NTNU

Norwegian University of
Science and Technology

# Audiovisual Contents Segmentation

**Kristoffer Johan Sundøy**

Master of Science in Communication Technology

Submission date: August 2010

Supervisor: Magne Hallstein Johnsen, IET

Co-supervisor: Torbjørn Svendsen, IET

Jean-Luc Dugelay, Institute EURECOM, Sophia-Antipolis

Slim Essid, Audio, Acoustics and Waves Group, TELECOM ParisTech

Norwegian University of Science and Technology

Department of Electronics and Telecommunications

# Problem Description

Applying machine learning techniques (such as Support Vector Machine) to automatically segment the audiovisual documents (such as TV shows) into broad semantic classes such as music, speech, laughs or applause.

Assignment given: 01. March 2010
Supervisor: Magne Hallstein Johnsen, IET

TELECOM ParisTech
Département Traitement du Signal et des Images
Groupe Audio, Acoustique et Ondes

Institute EURECOM
Département des Communications Multimédia
Groupe traitement de la parole et d'audio

Norwegian University of Science and Technology, NTNU
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Electronics and Telecommunications

Master's thesis

# Audiovisual Content Segmentation

Kristoffer Sundøy

August 29th, 2010

Supervisors:

| | |
|---|---|
| Slim Essid | TELECOM ParisTech |
| Felicien Vallet | TELECOM ParisTech |
| Nickolas Evans | Institute EURECOM |
| Magne Hallstein | NTNU |
| Torbjørn Svendsen | NTNU |



The objective of this thesis is to detect high level semantic ideas to help to impose a structure on television talk shows. Indexing TV-shows is a subject that, to our knowledge, is rarely talked about in the scientific community.

There is no common understanding of what this imposed structure should look like. We can say that the purpose is to organise the audiovisual content into sections that convey a specific information. It thus encompasses issues as diverse as scene segmentation, speech noise detection, speaker identification, etc. The basic problem of structuring is the gap between the information extracted from visual data flow and human interpretation made by the user of these data. Numerous studies have examined the organisation of highly structured video content. Thus, the state of the art has many studies on sport or newscast transmissions.

Our goal is to detect key audiovisual events using a variety of descriptors and generic classifiers. We propose a generic approach that is able to assess all TV-show indexing problems. This enables an operator to use one single tool to infer a logical structure. Our approach can be considered as "semi-automatic" in the sense that the training data is collected on the fly by the operator who is asked to arbitrarily select one video excerpt of each concept involved. We have assessed a wide selection of audio and video features, used MKL as a feature selection algorithm and then built various content detectors and segmentors useful for imposing broad semantic classes on television data.

This master's thesis was set forth by TELECOM ParisTech and was begun there March 1, 2010. This final report was submitted to TELECOM ParisTech, NTNU and Institute EURECOM August 29, 2010.

L'objectif de cette thèse est de détecter les conceptes de haut niveau sémantique pour aider à imposer une structure sur les émissions de télévision. La structuration automatique des émissions de télévision est un sujet qui, à notre connaissance, est encore peu abordé par la communauté scientifique.

Il n'existe pas de définition consensuelle de la structuration. On peut dire que le but de celle-ci est d'organiser le contenu audiovisuel en sections véhiculant une information propre. Elle regroupe donc des problèmes aussi disparates que la détection de changement de plans de montage, la segmentation en scènes, le résumé automatique... Le problème fondamental de la structuration est l'écart observé entre l'information extraite des données du flux audiovisuel et l'interprétation humaine faite par l'utilisateur de ces mêmes données. De nombreuses recherches se sont penchées sur l'organisation de documents vidéos aux contenus très structurés et/ou reproductibles.

Notre objectif est de détecter les principaux événements de l'audiovisuel en utilisant une variété de descripteurs et de classifiers génériques. Nous proposons une approche générique capable d'évaluer tous les problèmes de structuration automatique des émissions de télévision. Cela permet à un opérateur d'utiliser un seul outil pour en déduire une structure. Notre approche peut être considérée comme "semi-supervisée" dans le sens que les données d'apprentisage sont collectées à la volée par l'opérateur qui est invité à choisir arbitrairement un extrait vidéo de chaque concept en cause. Nous avons utilisé des machines à vecteurs de support à noyaux multiples (MKL) comme algorithme de sélection de descripteurs et pour construire des classifieurs sur un grand ensemble de descripteurs audios et vidéso.

Cette thèse, énoncée par TELECOM ParisTech, a été commencée le 1er mars 2010. Ce rapport final a été soumis pour TELECOM ParisTech, NTNU et l'Institut EURECOM, le 29 août 2010.

# Contents

# 1. Introduction

With the introduction of the internet and advanced coding methods, the amount of digital data is, and has been for some time, increasing fast. Almost everyone is able to provide content, accessible by large portions of the population with limited central control. The availability, however, is dependent on efficient indexing of the data. Users require automatic methods to filter, process and store incoming data. For instance, as far as professionally produced content is concerned, it would be meaningful to infer a structure like a DVD chapter menu. Some of these functions will be aided by attached meta-data, which provide information about the content. However, as meta-data are not always provided, and because local processing power has increased tremendously, interest in local automatic multimedia analysis has increased. One approach is to use pattern recognition to automatically segment and structure.

Pattern recognition is the scientific discipline whose goal is the classification of objects into a number of categories or classes. Depending on the application, these objects can be images, signal waveforms or any other type of measurement that needs to be classified. Pattern recognition has a long history, but before the 1960s it was mostly the output of theoretical research in the area of statistics [Theodoridis 99]. As our society evolves from the industrial to the postindustrial phase, automation in industrial production becomes increasingly important. This trend has pushed pattern recognition to the high edge of today's engineering applications and research. Example areas where pattern recognition plays an important role today are machine vision systems, character recognition, computer-aided diagnostics and speech recognition. Very recent gaming platforms such as Microsoft's Xbox Kinect and Sony Play-station Move use real-time content based video information retrieval systems.

## 1.1. Motivation

The work is linked to Félicien Vallet's PhD thesis work on television content structuring, a subject yet rarely talked about by the scientific community. The basic problem of structuring is the semantic gap between the information extracted from visual data flow and human interpretation made by the user of these data. Numerous studies have examined the organisation of reproducible highly structured video content. Thus, the current research has many studies on sports or newscasts transmissions. The objective of Félicien Vallet's thesis is to propose an automatic approach to infer a hierarchical structure in the semantic level of television talk shows and is being carried out in cooperation with Institut national de l'audiovisuel (Ina). Ina is a French semipublic agency that conserve and do research to archive, digitise and safeguarded audiovisual assets.

There is no common understanding of what the imposed structure should look like and thus this encompasses issues as diverse as scene segmentation, speech noise detection, speaker identification, etc.

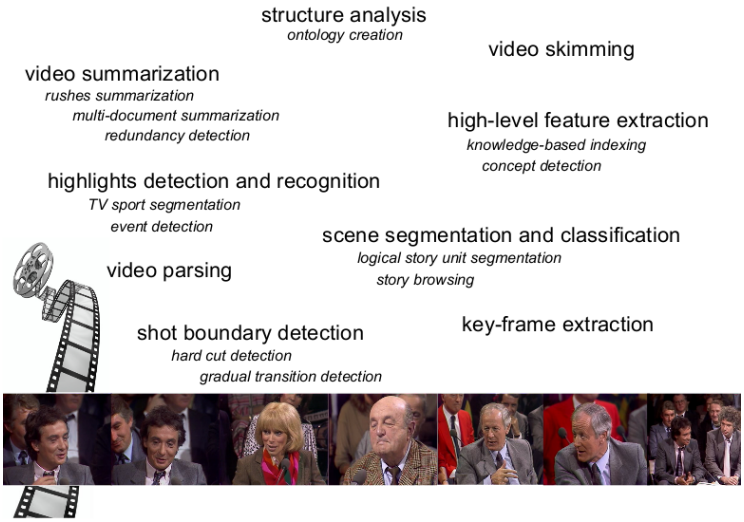
We propose a generic approach that is able to assess all these problems. This enables an operator, for instance an Ina archivist handling a given talk-show, to use one single tool to infer this structure. Our approach can be considered as "semi-supervised" in the sense that the training data is collected on the fly by the operator who is asked to arbitrarily select one video excerpt of each concept involved.

To structure a TV show, we need to detect high level semantic ideas. To do this we are dependent on all information flows available and to be able to integrate the information, known as multi-modal fusion. The database provided by Ina, see section 4.1.1), has several videos with textual annotations. These annotations, detailing the conduct of the program and the identity of persons involved in them, are a valuable source of knowledge for the approach of structuring that we envisage. Despite the restriction to one database, we aim at a generic approach, replicable and exportable to other collections of such TV shows.

## 1.2. Current research

There exists a variety of simple to complex semantic interpretation techniques and thus a taxonomy is given in figure 1.1. The figure show a increasingly higher level tasks. Given the diversity of approaches proposed for each sub-problem of structuring, a hierarchical classification of TV-shows is much more challenging than it may appear.

Figure 1.1.: Video structuring taxonomy



### 1.2.1. Fundamental entity

Shot boundary detection is a step almost inevitable for every structuring problem. As we have explained earlier, the shot is often regarded as the fundamental entity of audiovisual documents. The leading forum for video shot detection benchmarking has since 2001 been [Trecvid 10]. Every year, researchers from all over the world send their algorithms to be evaluated for hard cut and gradual transition detection. The evolution of results in [Smeaton 09] indicates, however, that this problem is now about to be abandoned, as the performance of existing systems is found to be satisfactory for current applications.

There are, nevertheless, studies that acknowledge that there is a gap between shots and the smallest perceptual segment of a television show. [Yeung 96], [Hanjalic 99a] and [Zhou 02] proposes to concatenate shots that belong to the same scene and hereby get closer to what is the smallest perceptual segment. This approach still limits the smallest blocks to start and end with the shots, but increases the freedom to add several shots in one basic segment. It is shown that a good shot boundary detector is important for a satisfactory outcome.

As the information contained in a shot often is very redundant, it is often unnecessary (and cumbersome) to use all of it. In practice, we often extract a characteristic frame or the key-frame for a given shot to represent the group of frames that makes the shot. These key-frames may serve as a resume of the television show or structure. A variety of methods have been proposed for selecting the key-frames. The most basic approaches simply select the first, the last or the middle frame of each shot. In more advanced studies, such as [Hanjalic 99b], clustering of frames is done from all the video. Their representatives are then chosen as the frames that minimise the Euclidean distance to the ensemble of each cluster. All the key-frames together provide a summary of the video. Finally, [Liu 03] uses motion vectors for selecting key-frames by creating a model of motion energy and by selecting the frame of maximum intensity. The summaries videos and products are subject to a panel of testers who provide their satisfaction of the method.

Some approaches to video summary detect the redundancy of the video information. They can also be seen as low-level semantics structuring depending on the parameterisation of the redundancy detection. Using this approach, the summary is generated by eliminating repetitions. [Dumont 08] applies a sequence alignment method to identify redundant parts of video for the task summary of TRECVid activity. A similar method, but applied to multi-document summary video, is given in [Wang 09]. This study measure the similarities between and within the video. Both tasks compute similarity scores on key-frames extracted [Wang 09] or created [Dumont 08] every second.

The structuring of a video stream can also be made without a human inference. This type of structuring is applied to perform macro segmentation and automatic indexing of large amounts of television data. [Naturel 05] present several methods of silence detection and monochrome images to find structural breaks (e.g. a shift from advertising to a program). An automated labelling of sections and classification adjustments are then made for the Electronic Program Guide (EPG).

### 1.2.2. Concept detection

The imposed audiovisual structure can be further enhanced e.g. by supervised approaches. Thus, some segmentation and classification systems are trained to detect concepts defined by the user.

Studies on sports videos are generally intended to recognise specific events. [Assfalg 02], [Baillie 04] and [Xiong 03] recognise the highlights of a football game (goals, penalties, etc.) using Hidden Markov Models (HMMs). The first method is based on the video features only. The camera movements are the observations of the Markov model, while the movement of the ball are the hidden states. Each event therefore has a specific signature characterised by an ensemble of visual features. The second method [Baillie 04] is based on audio only and classifies non-speech and crowd, high pitched sounds, crowd cheering, etc. In the last approach [Xiong 03] both audio and video are considered separately before they are combined. Other approaches like [Chen 03] use heuristic rules for detection purposes. These rules require, for example that the candidate video sequences (or the key-frames) have more than 40 percent grass or that a person occupies a significant part of the image.

