 = (1, 2) \text{ and } \mu_2 = (-3, -5)
\]

To generate the alternative distribution \(q_b\), we simply modified the mean of the first Gaussian, \(\mu_1\) becoming \(\hat{\mu}_1 = (2, 2)\). As in the previous case, we also increased the dimensionality of this set of data by adding seven extra coordinates padded with white Gaussian noise.

### 3.3.3 Results and interpretation

In all examples, we changed either the number of points in each observations (either 50 or 150), the dimensionality (2 or 9) or the transformation applied to the original distribution. A small number of points, while reducing the accuracy of a test, allows us to get results closer to what we may get in real audio data sets in a segmentation problem. In the same manner, the real data sets will have a large number of features, so having a high dimensional input space allows us to evaluate the algorithms’ robustness against an increase of the dimension of the input, whereas a 2D space enables an easier visualisation.

**“Cross” distribution** In figures 3.3, 3.4, 3.5 and 3.6, we displayed the ROC curves we got on various observations of different couples of distributions.

\(\rho\) against \(q_1\) The first couple we tested was an observation of \(\rho\) and \(q_1\) (the mean was slightly shifted towards the North-East direction), with 150 points in a 2-dimensional space. We obtained perfect guesses with all algorithms as each observation was correctly labelled with the correct distribution it was drawn from. Thus the AUCs were the perfect 1.0.
The same test was then conducted with 50 points per observation, and results are displayed on figure 3.3. The algorithms behave quite well (expectedly giving worse results than the 150-point test, due to the smaller size of the sample), even if the KCD algorithm slightly underperforms the three others. KCD’s best result is however achieved with meta-parameters \( \sigma \) and \( \nu \) at the edge of the grid-search parameters space, so we may get better results by extending that space. The KFDA test yields better results than MMD and SVM-based algorithms.

\[ \begin{align*}
AUC &= 0.92552 \\
\sigma &= 1.1452, \quad \nu = 0.01 \\
AUC &= 0.92552 \\
\sigma &= 1.1452, \quad \nu = 0.05 \\
AUC &= 0.92552 \\
\sigma &= 1.1452, \quad \nu = 0.1 \\
AUC &= 0.91909 \\
\sigma &= 1.1452, \quad \nu = 0.4
\end{align*} \]

\[ \begin{align*}
AUC &= 0.90189 \\
\sigma &= 2.2904
\end{align*} \]

\[ \begin{align*}
AUC &= 0.89999 \\
\sigma &= 1.8323
\end{align*} \]

\[ \begin{align*}
AUC &= 0.89973 \\
\sigma &= 3.4356
\end{align*} \]

\[ \begin{align*}
AUC &= 0.89511 \\
\sigma &= 4.5808
\end{align*} \]

\[ \begin{align*}
AUC &= 0.88979 \\
\sigma &= 1.1452
\end{align*} \]

\[ \begin{align*}
AUC &= 0.96255 \\
\sigma &= 1.8323, \quad \gamma = 0.1 \\
AUC &= 0.96072 \\
\sigma &= 2.2904, \quad \gamma = 0.1 \\
AUC &= 0.95333 \\
\sigma &= 1.1452, \quad \gamma = 1 \\
AUC &= 0.95086 \\
\sigma &= 1.1452, \quad \gamma = 0.1 \\
AUC &= 0.94614 \\
\sigma &= 1.8323, \quad \gamma = 1
\end{align*} \]

\[ \begin{align*}
AUC &= 0.96255 \\
\sigma &= 1.8323, \quad \gamma = 0.1 \\
AUC &= 0.96072 \\
\sigma &= 2.2904, \quad \gamma = 0.1 \\
AUC &= 0.95333 \\
\sigma &= 1.1452, \quad \gamma = 1 \\
AUC &= 0.95086 \\
\sigma &= 1.1452, \quad \gamma = 0.1 \\
AUC &= 0.94614 \\
\sigma &= 1.8323, \quad \gamma = 1
\end{align*} \]

\[ \begin{align*}
AUC &= 0.96255 \\
\sigma &= 1.8323, \quad \gamma = 0.1 \\
AUC &= 0.96072 \\
\sigma &= 2.2904, \quad \gamma = 0.1 \\
AUC &= 0.95333 \\
\sigma &= 1.1452, \quad \gamma = 1 \\
AUC &= 0.95086 \\
\sigma &= 1.1452, \quad \gamma = 0.1 \\
AUC &= 0.94614 \\
\sigma &= 1.8323, \quad \gamma = 1
\end{align*} \]

\[ \begin{align*}
AUC &= 0.96255 \\
\sigma &= 1.8323, \quad \gamma = 0.1 \\
AUC &= 0.96072 \\
\sigma &= 2.2904, \quad \gamma = 0.1 \\
AUC &= 0.95333 \\
\sigma &= 1.1452, \quad \gamma = 1 \\
AUC &= 0.95086 \\
\sigma &= 1.1452, \quad \gamma = 0.1 \\
AUC &= 0.94614 \\
\sigma &= 1.8323, \quad \gamma = 1
\end{align*} \]