An example of a "hot" research area is speaker diarisation [Tranter 06]. Speaker diarisation is the process of determining "who spoke when" in an audiovisual document and the state of the art systems use an approach based on statistical models such as Hidden Markov Models or Gaussian Mixture Models. Most systems use Mel Frequency Cepstral Coefficients (MFCC's) only [Fredouille 09]. Nevertheless, there has been, very recently, attempts by [Vajaria 06], [Bozonnet 10] and [Friedland 09] to add video features to such diarisation systems. These experiments show a small increase in performance from a state of the art audio-only diarisation system. Several other examples of broadening the feature set exist. [Leens 09] have tried to interpret the content of a video scene using a depth camera. The article demonstrate an increase in robustness even though the technology is young and provides noisy signals, small resolution and ambiguities. An article published by [Zhang 06] claims to have found that bag-of-features models have more efficient for unsupervised discovery of categories and video retrieval.

Similarly, TRECvid has proposed a high level feature extraction task where one is supposed to find predefined concepts in videos. Examples of proposed concepts are sports, desert, flag, etc. Detecting these concepts is complicated because of the variation in semantic concepts to recognise. [Benmokhtar 07] extracts audio and video descriptors to learn these concepts using support vector machine classifiers. The decision is then made by merging the outputs of these classifiers with a neural network.

[Boujema 04] and [Luo 08] approach the task by trying to extract objects of interest in the video sequences (salient objects detection). In [Boujema 04], faces are detected, extracted and compared to a database for identification. The identification is done by comparing the dynamic programming entropy map of the detected face with those of the database. This approach is resistant to deformation and change in luminance. In [Luo 08], the authors identify characteristics of objects from the medical world (faces, offices, operating rooms) to index and classify the audiovisual documents. The video is considered as a series of images, where the images are segmented into homogeneous

regions before the objects are detected by a support vector machine classification process.

### 1.2.3. Concept relations detection

Other methods of detection of semantic concepts go by the definition of an ontology. An ontology is a formal representation of the knowledge by a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to describe the domain [Noy 01]. Such an architecture can make corrections to the output of classifiers and even derive new concepts from the field study.

[Benmokhtar 09] propose the improvement of their own system [Benmokhtar 07] by incorporating an ontology to the stage of final integration of video and audio features. This allows them to calculate an inter-concept similarity from the weighted average of measures: co-occurrence of concepts, visual similarity and semantic similarity. Thus two concepts may be closely related (e.g. "car" and "road") while others may be mutually exclusive (e.g. "interior" and "exterior").

Another possibility is to use fuzzy logic as in [Athanasiadis 09] to define relationships between concepts. Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise. In contrast with crisp logic, where binary sets have binary logic, fuzzy logic variables may have a truth value that ranges between 0 and 1 and is not constrained to the two truth values of classic propositional logic. A popular example is that the deduction of the concept "Sea" for an object to which the concepts "Sky" and "Beach" have been detected. If the concepts "Sky" and "Beach" have been detected, "Sea" is always geographically located in between.

## 1.3. Our contributions

Efficient and automatic content organisation and management of digital audio and video is a key to the success of future digital libraries, and has highlighted the need for automated audiovisual content segmentation. In our work we have assessed TV-shows, an application area yet rarely talked about in the scientific community. The segmentation of TV-shows is based upon the detection of key-events, concepts and relations between these concepts.

The work is linked to Felicien Vallets joint PhD thesis between Institut national de l'audiovisuel (Ina) and TELECOM ParisTech. The two institutions are partners in many research projects such as K-Space, Infom@gic and Quaero. It is desirable to find a generic approach that can assess a wide selection of structuring problems, leaving the operators with one single tool.

We have applied multiple kernel learning [Rakotomamonjy 08] support vector machines [Cristianini 00] to a wide selection of audio and video features. Our goal is to (1) use MKL as a feature selection algorithm and then (2) build various content detectors and classifiers useful for television data structuring and indexing. Three main groups of

experiments have been assessed to test our MKL classification approach. The first, low-level experiment, was to identify speakers. The second, midlevel classification task, was to detect broad semantic classes such as applause, speech, noise, music, etc. The third, high-level classification task, was to classify the audio segments depending on whether it is recorded on stage, in a studio, outdoors, etc. Executing these tasks successively will make up a full video structuring tool.

In this chapter we have seen how this task is a part of and a building block for current and future research. This thesis has five chapters and additional appendices, where the first section is this introduction. The second chapter gives a presentation of the multiple kernel learning based classification approach, the third a thorough theoretical background for multiple kernel support vector machines, the fourth chapter has the results and discussion of the executed experiments and the fifth chapter holds the conclusions and propositions for further work. In the appendices the reader will find references, indexes, some preliminary mathematics, and information about the computing languages and environments used.

# 2. MKL for SVM classification

We all do classification every day. The term "classification" describes any situation in which some decision or forecast is made based on currently available information. The classification procedure is then a formal method for repeatedly making judgements in new situations.

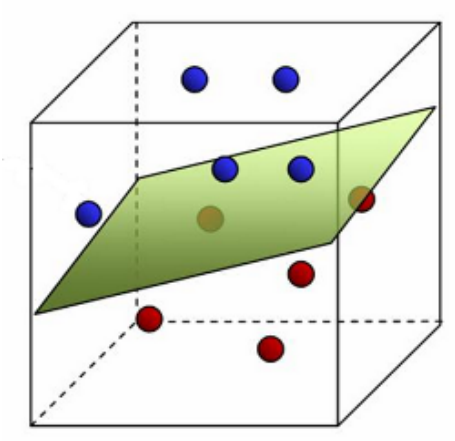


Figure 2.1.: 3-dimentional feature space

In our context, we define classification as a procedure that will be applied to a sequence of instances, in which each new instance must be assigned to one of a set of predefined classes. Classification then has two distinct meanings. We may be given a set of observations with the aim of establishing the existence of classes or clusters in the data, or we may know for certain that there are so many classes, and the aim is to establish a rule whereby we can classify a new observation into one of the existing classes. The former type is known as unsupervised learning (or clustering), the latter as supervised learning. Learning classifiers can generally be done in a supervised, semi-supervised or unsupervised fashion [Duda 73]. In supervised learning the machine is given the data with a sequence of corresponding outputs. The goal of the machine is to learn to produce the correct output given a new input. This output could be a class label (in classification) or a real number (in regression). Semi-supervised learning make use of both labelled and unlabelled data for training, typically a small amount of labelled data with a large amount of unlabelled data. In unsupervised learning the machine simply receives inputs, but obtains neither supervised target outputs, nor rewards from its environment. The goal is to build representations of the input that can be used for predicting future inputs. Figure 2 shows how a classifier is learned on training data, before it is applied to a subset

of discriminative features extracted from a validation data-set.

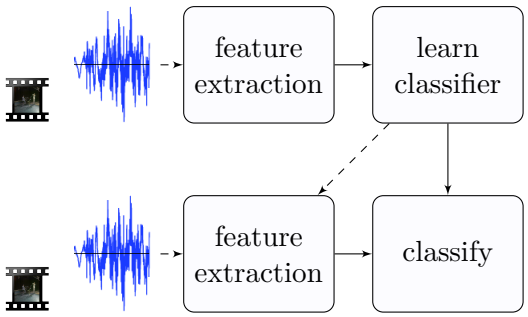



Figure 2.2.: Classification system

## 2.1. Machine learning

Historically, classification research can be divided into research on statistical classifiers, neural networks and machine learning. The goal has, nevertheless, always been (and still is) to (1) equal or exceed a human decision-maker's behavior, (2) to handle a wide variety of problems and to be generic and (3) to be used in practical settings with proven success [Michie 94].

Statistical approaches are generally characterised by having an explicit underlying probability model, which provides a probability of being in each class rather than simply a classification. In addition, it is usually assumed that the techniques will be used by statisticians, and hence some human intervention is assumed in variable selection, transformation and overall structuring of the problem. The goal of statistical approaches is to build models of known data by using one or several combinations of probability distributions such as Gaussian, as seen in figure 4.3. A mixture model is a probabilistic model for density estimation using a mixture distribution and can be regarded as a type of unsupervised clustering, used for its mathematical flexibility. For example, a mixture of two Gaussian distributions with different means may result in a distribution with a fatter tail than the basic Gaussian one, and is thus a candidate for modelling more diverse events. Popular methods consist of Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs).

Neural network approaches combine the complexity of some of the statistical techniques with the machine learning objective of imitating human intelligence: however, this is done at a more "unconscious" level and hence there is no accompanying ability to make learned concepts transparent to the user.

Machine Learning is generally taken to encompass automatic computing procedures based on logical or binary operations, that learn a task from a series of examples. Machine Learning aims to generate classifying expressions simple enough to be understood easily by the human. They must mimic human reasoning sufficiently to provide insight

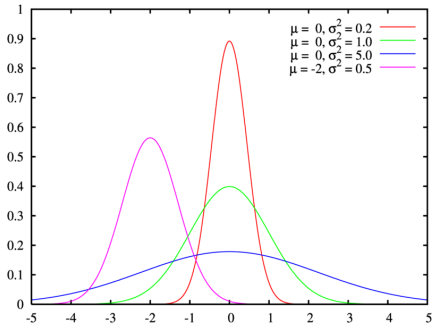



Figure 2.3.: Gaussian Mixture Model

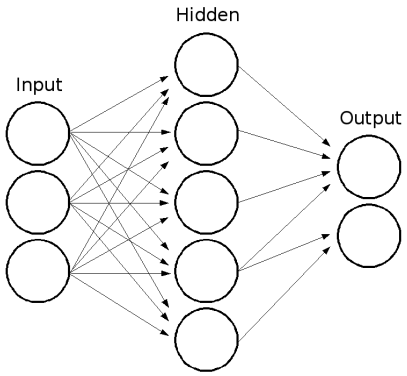



Figure 2.4.: a simple neural network

into the decision process. Like statistical approaches, background knowledge may be exploited in development, but operation is assumed without human intervention. Recently there has been a step in growth in the development of kernel-based machine learning algorithms and with frameworks such as Support Vector Machines (SVM), this have proven to be efficient training tools for solving learning problems like classification and regression. During the last decade [Vishwanathan 03], [Loosli 05], [Franc 08], [Fan 08], [Joachims 99] and [Chang 01] have all provided approaches to solve classical SVM problems.

We have chosen to apply machine learning, more specifically multiple kernel support vector machines to create our classifiers. This chapter is dedicated to the understanding of such systems. The goal of the system is (1) to select a subset of features that describes the data in a manner that separate the different classes and (2) to create a classifier that classifies unknown data. One of the advantages of multiple kernel learning support vector machines is that feature selection is embedded in the procedure. How this is done is described in detail in section 2.3.2.

### 2.1.1. Feature selection algorithms

When the input data to an algorithm is too large to be processed, it can be transformed into a reduced representation set of features (a features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data to perform the desired task using this reduced representation instead of the full size input.

Feature selection for machine learning purposes is the technique of selecting a subset of relevant features for building robust learning models. By removing most irrelevant and redundant features from the data, feature selection helps improve the performance of learning models by (1) enhancing generalisation capability, (2) speeding up learning process and (3) improving model interpretability.

Feature selection algorithms typically fall into two categories: feature ranking and subset selection [Guyon 03]. Feature ranking ranks the features by a metric and eliminates all features that do not achieve an adequate score. Subset selection searches the set of possible features for the optimal subset. There are three main approaches to feature selection, "embedded methods", "filter methods" and "wrapper methods".

The embedded methods approach select distinctive features or descriptors (group of features) in one single process by the conjoint optimisation of features and classification attributes. This is being done by assessing subsets of variables according to their usefulness to a given predictor. Embedded methods are considered very efficient [Guyon 03] because of the way they incorporate variable selection as part of the training process. This leads to better use of the available data by not needing to split the training data into a training and validation set; they reach a solution faster by avoiding retraining a predictor from scratch for every variable subset investigated.

The filter methods gives the available attributes a score based solely on the individual relevance. If one consider a set of $m$ examples $x_k, y_k$ where $k = [1, ..., m]$ consisting of $n$ input variables $x_{ki}$ where $i = [1, ..., n]$ and one output variable $y_k$ (labels), filter methods makes use of a scoring function $S(i)$ computed from the values $x_{ki}$ and $y_k$. The subset size must be chosen by the user.

An example of a filter approach is the Fisher discriminant based method. The basic idea of the Fisher discriminant based method, as given in [Cheng 92], is to calculate the optimal discriminant vector on the condition that the Fisher criterion function takes extremum. Furthermore to project higher-dimensional feature vectors on the obtained optimal discriminant vector for constructing a 1D feature space.

Wrapper methods use the learning machine of interest as a black box to score subsets of features according to their predictive power to optimise the performance of the considered application. This means that the features that has the greatest absolute distances in the feature space for data points with different labels are preferred. An example of a wrapping method is multiple kernel learning support vector machines, with embedded feature selection.

## 2.2. Creating the classifier

A classifier is a function that maps sets of input attributes to tagged classes. Both Data Mining and Machine learning are techniques related to the classifying of large amounts of data. The Data Mining approach tries to obtain patterns or models from the data collected. Machine learning builds a model that predicts whether a new example falls into one category or the other. Common classifier training approaches are Bayesian classifiers, Parzen classifiers, back-propagation classifiers, PCA classifiers and Support vector machines.

There are many important aspects to keep in mind when you are creating a classifier. The reliability of the rule is a first and logic aspect. Furthermore, for many applications, training a classifier can be complex and computationally heavy as long as the testing is fast and efficient. A classifier that is 90% accurate may be preferred over one that is

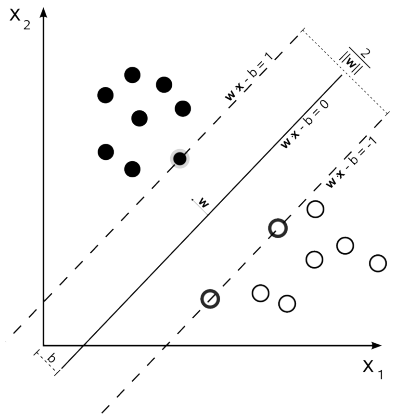



Figure 2.5.: A SVM classifier on a binary dataset

95% accurate if it is 100 times faster in testing. Such considerations would be important for the automatic reading of postal codes, or automatic fault detection of items on a production line for example. The training time may, nevertheless, be an issue, especially in a rapidly changing environment. In such cases it might be necessary to learn a classification rule quickly, or make adjustments to an existing rule. This section is dedicated to how to create an optimal classifier.

Imagine a set of training samples $x_1, ..., x_n$ which are assigned labels $y_1, ..., y_n$ ($y_i \subset -1, 1$). The basic idea of support vector machines is to search for a plane $w \cdot x + b = 0$ that separates the samples so that the distance $\frac{2}{\|w\|}$ between the hyperplane and the closest samples is maximised. Figure 2.5 has a graphical outline of such a classifier.

To this purpose we need some preliminary definitions [Pontil 98]

**Definition 2.1.** *The set $S$ is linearly separable if there exist $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that $y_i(w \cdot x_i + b) \geq 1$ for i = 1,2, ... N.*

The pair $(w, b)$ defines a hyperplane of equation

$$w^T x + b = 0 \tag{2.2.1}$$

named separating hyperplane. The signed distance $d_i$ of a point $x_i$ from the separating hyperplane $(w, b)$ is given by

$$d_i = \frac{w^T x + b}{\|w\|} \tag{2.2.2}$$

Combining definition 2.1 and inequality (2.2.2) for all $x_i \in S$ we get

$$y_i d_i \geq \frac{1}{\|w\|} \tag{2.2.3}$$

Therefore, $\frac{1}{\|w\|}$ is a lower bound on the distance between the points $x_i$ and the separating hyperplane $(w, b)$.

We then need to establish a one-to-one correspondence between separating hyperplanes and their parametric representation.

**Definition 2.2.** *Given a separating hyperplane $(w, b)$ for the linearly separable set S, the standard representation of the separating hyperplane is obtained by rescaling the pair $(w, b)$ into the pair $(\bar{w}, \bar{b})$ in such a way that the distance of the closest point, $x_j$, equals $\frac{1}{\|\bar{w}\|}$.*

Definition 2.2 leads to

$$\min_{x_i \in S} \{y_i(\bar{w} \cdot x_i + \bar{b})\} = 1 \tag{2.2.4}$$

This establishes inequality (2.2.2) as a tight inequality, meaning that no better limit value can be found. The optimal separating hyperplane can thus be defined. In the following we use notations $(w, b)$ and consider them as $(\bar{w}, \bar{b})$.

**Definition 2.3.** *Given a linearly separable set S, the optimal separating hyperplane is the separating hyperplane for which the distance of the closest point of S is maximum.*

The Vapnik-Chervonenkis (VC) dimension is a measure of the capacity of a statistical classification algorithm, defined as the cardinality of the largest set of points that the algorithm can shatter. If the VC-dimension of the family of decision surfaces is known, the theory of SVMs provides an upper bound for the probability of misclassification of the test set for any possible probability distributions of the data points.

### 2.2.1. Optimal Separating Hyperplane

Since the distance of the closest point equals $\frac{1}{\|w\|}$, the optimal separating hyperplane can be regarded as the solution of the problem of maximising $\frac{1}{\|w\|}$ subject to the constraint in definition 2.1.

$$\begin{gathered} \min \tfrac{1}{2} w^T w \\ y_i(w \cdot x_i + b) \geq 1 \quad \forall i = 1, ..., N \end{gathered} \tag{2.2.5}$$

Equation 2.2.5 is usually solved by means of the method of Lagrange multipliers. If we denote with $\alpha = \alpha_1, ..., \alpha_N$ the $N$ non-negative Lagrange multipliers associated with the constraints in definition 2.1, the solution to equation (2.2.5) is equivalent to determining the saddle point of the function

$$L = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i \{y_i(w \cdot x_i + b) - 1\} \tag{2.2.6}$$

A saddle point is a point in the domain of a function which is a stationary point but not a local extremum. The name derives from the fact that in two dimensions the surface resembles a saddle that curves up in one direction, and curves down in a different direction. At the saddle point, L has a minimum for $w = \bar{w}$ and $b = \bar{b}$ and a maximum for $a = \bar{a}$, and thus we can write

$$\frac{\delta L}{\delta b} = \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.2.7}$$

and

$$\frac{\delta L}{\delta w} = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0 \tag{2.2.8}$$

with

$$\frac{\delta L}{\delta w} = \{\frac{\delta L}{\delta w_1}, ..., \frac{\delta L}{\delta w_N}\}$$

Substituting (2.2.7) and (2.2.8) into the right-hand side of (2.2.6), we see that equation (2.2.5) reduces to the maximisation of the function [Pontil 98]

$$\Lambda = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.2.9}$$

subject to the constraint in equation (2.2.8) with $\alpha \geq 0$. This new problem is called dual problem and can be formulated as

$$\max \frac{1}{2} \alpha^T D \alpha + \sum_{i=1}^{N} \alpha_i \tag{2.2.10}$$

where

$$\textstyle\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \alpha \geq 0$$

and D is a $N * N$ matrix such that

$$D_{ij} = y_i y_j x_i \cdot x_j$$

As for the pair $(\bar{w}, \bar{b})$, from equation (2.2.8) it follows that

$$\bar{w} = \sum_{i=1}^{N} \bar{\alpha}_i y_i x_i \tag{2.2.11}$$

while $\bar{b}$ can be determined from $\bar{a}$, solution of the dual problem, and from the Kühn-Tucker conditions

$$\bar{\alpha}_i (y_i (\bar{w} \cdot x_i + \bar{b}) - 1) = 0 \tag{2.2.12}$$

Note that the only $a_i$ that can be nonzero in equation (2.2.8) are those for which the constraints in definition 2.1 are satisfied with the equality sign. This has an important consequence. Since most of the $\bar{\alpha}_i$ are usually null, the vector $\bar{w}$ is a linear combination of a relatively small percentage of the points $x_i$. These points are termed support vectors because they are the closest points from the optimal separating hyperplane and the only points of $S$ needed to determine the hyperplane (see figure 2.5). Given a support vector $x_j$, the parameter $\bar{b}$ can be obtained from the corresponding Kuhn-Tucker condition as

$$\bar{b} - y_j - \bar{w} \cdot x_j \tag{2.2.13}$$

The problem of classifying a new data point $x$ is now simply solved by looking at the sign of

$$\bar{w}^T x + \bar{b} \tag{2.2.14}$$

Therefore, the support vectors condense all the information contained in the training set S which is needed to classify new data points.

### 2.2.2. Outliers

It is only the closest points of the linearly separable set S that are needed to determine the optimal separating hyperplane. Nevertheless, to avoid over-fitting training data, some datapoints may be ignored or allowed on the wrong side of the classifier. Over-fitting occurs when a classifier, or model, is so well adapted to the training data that it loses it general aspect. This can be seen in figure 2.6. The opposite of over-fitting is under-fitting, seen in figure 2.7. This happens when one does not train the classifier, or model, sufficiently such that it does not adjust properly to the data set.

Ignored data-points are often referred to as outliers to the classifier. To account for

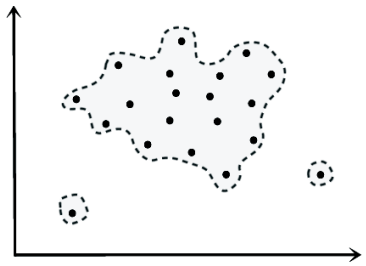


Figure 2.6.: over-fitted model

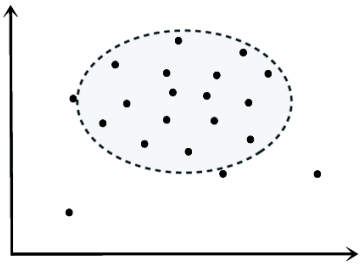


Figure 2.7.: under-fitted model

outliers, slack variables $\xi_i$ are introduced in the Kuhn-Tucker conditions. This changes equation (2.2.12) to: $y_i(x_i w + b + \xi_i - 1) \geqslant 0$. When solving this optimisation problem the sum of the $\xi_i$s is penalised in the objective function with a weight factor $C$ (to be chosen) to control the total number of outliers. In practice, the minimisation problem in equation (2.2.5) changes to:

$$\min_{wb} \|w\|^2 + C \sum_i \xi_i \tag{2.2.15}$$

Note that this term leads to a more robust solution, in the statistical sense, than the intuitively more appealing term $C \sum_i \xi_i^2$. In other words, the term $C \sum_i \xi_i$ makes the optimal separating hyperplane less sensitive to the presence of outliers in the training set. The parameter $C$ can be regarded as a regularisation parameter. The optimal separating hyperplane tends to maximise the minimum distance $\frac{1}{\|w\|}$ for small $C$, and minimise the number of misclassified points for large $C$. For intermediate values of C the solution of equation (2.2.15) trades errors for a larger margin.

As we have adapted equation (2.2.5) to equation (2.2.15), we need to rewrite the dual problem as well. Derived in the same manner as before, it can be written:

$$\max \frac{1}{2} \alpha^T D \alpha + \sum_{i=1}^{N} \alpha_i \tag{2.2.16}$$

where

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad 0 \leq \alpha_i \leq C$$

As for the pair $(\bar{w}, \bar{b})$, one can derive

$$\bar{w} = \sum_{i=1}^{N} \bar{\alpha_i} y_i x_i \tag{2.2.17}$$

while $\bar{b}$ still can be determined from $\bar{a}$, solution of the dual problem in equation (2.2.16), and from the new Kuhn-Tucker conditions

$$\bar{\alpha_i}(y_i(\bar{w} * x_i + \bar{b}) - 1 + \bar{\xi_i}) = 0 \tag{2.2.18}$$

$$(C - \bar{\alpha_i})\bar{\xi_i} = 0 \tag{2.2.19}$$

where the $\bar{\xi_i}$ are the values of the $\xi_i$ at the saddle point. Similar to the case without outliers, the points $x_i$ for which $\bar{\alpha_i} > 0$ are termed support vectors. The main difference is that here we have to distinguish between the support vectors for which $\bar{\alpha_i} > C$ and those for which $\bar{\alpha_i} = C$. In the first case, from equation 2.2.19 it follows that $\bar{\xi_i} = 0$, and hence, from equation (2.2.18), that the support vectors lie at a distance $\frac{1}{\|w\|}$ from the optimal separating hyperplane. These support vectors are termed margin vectors. The support vectors for which $\bar{\alpha_i} = C$, are misclassified points (if $\xi_i > 1$), points correctly classified but closer than $\frac{1}{\|w\|}$ from the optimal separating hyperplane (if $0 < \xi \leq 1$), or margin vectors (if $\xi_i = 0$). Neglecting this last rare occurrence, we refer to all the support vectors for which $\alpha_i = C$ as errors. All the points that are not support vectors are correctly classified and lie outside the margin strip.