The \( p \) distribution was then once again tested against \( q_1 \), with 150-point samples in a 9-dimension input space. ROC curves are showed in figure 3.4. KFDA clearly outperforms the other algorithms for such high-dimensional data.

\( p \) against \( q_2 \) (rotation of \( \theta = \pi/8 \)) With 150 points in a space of dimension 2 (figure 3.5), with the set of parameters selected, all but MMD yields very good results (AUC of 1.0). The latter has a good AUC of 9.7, with the smallest value of \( \sigma \) we
Figure 3.4: 9D distribution, 150 points, \( q_1 \)

tested, so we may achieve better results by decreasing that parameter more.

We tested the same distribution, then with 50 points per observation (figure 3.6). The KCD SVM-based algorithm gives the best results of all four and seems to be less affected by the addition of noise in the extra dimensions. MMD is still behind: we tried smaller values of \( \sigma \), and while the area under the curve increased to 0.73 when \( \sigma \) reached 0.68, a further decrease of the parameter gave worse results.

We also tested an observation of 150 points 9D-distributions, \( p \) and \( q_2 \), \( \theta = \pi/4 \) (and also other values of \( \theta \) that would make \( p \) and \( q_2 \) more similar). No matter what algorithm (with any meta-parameters) was used, the ROC curve was close to the diagonal and the area under the curve around 0.5, which means that they completely failed in evaluating the hypothesis, not performing better than a totally random decision.

**“Blobs” distribution** In the same manner as with the previous examples, the KFDA algorithm exhibits better performance than all other algorithms: it gets absolutely a
better ROC and is more resistant to the addition of extra, noisy dimensions.
Figure 3.6: 2D distribution, 50 points, $q_2, \theta = \pi/8$
Figure 3.7: 2D distribution, 50 points, “blobs”
Figure 3.8: 9D distribution, 150 points, “blobs”
Chapter 4

Application to onset detection in audio files

4.1 Introduction and background

Music information retrieval (MIR) is a hot topic in the scientific community. As an interdisciplinary science, it involves computer science, signal processing, sociology, etc. Research is very active in this field and it concerns various subjects like indexing, transcription, compression, transformation of music.

Onset detection is the process where one is looking for the time instants of a music note. It can be one step in a larger framework like rhythm analysis or music retranscription, i.e. a kind of music reverse engineering.

Many approaches have been tried to tackle the problem of onset detection, but we propose a less traditional method with a machine learning approach based on kernel methods.

4.2 Related work

[BDA+05] provided a complete survey on onset detection methods. All have in common the computation of a reduction of the original signal into a downsampled version which however preserved enough information to enable the retrieval of a musical profile of the studied piece of music.

Onset detectors can basically be divided into two categories: deterministic methods and statistical methods. The former uses the characteristics of the signal like a time-frequency or spectral representation to find the onsets, after a differentiation phase. The latter is based on the construction of probability models to try to guess potential onset times.
Recently, machine learning techniques using support vector machines have been tried to compute a detection function, this is the approach we take in this chapter.

4.3 Onset detection

4.3.1 General scheme

The approach for the onset detection problem we have chosen is the one described by [DDDO5], an online (or sequential) approach. Given a signal, a set of descriptors \( X = \{x_i\}_{i=1}^{n} \) are extracted. These features can be based for example on a time-frequency representation, on energy, phase, amplitude or on Fourier spectral coefficients. Then, a detection function \( D_f(t) \) which is a reduced version of the original signal is computed using these descriptors. The peaks of the onset detection function shall then coincide with note onset times. This is based on the principle that a note onset shall correspond to a change in the characteristics of the signal.
The onset times are eventually found in the peak-picking procedure, which consists of a comparison with a threshold and a selection of relevant points.

Figure 4.1 shows the general flowchart of the onset detection framework: from the original signal, a set of features are extracted (section 4.3.2) which will be used to derive a detection function (4.3.3) that will help us find the onset times by peak-picking (4.3.4).