### 2.2.3. The Kernel

As shown in section 2.2.1, data is not always linearly separable in their input space. A kernel function $k(x, y)$ can be used to project data in the input space into a higher dimension feature space in which linear analysis exhibits nonlinear behaviour in the input space. This projection is performed using a nonlinear mapping function. Consequently, many of the existing learning algorithms that depend only on inner products may be rewritten using kernels. This is called "the kernel trick".

Support Vector Machines, as described in [Cristianini 00], is a kernel based method to train a linear classifier between binary data-sets. Formally, a kernel function k is defined as [Abbasnejad 10]:

$$\begin{gathered} k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \\ k : X * X \to \mathbb{R} \end{gathered} \tag{2.2.20}$$

where $\langle ., . \rangle$ denotes the inner product operation. Mapping function $\phi : \mathbb{R}^d \to \mathcal{H}$ maps data points $x_i$ and $x_j$ to Hilbert space $\mathcal{H}$ in a nonlinear manner. The kernel function $k$ can be used in the construction of a $n \cdot n$ matrix:

$$K = (k(x_i, x_j))_{ij} \tag{2.2.21}$$

called the Kernel matrix or Gram matrix of k with respect to $x_1, x_2, ..., x_n \in \mathbb{R}^d$. The gram matrix is always symmetric and positive semi-definite, but can grow very large as the number of kernels and samples increase.

In a kernel function k, the mapping function $\phi$ creates the feature space in which the relation between data points is defined as a linear combination of their features. Consequently, it is implied that the kernel function defines the local relation between a pair of points regardless of the others in the data-set under study. An inner product also corresponds to the cosine between two input vectors. The angle between two vectors is used as the reference for their similarity: a larger angle corresponds to smaller similarity value. A good feature space defined by k is the one that best determines the similarity of points for a specific data-set.

According to Mercer's Theorem, any matrix may be interpreted as the inner product in the feature space as long as it is positive-definite, or conversely any positive-definite matrix may be used as a kernel matrix. Following the introduction of this theorem, a large number of kernels have been formulated. Among them are variations of parametric and non-parametric kernels.

A test vector $x$ is then classified with respect to the sign of the equation (2.2.22).

$$f(x) = \sum_{i=1}^{n_s} \alpha_i y_i k(s_i, x) + b \tag{2.2.22}$$

where $s_i$ are the support vectors, $\alpha_i$ are Lagrange multipliers, and $n_s$ is the number of support vectors. Researchers such as [Barrington 08], [Xu 08], [Zhang 06], [Rubner 00] and [Zamolotskikh 07] discuss which kernels that apply best to which type of features. Some common kernels are:

| | |
|---|---|
| Gaussian | $k(x, y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$ |
| Polynomial | $k(x, y) = (x \cdot y)^{\delta}$ |
| Linear | $k(x, y) = x \cdot y$ |

Moreover, choosing a good kernel is a nontrivial model selection task from a statistical point of view. The problem of learning has been traditionally solved using cross-validation [Abbasnejad 10]. Cross-validation generally consists of testing a supervised algorithm with a range of possible kernels and hyper-parameters on a subset of training examples. Subsequently, the kernel and its corresponding hyper-parameters that produced the best average results are selected. Several runs of training and testing cycles are required to obtain a good result, which may not be viable for large data-sets. The cross-validation approach, however, is time consuming and is hence often limited to just a few kernels and hyper-parameters, and most notably it is not applicable to unsupervised algorithms.

### 2.2.4. Imbalanced data sets

If one consider an imbalanced bi-class data set, the soft margin will have a biased impact on the classification. A proposed solution is to use a different $C$ factor for positive and negative training examples so that the solution is not biased by the over-representation of one class at the expense of another, resulting in equation (2.2.15) looking like equation (2.2.23).

$$\min_{wb} \|w\|^2 + C^+ \sum_i \xi_i + C^- \sum_i \xi_i \tag{2.2.23}$$

The testing classifies the data according to the classifier (hyperplane) created in the previous step. Finally, the amount of mistakes that was done by the testing is counted and given as the rank of that specific classifier.

### 2.2.5. Multi-class classification

The need for classifiers that classify the data into more than two data-sets is obvious. There are several ways to classify into more than two classes with SVM, but he dominating approach for doing so is to reduce the single multi-class problem into multiple binary problems. Each of the problems yields a binary classifier, which is assumed to produce an output function that gives relatively large values for examples from the positive class and relatively small values for examples belonging to the negative class.

When applying classifiers to supervised problems, every assessment has to be considered as a bi-class assessment. How to effectively extend it for multi-class classification is still an ongoing research issue. Several methods have been proposed where one typically construct a multi-class classifier by combining several binary classifiers. Direct multi-class optimisation is to complex to solve in reasonable time. In the first case every label is tested against every other label before the classifiers are combined. In the second case, every label is tested against all the other labels before the classifiers are combined. [Hsu 02] assess the three main approaches to be able to find more classes: one-versus-one, one-versus-all, directed acyclic graph SVM (DAGSVM). The DAGSVM training phase is the same as the one-versus-one approach by solving binary SVMs. However, in the testing phase, it uses a rooted binary directed acyclic graph which has internal nodes and leaves. [Hsu 02] states that the one-versus-one and DAGSVM approaches gives better results and less computation time than the one-versus-all.

Referring to [Hsu 02] the 'one versus one" approach was chosen in this project. "One versus one" constructs $\frac{k(k-1)}{2}$ classifiers where each one is trained on data from two classes. For training data from the i-th and the j-th classes, we solve the following binary classification problem:

$$\begin{aligned} &\min_{w^{i,j}b^{i,j}\xi^{i,j}} \tfrac{1}{2}(w^{i,j})^T w^{i,j} + C\sum_t \xi_t^{i,j}(w^{i,j})^T \\ &(w^{i,j})^T\phi(x_t) + b^{i,j} \geq 1 - \xi_t^{i,j} \forall y_t = i \\ &(w^{i,j})^T\phi(x_t) + b^{i,j} \leq -1 + \xi_t^{i,j} \forall y_t = j \\ &\xi_t^{i,j} \geq 0 \end{aligned} \tag{2.2.24}$$

There are different methods for doing the future testing after all classifiers are constructed. A simple strategy, the "MaxWins" voting strategy, is suggested in [Hsu 02]: if $sign(w^{i,j})^T\phi(x_t) + b^{i,j}$ says $x$ is in the i-th class, then the vote for the i-th class is added by one. Otherwise, the j-th is increased by one. Then we predict $x$ is in the class with the largest vote.

## 2.3. Multiple Kernel Learning

While classical kernel-based classifiers are based on a single kernel, in practice it is often desirable to base classifiers on combinations of multiple kernels. This section is dedicated to the how to create an optimal set of kernels.

Binary classification with multiple kernel learning (MKL) was introduced by [Lanckriet 04]. MKL allows an automated selection of kernels from a family of potential candidates. This is a constrained problem that rapidly becomes intractable as number of learning examples or kernels becomes large, see theory section 2.2.3. To reduce the time and space complexities, a popular technique is to obtain low-rank approximations on the kernel matrix, by using the Nystrom method [Williams 01], greedy approximation [Smola 04], sampling [Achlioptas 02] or matrix decompositions [Fine 01].

Researchers such as [Bach 04], and [Zien 07] have proposed algorithms that are able to handle medium scale and multi-class problems. [S.Sonnenburg 06] have proposed a large scale multiple kernel learning toolbox that implements several of the state of the art SVM algorithms, while [Rakotomamonjy 08] proposes a new MKL algorithm that handles large scale problems.

[Kloft 10] divides recent MKL approaches into three categories. The *first* approach leaves the kernel as a variable and demands that the kernel is taken from a linear span of base kernels $k := \sum_{i=1}^{M} \theta_i k_i$. The learning procedure then optimises over the $\theta$ as well as the kernel classifier parameters. This approach can for instance be found in [Kloft 09]. The *second* approach optimises over all the kernel classifiers for each of the base kernels, but modifies the weighting to a block norm (a norm of the vector containing the individual kernel norms). This approach can for instance be found in [Aflalo 10]. The *third* approach has only the best kernels contributing (sparse kernel weights) to the final classifier by extending the second approach with an elastic net weighting. This approach was recently described in [Tomioka 10].

### 2.3.1. The set of kernels

Selecting the kernel function and its parameters (e.g. degree in the polynomial kernel or spread in the Gaussian kernel) is an important issue in training. Generally, a cross-validation procedure is used to choose the best performing kernel function among a set of kernel functions on a separate validation set different from the training set.

Multiple Kernel Learning (MKL) has emerged as a useful tool when the specific kernel is unknown, or when there are various sources of information to combine. MKL searches over a convex combination of kernels for the one most suited to the task. When considering more than one kernel, one looks for a decision function of the form $f(x) + b = \sum_m f_m(x) + b$ where each function $f_m$ belongs to a different Reproducing Kernel Hilbert Space associated with a kernel $k_m$. The linear combination of kernels can be expressed as [Dileep 09]:

$$f(x) = \sum_i y_i d_m k(x, x_i) + b \tag{2.3.1}$$

The reasoning is similar to combining different classifiers: Instead of choosing a single kernel function and putting all our eggs in the same basket, it is better to have a set and let an algorithm do the picking or combination. In equation (2.3.1) each kernel has a weighting factor $d$, that if summed over all the $m$ kernels sums up to 1.

$$\sum_m d_m = 1 \tag{2.3.2}$$

The combination of kernels $K(x, x_i)$ is given by equation (2.3.8). Such classifiers can be constructed in a variety of ways. [Lanckriet 04] proposed the problem as a quadratically constrained quadratic program. [Bach 04] showed that the problem formulated by [Lanckriet 04] is equivalent to a mixed (l2, l1) norm regularisation on the weight vector and proposed an SMO-like algorithm to solve it. [S.Sonnenburg 06] formulated the problem as a semi-infinite linear program (SILP). Their algorithm is an instance of a class of exchange methods for solving SILPs. [Rakotomamonjy 08] used l2 norm regularisation on the weight vector and proposed an algorithm based on gradient descent. This solution is the one applied in our system, and hence presented in greater detail.

Adding equation (2.3.1) to the equations in section 2.2.1, slightly changes the primal and dual problem. [Rakotomamonjy 08] proves that the primal problem can be addressed by solving the following convex problem.

$$\min_{f_m,b,\xi,d} \frac{1}{2}\|f\|_H^2 + C\sum_i \xi_i \tag{2.3.3}$$

where

$$\begin{cases} y_i \sum_m \frac{1}{d_m} f_m(x_i) + y_i b > 1 - \xi_i \forall i \\ \xi_i > 0 \forall i \\ \sum_m d_m = 1, d_m > 0 \forall m \end{cases}$$

This is referred to as the primal problem. [Rakotomamonjy 08] simplifies this problem by constraining it to the following expression.

$$\begin{gathered} \min_{f_m,b,\xi,d} \frac{1}{2}\sum_m \frac{1}{d_m}\|f\|^2 + C\sum_i \xi_i \\ y_i \sum_m f_m(x_i) + y_i b > 1 - \xi_i \forall i \\ \xi > 0 \forall i \\ \sum_m d_m = 1, d_m > 0 \forall m \end{gathered} \tag{2.3.4}$$

The Lagrangian of this problem can be expressed as

$$L = \frac{1}{2}\sum_m \frac{1}{d_m}\|f_m\|_H^2 + C\sum_i \xi_i + \sum_i \alpha_i(1 - \xi_i - y_i \sum_m f_m(x_i) - y_i b) - \sum_i v_i \xi_i \tag{2.3.5}$$

If one calculates the derivatives and sets them to zero, the following conditions appear.

$$\begin{aligned} \tfrac{1}{d_m} f_m &= \textstyle\sum_i \alpha_i y_i k_m(x_i) \forall m \\ \textstyle\sum_i \alpha_i y_i &= 0 \\ C - \alpha_i - v_i &= 0 \forall i \end{aligned} \tag{2.3.6}$$

From this the associated dual problem can be derived.

$$\begin{aligned} max_\alpha -\tfrac{1}{2} \textstyle\sum_{i,j} \alpha_i \alpha_j y_i y_j \sum_m d_m k_m(x_i, x_j) + \sum_i \alpha_i \\ \textstyle\sum_i \alpha_i y_i = 0 \forall i \\ C > \alpha_i > 0 \forall i \end{aligned} \tag{2.3.7}$$

SimpleMKL, a steepest descent algorithm described in [Dileep 09], gives a higher $\gamma$-value for kernels that leads to a better classification. $\gamma = 0$ infers that one can disregard the kernel when classifying the specific set of features. SimpleMKL does thus, ensure feature selection $d \ll D$ and a classification for a given set of features $D$.

$$K(x, x_i) = \sum_{m=1}^{M} d_m K_m(x, x') \tag{2.3.8}$$

Every kernel $K_m(x, x_i)$ in equation (2.3.8) has a complexity parameter $d$, where $d_m > 0$ and $\sum_{m=1}^{M} d_m = 1$. A small $d$ gives a complex solution while a big $d$ gives a less complex solution, tending to be a plane.