4.3.2 Feature selection

In this section, we will show how the components of the feature vectors used in the dissimilarity measure algorithms are constructed.

A weighted cepstrum based parametrisation

Cepstral based features were originally developed and widely used in the speech processing community. But this technique has recently been adapted for music information retrieval tasks and been proved to show good results for modelling music [Log00]. [COCM01] propose a two-step weighted least-squares procedure to compute the cepstral coefficients, resulting in a parametrisation that is resistant to noise and high pitch, when compared to a traditional auto-regressive approach.

The cepstral representation of a signal is used to estimate its spectral envelope. The cepstral coefficients are basically a projection of the logarithm of the spectrum of the signal on a basis of sinusoidal functions. Figure 4.2 shows the first functions of this basis. It is noticeable that the first coefficient $c_0$ has a special meaning: it represents the signal level. $c_1$ can also be interpreted as a spectral tilt.

Thus we can write the cepstral parametrisation $S$:

$$S(\omega_k) = s_k = \exp \left( c_0 + 2 \sum_{n=1}^{p} c_n \cos \omega_k n \right)$$

(4.1)

where $c = (c_0, \cdots, c_p)^T$ are the cepstrum coefficients.

The cepstrum coefficients $c$ are computed through two steps. First a rough approximation of the signal is estimated with a least-squares minimisation method with penalisation of the higher order sinusoids which are more sensitive to high pitch errors. The result is re-injected into the minimisation problem as the level of noise in the local signal-to-noise ratio that is used as the weight in our least-squares problem.

This yields to the following algorithm:

\[\text{the linear scale is perceptually less meaningful than the log-scale in speech and music spectra}\]
Algorithm: Cepstral estimation based on the Gaussian estimation

1. Compute the penalised least-squares solution

\[ \hat{c} = (C^T C + \lambda R)^{-1} C^T v \]  

(4.2)

where \( C \) is the cepstrum regression matrix,

\[ C = \begin{bmatrix} 1 & 2 \cos(\omega_1) & \cdots & 2 \cos(\omega_1 \rho) \\ \vdots & \vdots & & \vdots \\ 1 & 2 \cos(\omega_L) & \cdots & 2 \cos(\omega_L \rho) \end{bmatrix} \]  

(4.3)

\( R = 2 \cdot \text{diag}(0, 1^{2r}, 2^{2r}, \cdots, p^{2r}) \) is the diagonal regularisation matrix, \( \lambda \) a smoothing parameter and \( v = (\log x_1, \cdots, \log x_K)^T \) is the vector of log-power measurements.

2. Compute the weights \( \gamma_k = \hat{s}_k / n_k \) for \( k = 1, \cdots, K \), where \( \hat{s}_k \) is computed using (4.1) from the vector of cepstral coefficients \( \hat{c} \) estimated using (4.2).

3. Solve the penalised weighted least-squares problem by

\[ \hat{c} = (C^T \Gamma C + \lambda R)^{-1} C^T \Gamma v \]  

(4.4)

where \( \Gamma = \text{diag}(\gamma_1, \cdots, \gamma_K) \) is the diagonal matrix of weights.

Figure 4.2: First 16 cosine basis functions
Delta features

Given a set of \( N \) already computed features, another set of \( N \) features can be computed. The delta \( D_k = (d_i^{(k)})_{i=1\ldots l} \) (where \( l \) is the length of the signal) of the feature \( X_k = (x_i^{(k)})_{i=1\ldots l} \) is an approximation of the derivative of the feature.

\( d_i^{(k)} \) is the slope of the straight line calculated by the linear least-squares method for points in a \( 2m \)-wide window \( (x_{i-m+1} \cdots x_{i+m-1}) \) around \( x_i^{(k)} \).

![Figure 4.3: First cepstral coefficient of a 6-sec. extract of a piano tune and its associated delta (blue bold line). \( m = 3 \)](image)

4.3.3 Reduction

The generation of the onset detection function is done using the following procedure: at the time position \( t \) in the signal, \( \mathbf{x}_{t,1} = \{x_i\}_{i=t-m_1 \ldots t-1} \) and \( \mathbf{x}_{t,2} = \{x_i\}_{i=t \ldots t+m_2-1} \) are the sets of descriptors of length \( m_1 \) and \( m_2 \) before and after current position \( t \), as in figure 4.4. These two sets are compared using a dissimilarity measure, a statistic \( T \) that can viewed as a “distance” between the two underlying distributions from which \( \mathbf{x}_{t,1} \) and \( \mathbf{x}_{t,2} \) are drawn: \( D_f(t) = T(\mathbf{x}_{t,1}, \mathbf{x}_{t,2}) \). The algorithms we described in section 2.4 provide such statistics and [DDD05] precisely use the KCD (2.4.1) in their onset detection algorithm.