There can be two uses of MKL: Different kernels correspond to different notions of similarity and instead of trying to find which works best, a learning method does the picking for us, or may use a combination of them. Using a specific kernel may be a source of bias, and in allowing a learner to choose among a set of kernels, a better solution can be found. Furthermore, different kernels may be using inputs coming from different representations possibly from different sources or modalities. Since these are different representations, they have different measures of similarity corresponding to different kernels. In such a case, combining kernels is one possible way to combine multiple information sources. There is significant amount of work in the machine learning literature for combining multiple kernels.

We do know that some kernels apply better to some descriptors than others. A natural optimisation would hence be to set given kernels to given descriptors (e.g. a probability product kernel or $x^2$-kernels to colour histograms and a Gaussian kernel to cepstral coefficients) before applying the steepest descent algorithm.

Many kernel methods are formulated as quadratic programming ($QP$) problems. If we denote the number of training patterns by $m$, the training time complexity of $QP$ is then $O(m^3)$ and its space complexity is at least quadratic. Hence, a major stumbling block is in scaling up these QP's to large data sets, such as those commonly encountered in data mining applications. We do, however, never optimise over the whole set, but reduce the complexity by letting MKL select a subset of training patterns.

### 2.3.2. Embedded feature selection

The MKL approach presented in section (2.3.1) manipulate entire kernels. In contrast, one might write each kernel as the sum of rank-one kernels,

$$K_i = \sum_{j=1}^{N} M_{ij} \tag{2.3.9}$$

before applying the approach by [Rakotomamonjy 08] to (a subset of) the collection of rank-one kernels, $M = \{M_{ij}\}$. This is a form of feature extraction. If most of the discriminating information contained in the $K_i$ can be represented by a small number of $M_{ij}$, we anticipate that this procedure will produce better classifiers than procedures based on convex combinations of the $K_i$ [Tuia 09]. In this approach we build as many kernels as we have initial variables. Thus, in the case of large datasets, the original formulation for SimpleMKL strategy cannot be used for feature selection [Tuia 09].

We have modified SimpleMKL to apply specific kernels to subsets of features. Looking at the weighting of the kernels after building the classifier is a form of feature selection.

## 2.4. Probability computation

After classifying the test data it is interesting to compute the probability that a point is a class given the classifier. The posterior probability can be expressed with .

$$P(y = 1|f) = \frac{p(f|y = 1)P(y = 1)}{\sum_{i=[-1,1]} p(f|y = i)P(y = i)} \tag{2.4.1}$$

However, [Platt 99] suggests to use a to fit the posterior $P(y = 1|f)$ directly, instead of estimating the class-conditional densities $p(f|y)$. The parameters of the model are adapted to give the best probability outputs. In [Platt 99] it is shown empirically how class-conditional densities between the margins are exponential. Bayes' rule, equation (2.4.1), on two exponentials suggests using a parametric form of a sigmoid.

$$P(y = 1|f) = \frac{1}{1 + exp(Af + B)} \tag{2.4.2}$$

This is equivalent to assuming that the output of the SVM is proportional to the log odds of a positive example. This sigmoid model is different from the one proposed in equation (2.4.1) because it has two parameters, A and B, trained discriminatively, rather than one parameter estimated from a tied variance. The parameters are fit using maximum likelihood estimation from a training set. In practice this is done by minimising the negative log likelihood of the training data, which is a cross-entropy error function:

$$\min_{A,B}[-\sum_{i} t_i log(p_i) + (1 - t_i) log(1 - p_i)] \tag{2.4.3}$$

where

$$p_i = \frac{1}{1 + \exp(Af_i + B)} \tag{2.4.4}$$

and $t_i$ are target probabilities from the training set defined as:

$$t_i = \frac{y_i + 1}{2} \tag{2.4.5}$$

## Summary

Multiple kernel support vector machines is a supervised way of training a classifier in a high dimensional feature space. The basic idea of support vector machines is to search for a plane $w \cdot x + b = 0$ that separates the samples so that the distance $\frac{2}{\|w\|}$ between the hyperplane and the closest samples is maximised.

To avoid over-fitting training data, some datapoints may be ignored or allowed on the wrong side of the classifier. Over-fitting occurs when a classifier, or model, is so well adapted to the training data that it loses it general aspect. To account for outliers, slack variables $\xi_i$ with a weighting factor $C$ are introduced in the Kuhn-Tucker conditions.

When applying classifiers to supervised problems, every assessment has to be considered as a bi-class assessment. The "One versus one" approach to solve multi-class problems constructs $\frac{k(k-1)}{2}$ classifiers where each one is trained on data from two classes.

As shown in section 2.2.1, data is not always linearly separable in their input space. A key elements of support vector machines success as a learning technique is the non-linear feature mapping designed by the kernel function. Mapping function $\phi : \mathbb{R}^d \to \mathcal{H}$ maps data points $x_i$ and $x_j$ to Hilbert space $\mathcal{H}$ in a non-linear manner.

Multiple Kernel Learning (MKL) searches over a convex combination of kernels for the one most suited to the task. When considering more than one kernel, one looks for a decision function of the form $f(x) + b = \sum_m f_m(x) + b$ where each function $f_m$ belongs to a different Reproducing Kernel Hilbert Space associated with a kernel $k_m$.

By applying sets of kernels to subsets of features and letting the MKL-algorithm calculate kernel weights, we use MKL to do feature selection.

# 3. Audiovisual event classification system

This chapter describes the classification system used for our audiovisual analysis tasks. We have developed a system based on the five stages shown in figure 3.1: feature extraction, preprocessing, training, testing and post-processing. The goal of the system is (1) to select a subset of features that describe the data in a manner that optimise the discrimination of the different classes and (2) to create a classifier that classifies unknown data.

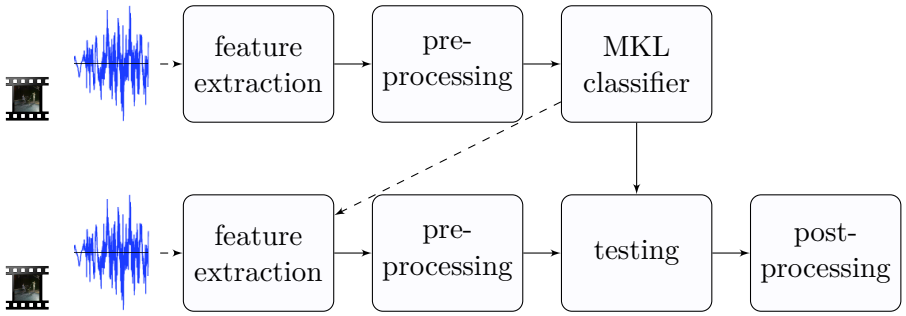



Figure 3.1.: Classification system

We start by creating two separate data-sets, one for training and one for testing. After extracting the features from the training-set, an important goal is to select the features that give the best classification. These features are pre-processed (normalised) before a classifier is trained to separate the data as good as possible. The selected features are then extracted from the test-set and pre-processed in the same way as the training data. The test data can thus be classified by the classifier created in the previous step. Finally the output data vectors are post-processed with heuristic approaches to optimise the results.

The following five sections describe these steps in further detail.

## 3.1. Feature extraction

The first step of our classification system is feature extraction. For most classification tasks there is a general acceptance to use a specific subset of features. However, several recent authors (e.g. [Leens 09], [Bozonnet 10] and [Friedland 09]) add more to their systems. [Leens 09] demonstrate an increase in robustness even though the technology is young and provides noisy signals, small resolution and ambiguities. [Bozonnet 10]

and [Friedland 09] add video features to diarisation systems. These experiments show a small increase in performance from a state of the art audio-only diarisation system.

We would like to keep a generic approach for our experiments and have hence extracted over 800 different audio and video features from the "Grand Echiquier" database. These include common features such as variations of cepstrum coefficients and image histograms, but also more "experimental" features such as acceleration of a detected face. The next two subsections will provide a deeper understanding of these features.

### 3.1.1. Audio features

To describe audio for different applications, a lot of audio features have been proposed. The most recent ones come from the speech recognition community, studies on sound classification, but also from the results of psycho-acoustical studies.

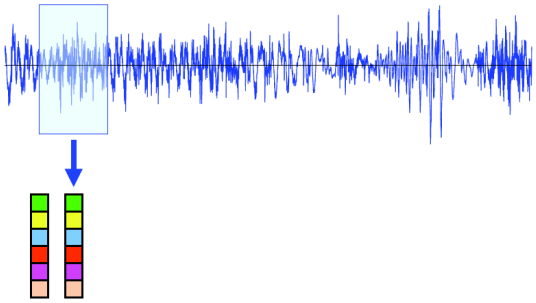


Figure 3.2.: Extraction of audio features

Audio features can broadly be divided into groups of the kind of signal representation used to extract them. *Temporal features* such as magnitudes and zero crossing rates are extracted from the time domain, *spectral features* such as spectral slope variation are extracted after spectral analysis in the frequency domain, *harmonic features* are instantaneous features such as the fundamental frequency computed from the sinusoidal harmonic modeling of the signal, while *psycho-acoustic features* such as loudness are computed with a weighting that tries to simulate human perception. The extracted audio feature sets are given in table 3.1 and described more thoroughly in appendix A.1.

Table 3.1.: Audio features

| Family | Feature | Dimension |
|---|---|---|
| Temporal | Zero Crossing Rate | 1 |
| | Moments | 4 |
| | Auto correlation | 49 |
| | Linear Predictive Coding | 2 |
| Spectral | Line Spectral Frequencies | 10 |
| | Spectrum Flatness | 19 |
| | Spectral Crest Factor | 19 |
| | Spectral decrease | 1 |
| | Spectral rolloff | 1 |
| | Spectral variation | 1 |
| | Spectral moments | 4 |
| | Spectral slope | 1 |
| | Mel-Frequency Cepstral Coefficients | 13 |
| | $\Delta$ MFCC | 13 |
| | $\Delta\Delta$ MFCC | 13 |
| | bandwidth | 1 |
| Harmonic | Fundamental frequency | 1 |
| | entropy | 1 |
| Psycho-acoustic | Loudness | 24 |
| | Loudness sharpness | 1 |
| | Loudness spread | 1 |

### 3.1.2. Video features

We deal with a particular type of content, TV talk-shows. Contrary to databases used in biometric and speaker diarisation works, the video content is edited i.e. many cameras are used during the shooting. The shot is chosen by the TV director (typically through a video switcher) who generally tries to follow the speaker. Hence, though the field size of the shot is varying (from long shots to close-ups), most of the time "you see who you hear", see figure 3.3. Assuming this is true, we extract visual features from images of the persons appearing on the screen. It is worth mentioning that the live nature of the talk-shows makes our task quite challenging, especially owing to the potentially noisy sound conditions and the complete spontaneity of the speech.

Figure 3.3.: You see what you hear

Video features have historically been divided into colour descriptors, texture descriptors, shape descriptors and motion descriptors [MPEG-7 02]. Recent literature propose to represent the visual texture of images containing objects using order-less models. Order-less means that they retain only the frequencies of the individual features, and discard all information about their spatial layout. We hence chose to divide the video features into three traditional classes: *colour features*, *texture features* and *Object features*. In addition we add a call for *Movement features*. The video feature sets extracted are given in table 3.1.2 and described more thoroughly in appendix A.2.