That window of length \( m_1 + m_2 \) is slid throughout the whole signal to get the detection function.
4.3.4 Peak-picking

The last step in the onset detection framework is to extract the onset times from the detection function $D_f$. An ideal detection function should present some sharp and well localised peaks that would precisely identify onsets. However, a single detection function presents some variation in size and shape, and several signals do not produce detection functions with the same characteristics. So, just picking some outliers in $D_f$ is generally not enough to find the onsets in the signal. Some post-processing is then required: a thresholding and the selection of the relevant points.

Thresholding

Following the line of the hypothesis testing, the problem of the onset detection can be reduced to:

$$H_0 : D_f(i) \leq \eta(i)$$
$$H_1 : D_f(i) > \eta(i)$$

The points fulfilling $H_1$ are candidates to be a detected onset.

Multiple strategies can be chosen to fix the threshold $\eta(i)$:

- $\eta_i = \eta$ is a fixed positive constant. This is the simplest solution, but has the disadvantage that for $D_f$ with high dynamics, the detector may miss a lot of onsets in parts with a low mean or produce a lot of false positives in parts where the threshold is too low.

- An adaptive threshold may then be needed. Two strategies were used in our experiments, both are based on moving quantiles of the detection function that is in a window of length $M$ centred on the current time position. If $q_K$ denotes the $K^{th}$ percentile, we have the following:
One strategy is based on the $K^{th}$ percentile in the window:

$$\eta(t) = q_{K}\{D_f(i)\}_{i=t-M/2\ldots t+M/2}^2$$ (4.6)

- the other one, inspired by the construction of box-plots, is based on interquantile range:

$$\eta(t) = q_{25} + \lambda q_{50-25}\{D_f(i)\}_{i=t-M/2\ldots t+M/2}$$ (4.7)

Moving quantile was preferred over a moving mean as it less influenced by outliers with a large value that would noticeably modify the threshold in the case of the mean.

**Peak detection**

Not all the points above the selected threshold may eventually be chosen as onsets. Obviously, only the local maxima should be selected as onsets. In our implementation, we considered as a local maximum a point at time $t$ of the detection function if the following proposition holds, given an integer $w$:

$$\forall i \in [t-w, t+w], i \neq t \Rightarrow D_f(t) > D_f(i)$$ (4.8)

i.e. $D_f(t)$ is greater than all the points within a distance of $w$. We have also chosen to discard the peaks (local maxima selected with the previous criterion) that are too close to another peak with a greater magnitude. Two onsets may be close, but only the greater shall be perceptually significant. It is acknowledged that a careful listener shall not distinguish two onsets that are within about 50ms.

The following algorithm is used to select the most relevant peaks:

**Algorithm: peak-picking** Input: detection function $D_f$, list of the local maxima, number of frames $w$ for which we want to separate the peaks.

1. sort the peaks by order of magnitude, while keeping their position in the detection function;
2. scan that sorted list of peaks in descent order;
3. if $w$ is greater than the difference between the index of the current peak and the index of one previous (bigger) peak, then discard current peak

The resulting list of peaks is the list of onsets of the signal.

\footnote{for the sake of clarity, we are not going into the details of the rounding of the indices}
4.4 Elements of computational complexity

Let us review the computational complexity of the online onset detectors. To compute a detection function, a time index scans the whole signal. At each of the $L$ frames, a dissimilarity measure is calculated from $m$ descriptors, which are $n$-dimensional vectors.

So the complexity $c$ of the computation of the detection function is:

$$c(L, m) = n \cdot O(f(m))$$

(4.9)

where $f$ is a function that depends on the dissimilarity measure function used.

Each of the kernel-based methods have in common that they compute a Gram matrix, which is $O(m^2)$. Then depending on the dissimilarity measure, we get different complexities.

**MMD** Recall equation (2.22), the MMD is only a weighted summation of the elements of the Gram matrix, $f_{\text{MMD}}(m) = O(m^2)$.

**KFDA** The KFDA also involves a weighted summation of the elements of the Gram matrix, but the most computer intensive operation in the calculation of the dissimilarity measure is a matrix inversion, a rough approximation of the KFDA complexity is $f_{\text{KFDA}}(m) = O(m^3)$. This result can however be improved with some optimisations, like using a Cholesky transform to reduce the number of operations.

**KCD** The one-class SVM based algorithm requires the computation of the support vectors for the two sets compared, which involves two quadratic optimisation processes. The actual number of calculations mostly depends on the number of support vectors, i.e. the number of active constraints in the KKT conditions.

In practise, computing the dissimilarity measure using the KCD takes far more time than with the KFDA (and thus the MMD),

**Conclusion** We have three different algorithms with different computation times. However this complexity only depends on a small number of elements, $m$, which is the width of our working window and the online framework scales well with the total length of the signal. Consequently, providing that computing the dissimilarity measure takes less time that the actual length in seconds of the window – which has always been the case in our experiments – the onset detector can be implemented to operate in real time.
Chapter 5

Results

5.1 Implementation details

The algorithms we used in our experiments were implemented in the MATLAB language and the simulations were run under the version 2007a (7.4) of the software.

Signal processing  The music signals, originally monaural and sampled at 44.1 kHz, were downsampled to 16 kHz, centred and normalised by the maximum value of the signal. Computing the cepstral coefficients (4.3.2) first requires the calculation of the short-time Fourier transform (STFT) with a Hanning window of length 512 and a hopsize of 128. Recall that the STFT of a signal \(x\) is defined as:

\[
\tilde{X}(m, k) = \sum_{n=0}^{N-1} w(n)x(Mm + n)e^{-j2\pi kn/N}
\]  

where \(m\) is the frame number, \(k\) the frequency bin index, \(w\) a finite window of length \(N\) (Hanning window in our case) and \(M\) the hopsize.

With the chosen parameters, one frame in the feature vectors or the detection function represents \(T = 128/16000 = 8\text{ms}\).

Feature extraction  The main features we used were the set of the first 20 cepstral coefficients. We also chose to calculate the delta features that could be used as standalone features or concatenated to the cepstral coefficients to form an augmented feature vector of dimension 40.

The feature vectors, whether cepstral, delta or both, were normalised in a very particular way before being used in the dissimilarity measure:

\[
m_X = \frac{2}{L(L+1)} \sum_{i<j<L} \|x_i - x_j\| 
\]  

43
The set \( X = (x_i)_{i=1}^{L} \) (where \( L \) is the total number of frames in the signal) is normalised by \( m_X \) which is the average pairwise distance of all the vectors extracted from the signal. The same is applicable for the set \( D \) of delta features, and if both \( X \) and \( D \) form the final feature vector, then it becomes \( \left[ \frac{X^T D}{m_X m_D} \right]^T \).

**Kernels** This particular normalisation was done because of the calculation of the Gram matrix that then follows. Recall that for the Gaussian kernel, that we use in our experiments, \( k(x, z) = \exp\left(-\frac{1}{2\sigma^2}\|x - z\|^2\right) \). Choosing a value of the kernel bandwidth \( \sigma \) is choosing a multiplicative coefficient to \( m_X \), which is the approach taken by [DDD05]. Some preliminary experiments also showed us that \( m_X \) was a critical value around which the shape of the detection function greatly changed. We have chosen values of that coefficient in the range \([0.4–2]\). It is however noticeable that computing the average pairwise distance was quite computer intensive (quadratic complexity in the number of frames).

**Dissimilarity measures** The number of feature vectors we take to compute the dissimilarity measure \((m_1 \text{ and } m_2 \text{ in 4.3.1}) \) has to be chosen. We picked the values of 10, 20 and 30 for both immediate past and immediate future subsets, which corresponds to 80, 160 and 240 ms. with our STFT parameters. The choice of a value of that window length is not totally harmless: a large number of examples is generally preferable to get accurate results, but a large analysis window may smooth and hide events that are shorter than the window length.

The kernel-based methods were written using the SVM toolbox [CGGR05] for the computation of the Gram matrices.

**Thresholding** Our thresholding methods involved a moving quantile window, whose size must be chosen. We fixed the size of this window to 200 frames, \( i.e. \) 1.6 seconds to allow the threshold to follow the general trend of the signal without smoothing the peaks. Some padding with the length of the window was added before and after the detection functions so as to cope with the absence of data to calculate the quantiles. This padding is some low variance Gaussian noise with the same mean as the beginning (resp. end) of the detection function.