Table 3.2.: Video features

| Family | Feature | Dimension |
|---|---|---|
| Colour | Image histogram | 48 |
| | Colour layout | 12 |
| | Colour structure | 32 |
| | Main dominant colour | 6 |
| | Mean dominant colour | 3 |
| | Scalable colour | 64 |
| | Face colour layout | 12 |
| | Face scalable colour | 64 |
| | Face main dominant colours | 6 |
| | Face mean dominant colour over 40% | 3 |
| | Face mean dominant colour over 60% | 3 |
| | Suit colour layout | 12 |
| | Suit scalable colour | 64 |
| | Suit main dominant colours | 6 |
| | Suit mean dominant colour over 40% | 3 |
| Object | Region shape | 35 |
| Texture | Edge histogram | 80 |
| | Gabor texture | 24 |
| | Haralick texture | 78 |
| Movement | Global movement intensity | 1 |
| | Global speed intensity | 1 |
| | Global acceleration intensity | 1 |
| | Face movement intensity | 2 |
| | Face movement orientation | 2 |
| | Face speed intensity | 2 |
| | Face speed orientation | 2 |
| | Face acceleration intensity | 2 |
| | Face acceleration orientation | 2 |
| | Suit movement orientation | 2 |
| | Suit movement intensity | 2 |
| | Suit speed orientation | 2 |
| | Suit speed intensity | 2 |
| | Suit acceleration orientation | 2 |
| | Suit acceleration intensity | 2 |
| | Head minus body movement intensity | 1 |
| | Head to body movement intensity | 1 |
| | Head minus body speed intensity | 1 |
| | Head to body speed intensity | 1 |
| | Head minus body acceleration intensity | 1 |
| | Head to body acceleration intensity | 1 |

### 3.1.3. Object detection

Some of the video features from the previous section are based on the fact that faces, and clothes, can be detected in an image as shown in figure 3.4. The use of a facial recognition system is rather difficult in our task due to the camera movements, the changing field size and angles of shot, the changing postures of the filmed persons and the varying lightning conditions. In order for the viewer not to confuse the participants on a TV set, their dress are usually carefully chosen. Our main assumption being that we often "see who we hear", we suppose that the information carried by the dress can help us for speaker identification and have the advantage that it can be extracted even more robustly than the persons facial features.

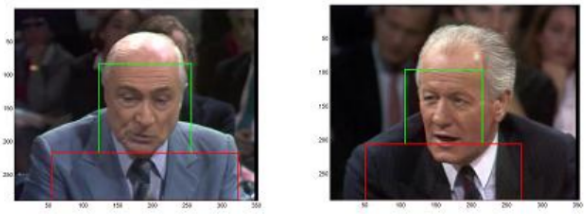


Figure 3.4.: Face and clothing detection

[Jaffré 04] proposes an efficient method for this detection, based on the well-known method of [Viola 01]. After detecting a face in an image, we use a heuristic approach to detecting the persons clothing. It is assumed that the content of a rectangle below the face mainly consists of that persons clothing. The rectangle is limited to a maximal width of 1.75 times the width of the detected face and 1 time the height of the face. We then extract colour and movement features within the area considered as face and clothing. The movement descriptor are based on the findings of [Lucas 81] and standard MPEG-7 descriptors.

## 3.2. Pre-processing

Data preprocessing is an important step in the machine learning system. Data gathering methods are often loosely controlled, resulting in out-of-range values, missing values, etc. Analysing data that has not been carefully screened for such problems can produce misleading results.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult [Liu 98]. Data preparation and filtering steps can take considerable amount of processing time. The product of data preprocessing is the final training set. [Kotsiantis 06] access issues of data preprocessing that can significantly affect the generalisation of machine learning systems. Some of the challenges in temporal integration for audio classification are assessed in [Joder 09] and [Kim 04].

As our features are recorded with different sampling rates and have different scales, we need to do feature integration and normalisation. These two steps are described in further detail in the following subsections.

### 3.2.1. Audio and video fusion

Audio features are extracted every 10 milliseconds (100 Hz) while video features are extracted every 40 milliseconds (25 Hz, because there are 25 images / second on TV broadcasts). The promise of instantaneous semantic access to multimodal video repositories has triggered much attention for methods that automatically index segments of interest. Typical semantic video analysis methods first extract a set of features from the raw data. Choices here include a feature extraction method based on unimodal analysis, i.e. using only one information stream, or multimodal analysis, i.e. using two or more information streams.

In general, three modalities exist in video, namely the audio, the textual, and the visual modality. A multimodal analysis method for semantic indexing of video inevitably includes a fusion step to combine the results of several single media analysis procedures. There are two general fusion strategies within the machine learning trend to semantic video analysis, namely: early fusion and late fusion. Indexing approaches that rely on early fusion first extract unimodal features. After analysis of the various unimodal streams, the extracted features are combined into a single representation. Indexing approaches that rely on late fusion also start with extraction of unimodal features. In contrast to early fusion, where features are then combined into a multimodal representation, approaches for late fusion learn semantic concepts directly from unimodal features. These outputs are combined afterwards to yield a final detection score.

We have chosen early fusion because we want to exploit the feature selection properties of multiple kernel learning. To be able to create one matrix with temporarily aligned features, the audio features are down-sampled by averaging to 25 Hz. The other approach would have been to up-sample the video features, but down-sampling is chosen to reduce the computational load. As 25 goes to 100 by multiplication of a whole number, no interpolation is needed and the new sample is calculated as the average of the 4 previous samples.

### 3.2.2. Normalisation

As we extract a great variety of features from very different domains, the features have very different dynamic ranges. This implies, that if applied directly, features with large values will have a larger influence on the cost function than features with small values, although this does not necessarily reflect their respective significance in the design of the classifier. The goal of this step is hence to normalise the individual components of the extracted feature vectors in such a way that the resulting (normalised) vectors are better suited for classification.

Several normalisation techniques exist today, but this subject often is overseen in research papers. By examining different normalisation approaches it is possible to in-

crease the performance of the system. Some examples of different techniques presented in [Struc 09] are "unit length normalisation", "zero-mean and unit-variance normalisation", "linear scaling to unit range" and "rank normalisation and gaussianisation". Besides the linear methods, non linear methods can be employed in cases where the data are not evenly distributed around the mean. The softmax scaling is a popular candidate of "squashing" the data to have values between 0 and 1 [Theodoridis 99].

The main goal of our efforts has not been to optimise the results, and thus a fairly simple approach was chosen: linear scaling to unit range. Equation 3.2.1 shows how the feature matrix is re-computed.

$$data = \frac{data - \min(data)}{|\max(data) - \min(data)|} \tag{3.2.1}$$

The Matlab code for such an approach is presented below. In words, all the resulting

```
1 min_norm = min(data);
2 max_norm = max(data);
3
4 data = (data - repmat(min_norm,size(data,1),1))./ ...
5     repmat(abs(max_norm -min_norm),size(data,1),1);
```

normalised features will have values between 0 and 1.

## 3.3. Training

After preprocessing the features we enter the training phase of our system. As described in chapter 2, this step has two main objectives, feature selection and classifier decision function learning. We apply multiple kernel support vector machines. This is an embedded approach that create the classifier, select the most discriminative features and select the most efficient kernels all in one.

### 3.3.1. Feature selection

Feature selection for machine learning purposes is the technique of selecting a subset $d$ of relevant features from set a $D$ (where $d \ll D$), for building robust learning models. By removing most irrelevant and redundant features from the data, feature selection helps improve the performance of learning models by (1) enhancing generalisation capability, (2) speeding up learning process and (3) improving model interpretability. The subset $d$ should thus contain a set of features that describes the data as good, or even better, than set $D$. Several common approaches to feature selection are described in section 2.3.

We use the embedded approach of multiple kernel learning, as described in section 2.3.2. This approach prefer the features that has the greatest absolute distances in the feature space for data points with different labels. The MKL approach presented in

section 2.3.1 manipulates entire kernels. In contrast, one might write each kernel as the sum of rank-one kernels,

$$K_i = \sum_{j=1}^{N} M_{ij} \tag{3.3.1}$$

before applying the approach by [Rakotomamonjy 08] to (a subset of) the collection of rank-one kernels, $M = \{M_{ij}\}$. If most of the discriminating information contained in the $K_i$ can be represented by a small number of $M_{ij}$, we anticipate that this procedure will produce better classifiers than procedures based on convex combinations of the $K_i$.

### 3.3.2. Learning the classifier

While the theory and motivation for multiple kernel learning support vector machine approach are presented in section 2, we will in this section given a survey of the parameterisation. The most important hyper parameters are the slack-variable weight, $C$, and the kernel parameters.

The slack-variable is introduced to account for outliers. In practice, the minimalisation problem in equation 2.2.5 page 12 changes to:

$$\min_{wb} \|w\|^2 + C \sum_i \xi_i \tag{3.3.2}$$

In other words, the term $C \sum_i \xi_i$ makes the optimal separating hyperplane less sensitive to the presence of outliers in the training set. A common procedure to create grid values for testing slack-variable has $C$ values in powers of 2, ranging from 1 to 1000. The optimal separating hyperplane tends to maximise the minimum distance $\frac{1}{\|w\|}$ for small $C$, and minimise the number of misclassified points for large $C$. For intermediate values of C the solution of equation 2.2.15 trades errors for a larger margin. For high values of C almost no outliers are accepted.

### 3.3.3. Kernel selection

Selecting the kernel function and its parameters is an important issue in training. In a kernel function k, the mapping function creates the feature space in which the relation between data points is defined as a linear combination of their features. Moreover, Multiple Kernel Learning (MKL) has emerged as a useful tool when the specific kernel is unknown, or when there are various sources of information to combine. MKL searches over a combination of kernels for the one most suited to the task.

In our system we apply the approach of [Rakotomamonjy 08], a training algorithm based on gradient descent, presented in greater detail in section 2.3.1. The output of this algorithm is a classifier and a set of weighted kernels. As opposed to other approaches, kernels that perform poorly are set to weight zero, i.e. not influencing the result. We have made some adjustments to the algorithm to be able to specify a set of kernels to a subset of features.

## 3.4. Testing and evaluation

After creating a classifier it is useful to know how well it performs. The test step of the system is conducted as a black box test. Black-box testing is a method that tests the functionality of an application as opposed to its internal structures or workings.

The previous step created a classifier, selected some features and selected one or more kernels. After extracting the selected features, the datapoints are mapped to the feature space with the kernels. The classifier now provides decision surface for the datapoints. We use the "MaxWins" voting strategy described in section 2.2.5 to produce the final output when more than one classifier is trained.

## 3.5. Post-processing by median filtering

The term post-processing is used for processes applied to result of the main classification. The goal is to improve the quality of the result by applying more or less heuristic approaches. Depending on which experiment we are executing we can take advantage of prior knowledge of the signal content to, for example do smoothing. Median filtering is one kind of smoothing technique. All smoothing techniques are effective at removing noise in smooth patches or smooth regions of a signal, but adversely affect edges.

The post processing step in our system applies a median filter to the output labels. A median filter is a non-linear digital filtering technique, often used to remove salt and pepper noise. The median filter is an effective method that can suppress isolated noise without blurring sharp edges.

The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighbouring entries. The pattern of neighbours is called the order of the filter, which slides like a window, entry by entry, over the entire signal. This means that single spikes among the labels will be substituted, hopefully leaving the result closer to the ground truth. For an odd length window $n$, value $y(k)$ is calculated between $x(\frac{k-(n-1)}{2})$ and $x(\frac{k+(n-1)}{2})$.

For the proposed tasks it is expected that the window length will be strongly dependent on the hierarchic level. A longer filter can be applied when searching for major concepts such as set-location or the nature of the audio (speech, music, etc.) than when searching for speakers that might intervene with sections of a few seconds only.

# 4. Experimental study

In this section we present the executed experimental study on our classification system (described in section 3. The goal of this step is, nevertheless, *not* to improve the overall performance of every existing state of the art system! We would, however, want to show that the multiple kernel learning support vector machines approach to training classifiers can be applied to a wide range of tasks. The performance is in most cases as good as, slightly better or slightly worse as state of the art adapted systems. We hereby propose one single solution to a wide set of problems.

Although there are many approaches to audio segmentation they are often focused on a narrow type of audio such as speech to music separation, speaker recognition and music structure extraction. These methods work well for these specific tasks but are not generalisable even, for example, across different genres of music. A method to segment any given audio file that contains different types of sounds remains an open and important problem [Zhang 09].

Three main groups of experiments have been proposed to tests our MKL classification system. The goal is to show that our system is generic and performs well independently on the task at hand. The first, low-level experiment, was speaker identification. The second, midlevel classification task, was to detect broad semantic classes such as speech, noise and music. Both audio and video features appeared useful in this process. The third, high-level classification task, was to classify the audio segments depending on whether it is recorded on stage, in a studio, outdoors, etc. In these experiments many visual features where used along with the audio features.