### 5.2 Performance evaluation

To examine the performance of the algorithms presented in the subsequent chapters, a corpus of annotated audio files that gives us ground-truth information is used. This section presents that database and the evaluation procedure.
5.2.1 Audio database

The database was formed using samples collected from the Music Technology Group at Pompeu Fabra University (Barcelona, Spain) and from Centre for Digital Music at Queen Mary University (London, Great-Britain). The 206-sample collection’s global length is 1513 seconds (average sample length 7.34 seconds) and each sample was annotated by one expert listener that marked for a total of 4730 onsets corresponding to music notes.

The collection is composed of solo performance of various percussive and pitched instruments: brass winds (saxophone, trumpet), wood winds (flute and clarinet), bowed (violin and cello) and plucked strings (guitar, bass), piano, solo voices.

5.2.2 Evaluation procedure

The evaluation procedure of the algorithms is the one used in the annual audio onset detection contest \(^1\) from the Music Information Retrieval Evaluation eXchange (MIREX):

The detected onset times will be compared with the ground-truth ones. For a given ground-truth onset time, if there is a detection in a tolerance time-window around it, it is considered as a correct detection (CD). If not, there is a false negative (FN). The detections outside all the tolerance windows are counted as false positives (FP). Doubled onsets (two detections for one ground-truth onset) and merged onsets (one detection for two ground-truth onsets) will be taken into account in the evaluation. Doubled onsets are a subset of the FP onsets, and merged onsets a subset of FN onsets.

The data used in the MIREX contest is re-used each year, that is why their corpus is not publicly available.

The onsets are labelled by hand, which is quite an inaccurate and lengthy operation. A time window around the ground-truth onsets is thus chosen to allow a degree of tolerance because of the inaccuracy of the hand labelling process. This window can depend on the type of the instrument as an onset may be more or less well defined for a percussion beat or a note performed by a violin but we have chosen a fixed tolerance window of 70 milliseconds for all the categories of sounds of our database.

\(^1\)http://www.music-ir.org/mirex2006/index.php/Audio_Onset_Detection
To compare the different algorithms, we define the precision ($P$), recall ($R$) and F-measure ($F$) as performance indicators:

\[
P = \frac{CD}{CD + FP} \quad \text{(5.3)}
\]

\[
R = \frac{CD}{CD + FN} \quad \text{(5.4)}
\]

\[
F = \frac{2PR}{P + R} \quad \text{(5.5)}
\]

$F$ is actually the harmonic mean between precision and recall (the harmonic mean tends strongly towards the least element, so having a large precision but a small recall yields to a bad F-measure).

### 5.3 Examples

![Waveform](a) Waveform ![Spectrogram](b) Spectrogram

Figure 5.1: Waveform and spectrogram of a mandolin tune

As an illustration of the onset detection framework a sample example is taken from our audio corpus. It is a 3.6-second long solo music piece of mandolin, a plucked string instrument. Figure 5.1 shows its waveform and its spectrogram.

#### 5.3.1 Dissimilarity measures

Let us first exhibit the detection function generated with the help of the three dissimilarity measures we presented earlier. In figure 5.2, windows and kernel parameters
Figure 5.2: Detection functions. $w = 10$, $\sigma = 0.5$.
From top to bottom: MMD, KCD and KFDA ($\gamma = 10^{-3}$). Red vertical lines are ground truth onsets.

were set to the same values, i.e. no delta features were used, the length of the sliding window is $m = 20$ and the kernel bandwidth is $\sigma = 0.5m_X$.

First observations tells us that the approach taken is not illegitimate: most of the prominent peaks correspond to real onsets. The KFDA produces sharper peaks than the other two algorithms and seems to be less sensitive to the difficulties that KCD and MMD encountered, like the false positive after time 1.5 sec, or the obvious false negative after the onset at 2.5 sec.

5.3.2 KFDA: influence of $\gamma$

In figure 5.3, we show detection functions of the mandolin piece built by the KFDA-based detector, with different values of the regularisation factor $\gamma$. We notice some sharper peaks as $\gamma$ decreases, allowing the potential onsets to be recognized more easily, as their magnitude grows quickly, compared to the values of non-onsets.

As $\gamma$ takes larger values, the identity matrix dominates in the Rayleigh ratio, and
the dissimilarity measure tends to a MMD-based one, with a lower power.

We will show later that this trend will also be visible when we run our algorithms on a whole corpus of annotated audio.

## 5.4 Performance comparisons

Every year at the MIREX conference, many music information retrieval tasks are proposed to the community. That contest enables the comparison of state-of-the-art algorithms. An audio onset detection challenge is thus proposed. Using the data from MIREX would be a great opportunity to evaluate our framework by comparing it to previous contestant’s results, but as mentioned earlier, the MIREX database is not
publicly available.

Instead, our results will be compared to other algorithms developed internally by other teams at Telecom ParisTech, with our own evaluation procedure (similar to MIREX’s, though).

**Spectral Energy Flux** [ARD05, Are06] presents an enhanced version of the *Spectral Energy Flux* to calculate a detection function that is used to find note onset times. This algorithm was used as one phase in a rhythm analysis task and won the audio tempo extraction contest at MIREX ’05.

This method is based on the assumption that an onset is characterised by a change in the signal’s spectrum. First, the signal is divided into several frequency channels using a STFT (5.1). The power envelope of each frequency bin is then calculated and smoothed. The first order difference in each band is then taken using a digital differentiator and the detection function is eventually the summation of the results in each band.

$C_0 \delta$ We also ran the evaluation on a non-kernel algorithm, one that simply construct a detection function as the delta of the first cepstral coefficient.

### 5.4.1 Detection function comparison

**Synthetic results**

Figure 5.4 shows a synthetic view of the results of our experiments. We plotted recall against precision for the set of parameters $(\sigma, \gamma, \cdots)$ of each algorithm that gave the best F-measure, only varying the parameter setting the threshold level ($K^{th}$ quantile or multiplier of the interquantile range).

No matter which dissimilarity measure was used, the online approach we have taken is clearly outperformed by the spectral energy flux based detection function, and even by the first cepstral coefficient’s delta.

**Results by sound category**

In this part, we exhibit the performance of the algorithms by music category. For each algorithm, the best results by category were selected, thus they do not always correspond to the same set of parameters as the ones used to show the overall best performance in section 5.4.1.

As with the general results on the whole database of sounds, MMD and KCD are clearly outperformed by all the other algorithms. The results are more qualified for KFDA and SEF.
Figure 5.4: Precision - recall curves for all algorithms
Points with the best F-measure are highlighted

Table 5.1: Numeric results by category
In each category, we show precision, recall and F-measure for each method

The detection of percussive instruments’ onsets seemed to be an easier task for all the algorithms. These onsets represented more than two thirds of the whole database, which by the way may have led to greater results for the F-measure on the whole corpus.

Electro pieces gave very bad results for all our algorithms. By examining the resulting detection function, sharp peaks, well localised at the real onset times are however present. But the problem here is that the notes are separated by more than 500 frames of low level noise (which are actually supposed to be silence), which is beyond the 200 frames length we chose to calculate the threshold, yielding a lot of false positive detected onsets and catastrophic precision. We can see that the thresholding and peak-picking steps are critical. We also notice that on these particular examples, offsets are also detected as the performer keeps playing the note for a long period of
time (larger than the analysis window length). This is particularly true when delta features are used to form the feature vectors. The delta of $C_0$ for example shows a visible negative peak at the offset time.

Another difficult task is the detection of bowed strings onsets. This is known to be a non-trivial problem and does not usually yield very good results: the average F-measure over all the contestants at MIREX ’05 and ’06 were 0.518 and 0.544 respectively. Indeed these instruments can produce long and soft attacks as well as stronger ones. Vibratos are also problematic since the spectrum does not remain stationary as the energy rather oscillates around the note’s frequencies.

Although the SEF based onset detector overall performs better than the KFDA-based one, in some categories their rank is modified, which invites us to be careful to the repartition of a corpus of music into different categories. Some algorithms may be tuned or trained to yield better results with a certain category of files and may then have an overall worse performance when tested on an other set of files.

What generally penalises the overall performance is the usually low precision. What we saw with the electro samples was also a problem with other types of sounds: the peaks were present and sometimes could really discriminate an onset from a non-event, but the level of the threshold was often not appropriate.
5.4.2 Influence of the parameters on MMD

In figure 5.6 we plotted recall against precision for the MMD-based onsets detection functions tested on pitched phrases. We chose to limit ourselves to non percussive instruments because the performance were not as good as with percussions, which is where there could be room for improvement. All the points were grouped when they had the same window analysis length and the same sets of features for the computation of the kernel matrices. As a matter of fact, varying those parameters affected the results more than varying the bandwidth of the Gaussian kernel or the level of the threshold, allowing an easier visualisation.