Our MKL classification system, is easy to adapt to these different evaluation tasks. All training and testing where performed on data taken from the Ina database, described in section 4.1.1.

An important issue is the choice of which kernel to apply to which subset of features. This seems to be a neglected aspect in many state of the art kernel based classification systems, as Gaussian kernels generally are performing reasonably well. Exceptions to this generalisation does nevertheless exist, but this is an unexplored research area.

## 4.1. Evaluation environment

In this section we present the most important framework factors, the database, the performance measures and the testing protocol.

### 4.1.1. Database

The database of audiovisual documents is provided by Institut National de l'Audiovisuel (Ina). More specifically the provided database consist of more than 50 French language (but only 7 are assessed in this study), Grand Echiquier TV talk-show programmes from the 1970-80s. Each show focuses on a main guest, and other supporting guests, who are interviewed by a host presenter. The interviews are punctuated with film excerpts, live music and other performances.

Table 4.1.: Grand Echiquier database

| | |
|---|---|
| No. of shows | 7 |
| Av. Evaluation time | 147 min. |
| Av. Total speech | 50 min. |
| Av. No. of segments | 1033 |
| Av. segment length | 3 sec. |
| Av. Overlap | 5 min. |
| Av. No. speakers | 13 |
| most active | 1476 sec. |
| least active | 7 sec. |

Quantitative attributes of the database are summarised in Table 4.1. The average show length for the Grand Echiquier data-set is 147 minutes. On average there are 50 minutes of speech per show (i.e. with noise, music and film excepts removed). There are an average of 1033 speech segments per show with an average length of 3 seconds , where the speech overlaps on average 5 minutes (10%) per show.

On average there are 13 speakers per television show. The average speaking time for the most active speaker is 1476 seconds for the GE data-set and corresponds to the host presenter. The average speaking time for the least active speaker is only 7 seconds and corresponds to one of the minor supporting guests. Speakers with such little data are extremely difficult to detect and thus this aspect of the TV show data-set is likely to pose significant difficulties for speaker identification. Furthermore, the presence of one or two dominant speakers means that lesser active speakers will be comparatively harder to detect, even if they too have a significant floor time.

The data is available in MPEG2 format and the audio is sampled at 12.8 kHz due to a very narrow bandwidth in the original files. The audio track is extracted from the video, converted to mono by averaging right and left channels and down-sampled. Since video features are computed every 40 ms (25 frames/sec), we use temporal integration, 4

consecutive audio frames being temporally averaged, so that the audio and visual features are available at the same rate (25 Hz). We also assume that they can be considered as synchronous.

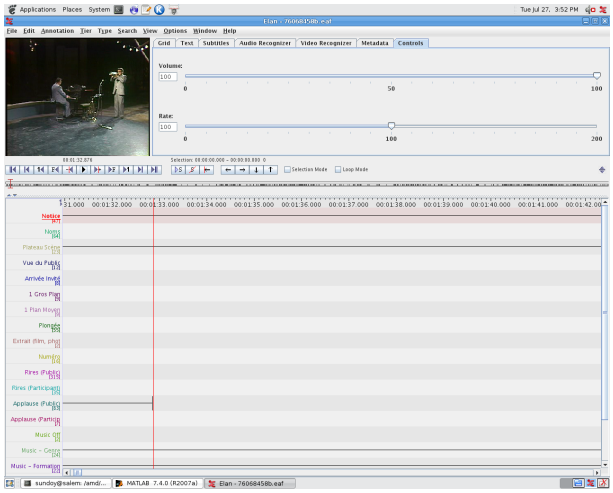


Figure 4.1.: Grand Echiquier annotations visualised in Elan

The database presents numerous characteristics and challenges that have made it popular among both French and other European multimedia research projects, e.g. the "European K-space network of excellence" and "Inform@gic".

### 4.1.2. Presenting results

In this section we will give a brief presentation of the different evaluation measures used to present the results in section 4.

#### Precision and Recall

Precision for a given class is the number of true positives, in other words it is the number of items correctly labelled as belonging to the class, divided by the total number of elements labelled as belonging to that class.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{4.1.1}$$

Recall is defined as the number of true positives divided by the total number of elements that actually belong to the class.

$$\text{Recall} = \frac{tp}{tp + fn} \tag{4.1.2}$$

In a classification task, a precision score of 1.0 for a class C means that every item labelled as belonging to class C does indeed belong to class C (but says nothing about

the number of items from class C that were not labelled correctly) whereas a Recall of 1.0 means that every item from class C was labelled as belonging to class C (but says nothing about how many other items were incorrectly also labelled as belonging to class C).

#### F-measure

Usually, Precision and Recall scores are not discussed in isolation, but are combined into a single measure such as the F-measure. The F-measure is the weighted harmonic mean of precision and recall. The F-measure can be calculated as shown in equation 4.1.3.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.1.3}$$

This is also known as the F1 measure, because recall and precision are evenly weighted. Several different weightings schemes exists, but we have chosen too keep the precision and recall unweighted.

#### Confusion matrix

A confusion matrix [Kohavi 98] contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix.

When a data set is unbalanced (when the number of samples in different classes vary greatly) the error rate of a classifier is not representative of the true performance of the classifier. This can easily be understood by an example: If there are 990 samples from class A and only 10 samples from class B, the classifier can easily be biased towards class A. If the classifier classifies all the samples as class A, the accuracy will be 99%. This is not a good indication of the classifier's true performance. The classifier has a 100% recognition rate for class A but a 0% recognition rate for class B.

A confusion matrix is a visualisation tool where each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mis-labeling one as another). We have added precision and recall rates to the outer cells of our confusion matrices. The bottom right cell holds the global error rate.

### 4.1.3. Protocol

Since our goal is to find the classifier having the best performance on new data, the simplest approach to the comparison of different classifiers is to evaluate the f-measure using data which is independent of that used for training. In order to perform statistically valid results we need to subdivide our database into "training", "validation", and "test" sets. The distinctions among these subsets are crucial and should the sets should always be non-overlapping, but the terms "validation" and "test" sets are often confused.


First, a classifier is trained with standard parameters. The performance of the classifier is measured by evaluating the F-measure of an independent validation set. After repeating this step for a wide set of parameters, local optima can be found. The classifier having the smallest error with respect to the validation set is selected. Since this procedure can itself lead to over-fitting to the validation set or training, the performance of the selected classifier should be confirmed by measuring its performance on a third independent set of data called the test set.

The three different sets are defined by [Ripley 96].

**Definition 4.1.** *Training set: A set of examples used for learning, that is to fit the parameters of the classifier.*

**Definition 4.2.** *Validation set: A set of examples used to tune the parameters of a classifier.*

**Definition 4.3.** *Test set: A set of examples used only to assess the performance [generalisation] of a fully-specified classifier.*

The crucial point is that a test set is never used to choose among two or more classifiers, so that the error on the test set provides an unbiased estimate of the generalisation error (assuming that the test set is representative of the full database). Any data set that is used to choose the best of two or more classifiers is, by definition, a validation set, and the error of the chosen network on the validation set is optimistically biased.

We use the manually labelled data from Ina as the ground truth in our experiments, a general description of the database used can be found in the previous section (4.1.1). The results compared to this ground truth are presented as confusion matrices with the precision and recall rates computed in the rightmost and lower cells. The f-measure (a unweighted combination of precision and recall) is given in the bottom right cell. See sections 4.1.2 and 2.4 for further details on how these values are computed. In the tables the left matrix is the result of using the same data-set for training and testing and has, as expected, better results than the right matrix, where the selected test-set is used for testing.

Both the data used for creating the classifier and a separate data sets are applied to the classifier and the results are presented in side by side matrices. The left matrices has the result of the classification of the data used for training the classifier, while the right

| | Predicted class | |
|---|---|---|
| Actual class | | Recall rate |
| | Precision rate | FM |

matrix has the results of applying the classifier to a validation or test-set (specified in each case). Generally a classifier should perform very well on the data used for training, without reaching 100%. As we have seen in section 2, the support vector machine learning trains an optimal, but not necessarily perfect classifier.

## 4.2. Experimental results

In this section we present the experimental results found with our classification system. We remember the most important parameters, $C$ and $\sigma$, frequently modified in our experiments.

The $C$ value is the slack-variable as described in section 2.2.2 and can be regarded as a regularisation parameter.

$$\min_{wb} \|w\|^2 + C \sum_i \xi_i \tag{4.2.1}$$

The optimal separating hyperplane tends to maximise the minimum distance $\frac{1}{\|w\|}$ for small $C$, and minimise the number of misclassified points for large $C$. For intermediate values of C the solution of equation 4.2.1 trades errors for a larger margin.

The $\sigma$ value is the width of the gaussian kernel. We do, as [Essid 05], use a feature dimension weigthed expression for the gaussian kernel in our experiments. This changes the expression in section 2.2.3 to

$$k(x, y) = \exp\left(-\frac{\|x - y\|}{d\sigma^2}\right) \tag{4.2.2}$$

where $d$ is dimention of the features under evaluation. Variables $d$ and $\sigma$ are often combined into a single parameter $\gamma$.

$$\gamma = \frac{1}{d\sigma^2} \tag{4.2.3}$$

Equation 4.2.2 can thus be written as:

$$k(x, y) = \exp -\gamma\|x - y\| \tag{4.2.4}$$

For the speaker identification task we present a series of development and testing steps, with carefully selected training and testing data-sets. For audio segmentation we first present some preliminary tests on different shows before we display the results of a inter-show validation. For the high level concept detection task we present some simple tests performed on different shows. The selection of training and testing data is described for every experiment.

The results are presented with precision, recall and F-measure as given in section 4.1.2.

### 4.2.1. Speaker identification

Content structuring or segmentation is an essential process in a number of application domains such as automatic video archiving and retrieval, scene categorisation or automatic summary generation. In specific audiovisual scenes such as those of TV talk-shows, an automatic segmentation typically aims to obtain meaningful and semantically rich segments such as “musical passage”, “film excerpt”, “main guest on screen” or “arrival of a new guest”. From this viewpoint, speaker identification can be regarded as an important tool among others, contributing to the construction of a semantic segmentation.

Speaker recognition is the process of automatically recognising who is speaking on the basis of individual information included in speech waves. This technique makes it possible to use the speaker's voice to verify their identity and control access to services such as voice dialling, banking by telephone, telephone shopping, database access services, information services, voice mail, security control for confidential information areas, and remote access to computers.

Speaker recognition can be classified into speaker identification and speaker verification. *Speaker identification* is the task of identifying the exact person who is speaking. The speaker is initially unknown, and must be determined after being compared to templates. There can often be a very large number of templates that are involved in identifying a speaker, as it is difficult to correctly identify a speaker. *Speaker verification* is the task of determining if the speaker is who he or she claims to be. The speaker's voice is compared to only one template: the person who he or she is claiming to be.

Our preliminary studies where all executed as speaker identification experiments. Three speakers from a "Grand Echiquier" episode are shown in figure 4.2.

Figure 4.2.: Leonard Cohen, Raymond Devos and Christian Duvalex

We assume that there is no training database with examples of every speaker available in advance that can be used to learn classifiers in a totally supervised fashion. Our approach can, nevertheless, be considered as "semi-automatic" in the sense that the training data is collected on the fly by the operator of our system, for instance the Ina archivist handling a given talk-show, who is asked to arbitrarily select one video excerpt of each speaker involved. These excerpts are then used to learn classifiers that are subsequently applied on the whole show to perform speaker identification at every time instant. It is worth mentioning that when increasing the complexity of the multiple kernel learning support vector machine (number of kernels, number of speakers and number of features) the need for training data increases as well. The manual selection of excerpts by the operator is greatly simplified by the two following facts. First, the total number of speakers to look for is known a priori since a short textual notice, giving a list of all guests to be identified is available for every talk-show video. Second, the operator can quickly locate examples of the different speakers using visual inspection through a temporal slider and / or the fast forward video mode. This is the same protocol as used in [Vallet 10].

It is easy to think of other application areas of this "semi-automatic" approach. One could, for example, imagine going to a meeting and signing in by speaking into a microphone for a few seconds or for people with hearing impairs, one could single out close friends and family.