From these figures, we can make a few remarks. First, a longer analysis window yielded better results than a shorter one. Then using the cepstral coefficients was more efficient than using the delta features. Using both gave the worst results.
Of course, these statements are true for the overall set of music, it would be interesting to test the behaviour of the MMD on the different types of music.

On KFDA

On one example, we saw earlier that a small value of the regularisation parameter $\gamma$ generally yielded sharper peaks that could be distinguished more easily from the non-peaks, giving a more “ideal” detection function, result that was confirmed by its best behaviour among the kernel based online onset detectors.

Now let us see the effects of the parameters on the KFDA-based onset detector. We did the same experiment as with MMD. Surprisingly, contrary to what happened with MMD, a smaller window resulted in better overall performance. Concerning the features, using X or D gave the almost the same best results. Let us examine the impact of the regularisation parameter $\gamma$. In the figure 5.7, we only plotted the precision–recall for the KFDA detector on pitched onsets, with $\sigma = 0.9$, $w = 10$ with either the cepstrum features or their delta alone (there is unfortunately no graphical distinction between the two). We changed the value of $\gamma$ from 2 to $10^{-7}$.

Figure 5.7: Influence of $\gamma$
The F-measures of detectors with large values of $\gamma$ are worse than with small ones. The results gets better as $\gamma$ decreases until a certain point: in the case of the detector with only the cepstrum features, the best result is gotten with $\gamma = 5 \cdot 10^{-5}$, while with the deltas, the F-measure continues to increase as $\gamma$ also decreases.

This fact is coherent with the theoretical and experimental results from [HBM08]: as the regularisation factor $\gamma$ decreases, the power of the two-sample problem test increases until an optimal point and then slowly decreases.

5.5 Discussion

It is unfortunate that our approach based on time descriptors and kernel dissimilarity measures does not have an overall performance matching the state-of-the art. One possible explanation is that how an onset is modelled in this approach is not really conforming to the reality. In our model, we assume that before and after a real onset, the distribution of points in the space is relatively stationary. Where this is the case, like in the electro tunes or the vocal parts, we obtained satisfactory results, performing better than the SEF in the case of the opera or having a well-shaped detection function in the case of the electro samples (even if, as we said, the thresholding window did not allow a good numerical result). But this model is failing in certain onset types, where we may for example effectively see a sudden increase in a given feature value, but then a rapid decay to the original value. Indeed our detection function may not see itself increased enough to be above the threshold to yield a detection, whereas a difference based method would be able to detect that type of onset.

This confirms that the music onset detection task is very difficult, mainly because it faces a lot of different situations with onsets with very different characteristics, making it harder to design a general algorithm that would give very good results no matter which signal it has to process.

It can be interesting to apply our framework to audio segmentation with detection of change between longer sequences, for example with locutor detection in speech conversations or scene change detection. The time constant would be closer to the second than to the dozen of milliseconds as in onset detection, and the stationary model may fit better in this case.

Be that as it may, our experiments confirmed the theoretical results of the good performance of the Kernel Fisher Discriminant, compared to the Maximum Mean Discrepancy and to the one-class SVM, while the computational complexity remains modest: the online approach may allow a real-time implementation as the complexity is linear with respect to the signal length.
Chapter 6

Conclusion

This thesis was devoted to the evaluation of kernel methods for change detection problems, particularly on music note onset detection. Methods for the two sample problem, that is comparing the similarity of two sets of vectors by trying to appreciate whether they are drawn from the same distribution, were used in a sliding time window to construct a detection function that would enable us to find the onset times in a music piece. This approach has the advantage to be model-free, as we do not need to find an accurate model to describe our signal. However some tuning is still required; the effort was particularly focused on the tuning of the meta-parameters of our kernel methods. And moreover, a post-processing step on the detection function is required to find the final onset times; this step is very critical for the performance of the framework.

Although we did not achieve outstanding results on the onset detection problem itself – which is by the way a difficult problem due to the variety of signals – as we do not match the performance of the state-of-the-art algorithms, our study still provided some useful information on kernel methods, which maybe re-used on other tasks for which our framework may be more appropriate.

A possible perspective would be to evaluate a kernel-based batch multiple change point estimation that rely on a dynamic programming approach on the whole signal to find the possible onsets. Some preliminary and non optimised work was completed, but the main bottleneck was the multiple loops in the algorithms, disallowing us to test it on some real world signal: the complexity of the algorithm is polynomial on the length of the signal, while the online based approach was linear.
Bibliography