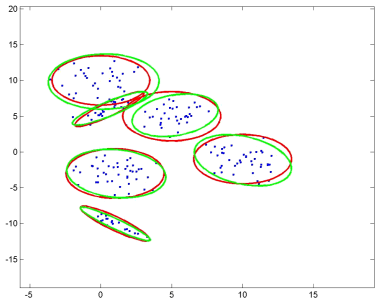


Figure 4.3.: Gaussian Mixture Models

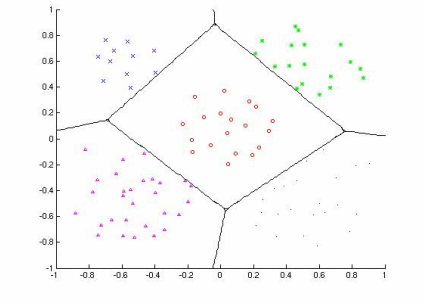


Figure 4.4.: A linear classifier

An alternative to our approach is to use a speaker diarisation system. Speaker diarisation is the process of annotating an input audio channel with information that attributes (possibly overlapping) temporal regions of signal energy to their specific sources. More specifically it can be described as the unsupervised identification of each individual speaker and in what time frame(s) the speaker(s) can be found. However, since speaker diarisation is totally unsupervised, the operator would still need to assign a speaker to each cluster with the inconvenience that some speakers may not be found among the clusters determined automatically (as these systems are not perfect), hence our approach is not necessarily more costly in terms of human operator efforts. The first step in a state of the art speaker diarisation process is speech activity detection. Only voiced areas of the signal are considered in the segmentation and clustering. There are two main approaches to segmentation and clustering, bottom-up or topdown. The difference is weather the signal is segmented and then adding the segments to clusters or the opposite. The state of the art systems, such as [Bozonnet 10], calculate a Gaussian Mixture Model for each potential speaker.

When the classifier has been created you can test unknown data by extracting the features used to create the feature space and simply see what side of the classifiers the test is. [Ramona 10] show that once the SVM classifiers are trained, unknown data can be classified in real time.

The speaker identification experiments have the intention to separate speakers from each other. We have treated one show at the time. A typical show has a duration of about 2 to 3 hours, but our test are limited to the sections classified as speech. Depending on the show, this leaves 45 minutes 1 hour and 30 minutes of speech from about ten speakers. Two of them (the host and the main guest) are vastly over represented, speaking about 60 percent of the time. A frequently used show, named "CPB85104049" is described in detail in table 4.2.

These experiments started with creating a small-scale training and verification subset

from the Ina-database. In order to create a *small scale* data-set we selected 15000 instances (corresponding to 11 minutes and 40 seconds) from three major speakers from the show named "CPB85104049", giving 5000 instances per speaker. The first half of the instances (2500 instances) was used for training and the second half was used for validation. We extracted 50 frequently used features, in an attempt to have reasonably representative results. The features were:

Audio: 13 Mel frequency cepstrum coefficients
Audio: 10 Line spectral frequencies
Video: 6 Mean dominant colour (suits)
Video: 21 Motion features

All features are described in section 3.1.

There are several strategies for how to find the optimal parameters for a support vector machine training. We chose to run a series of experiments, changing one parameter at the time. The small scale data was used for these experiments.

Table 4.2.: Grand Echiquier CPB85104049

| | |
|---|---|
| Total duration | 5745s |
| No. speakers | 10 |
| No. speak sections | 64 |
| No. turns | 1048 |
| Av. turn duration | 5.3s |
| Standard deviation av. turn duration | 7.1s |
| Longest speech turn | 92s |
| Shortest speech turn | 0.2s |

Figure 4.5 shows how two state of the art algorithms differ for the same bandwidths ($\frac{1}{\gamma}$) of a Gaussian kernel. LIBSVM [Chang 01] (in blue) seems to have the higher overall performance than SimpleMKL [Rakotomamonjy 08] (in red). The impact of normalisation of the data is shown as the difference to the dotted lines. We have the best results for normalised data in small sigma values, while the results for unnormalised data seems more stable over wider ranges of $\sigma$. This would suggest that the the "selected" descriptors, the ones with higher numerical values, have some a classification property that is more stable for a large range of $\sigma$. Nevertheless, the important part of this graph is the difference between the two algorithms. The fact that LIBSVM uses a different optimisation approach than e.g. SimpleMKL, may explain the difference in result for two algorithms. This may seem as a strong incentive to continue with LIBSVM and leave the newer SimpleMKL behind. However, SimpleMKL is, as the name suggests able to do multiple kernel learning within the algorithm and one of the main goals of this work is to explore the use of multiple kernels. We will therefore continue to use SimpleMKL, hoping that several kernels may compensate for the fact that the optimisation strategy seems to have a worse performance than LIBSVM.

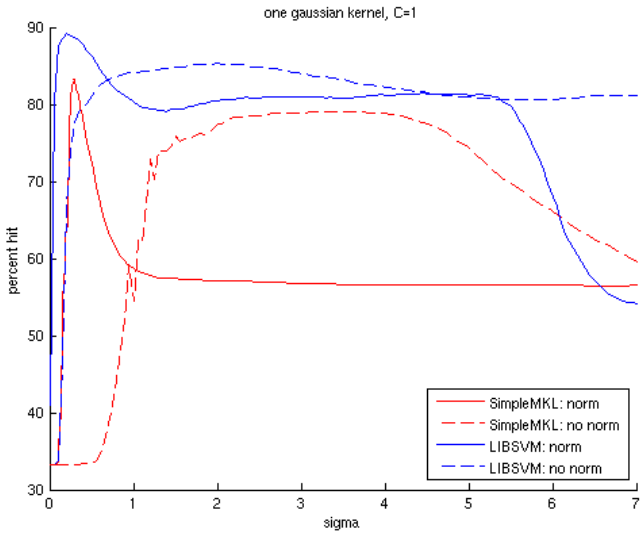



Figure 4.5.: SimpleMKL and LIBSVM, C=1, 1 Gaussian kernel

Choosing the parameters that gave the highest value (88.81%) for the LIBSVM algorithm ($C = 1$ and $\frac{1}{\gamma} = 2^{-2}$), we ran the SimpleMKL algorithm. The results are presented in confusion matrices where the columns represent the actual speakers while the rows shows the result of the classification. The confusion matrix in table 4.3 to the left is the result of using the same data-set for training and validation, while the matrix to the right has the actual validation-set.

Table 4.3.: CPB85104049, $\frac{1}{\gamma} = 0.25$, $C = 1$

| | spk 1 | spk 2 | spk 3 | Recall |
|---|---|---|---|---|
| spk 1 | 30% | 3% | 1% | 99% |
| spk 2 | 0% | 32% | 1% | 97% |
| spk 3 | 0% | 1% | 33% | 98% |
| Precision | 99% | 99% | 99% | 97% |

| | spk 1 | spk 2 | spk 3 | Recall |
|---|---|---|---|---|
| spk 1 | 25% | 7% | 2% | 74% |
| spk 2 | 4% | 27% | 2% | 84% |
| spk 3 | 1% | 1% | 31% | 93% |
| Precision | 83% | 79% | 93% | 85% |

The results for the third speaker are quite good, having a precision and recall of 93%. The first two speakers, however, have lower performance, both in terms of precision and recall. We would therefore like to explore the parameters further and hence executed a large grid of Gaussian kernel widths, $\sigma$, and slack-values, $C$. In figure 4.6 the overall accuracy (y-axis) for single Gaussian kernels support vector machines for different slack-variables (x-axis) and widths values (graphs) are displayed.

The validation results presented in figure 4.6 show how some kernel widths, such as $\frac{1}{\gamma} = 2^{-4}$, is unable to create a feature space that makes the data-points linearly separable. The result of this kernel is equal to random guessing, $\frac{1}{3}$, as there are three speakers. The

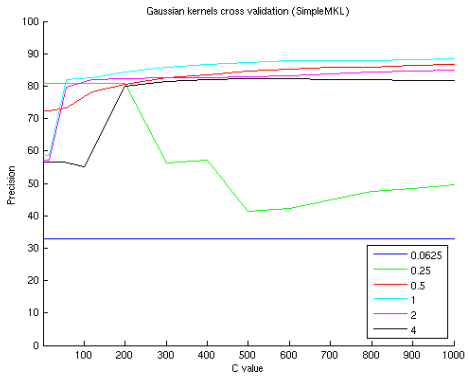



Figure 4.6.: Gaussian kernels

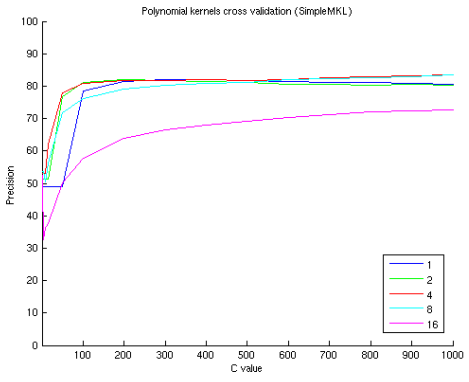



Figure 4.7.: polynomial kernels

kernels with widths $\frac{1}{\gamma} = \frac{1}{2}$, $\frac{1}{\gamma} = 1$, $\frac{1}{\gamma} = 2$ and $\frac{1}{\gamma} = 4$, however, are performing increasingly better for increasing values of the slack-variable, reaching a stable high value just above 80% for slack-variable, $C = 100 - 200$.

We did the same validation procedure with five polynomial kernels on the same dataset. The two parameters we assessed was the polynomial kernel degree, $\delta$, and the slack-value, $C$. These results are presented in figure 4.7, where the overall accuracy (y-axis) for different slack-variables (x-axis) and polynomial kernel degree (graphs) are displayed. These kernels have an equal performance to the Gaussian kernels, accounting for percentage variance, a result which is well known in the scientific community.

As described in detail in section 2.3 applying multiple kernels at the same time we can deduce the optimal kernel parameters and the features needed.

Still using the dataset from the show named "CPB85104049", we used five Gaussian kernels with widths $\frac{1}{\gamma} = [2^{-2}, 2^{-1}, 3 * 2^{-2}, 2^{0}, 5 * 2^{-2}]$ and slack variable $C = 50$. After training, the kernel weighting suggested clearly that the Gaussian kernel with $\frac{1}{\gamma} = 3 * 2^{-2}$ got $\frac{2}{3}$ of the weight the Gaussian kernel with $\frac{1}{\gamma} = 1$ got the remaining $\frac{1}{3}$ when creating the optimal classifier. This corresponds to the results provided in the validation, table 4.6.

The results of classifying the training data-set is displayed in the left confusion matrix of table 4.4, while the confusion matrix on the right has the values for classifying the validation-set.

Table 4.4.: CPB85104049, $\frac{1}{\gamma} = [2^{-2}, 2^{-1}, 3 * 2^{-2}, 2^{0}, 5 * 2^{-2}]$, $C = 50$

| | spk 1 | spk 2 | spk 3 | Recall |
|---|---|---|---|---|
| spk 1 | 23% | 10% | 0% | 69% |
| spk 2 | 2% | 32% | 0% | 91% |
| spk 3 | 2% | 2% | 29% | 88% |
| Precision | 82% | 72% | 97% | 84% |

| | spk 1 | spk 2 | spk 3 | Recall |
|---|---|---|---|---|
| spk 1 | 23% | 10% | 1% | 69% |
| spk 2 | 3% | 30% | 0% | 91% |
| spk 3 | 1% | 4% | 28% | 83% |
| Precision | 85% | 68% | 96% | 82% |

This shows that the chosen parameters are indeed creating good classifiers for the show named "CPB85104049" (validation). The strong point of this approach is that we learned the optimal kernel parameters for a given slack-values without searching the large grid of parameters separately one-by-one. To improve the results further we set the set of well performing kernels to each subset of features (descriptors). The first subset consist of 13 mel frequency cepstrum coefficients, the second 10 line spectral frequencies, the third 6 mean dominant colour and 21 motion features. By looking at the weighting of kernels in table 4.5 after training we can deduce which features are the most discriminative.

Table 4.5.: CPB85104049 Feature selection

| Descriptor | kernel $\frac{1}{\gamma}$ | weight % |
|---|---|---|
| MFCC | 0.5 | 22 |
| | 1 | 0 |
| | 2 | 0 |
| LSF | 0.5 | 3 |
| | 1 | 0 |
| | 2 | 0 |
| Video | 0.5 | 65 |
| | 1 | 10 |
| | 2 | 0 |

For the first and second descriptors the kernel with $\frac{1}{\gamma} = 0.5$ was the only one used, and it got respectively 22% and 3% of the total weight. For the third descriptor the kernels with $\frac{1}{\gamma} = 1$ and $\frac{1}{\gamma} = 2$ got respectively 65% and 10%. These results suggest that the video-features are very discriminative.

In table 4.6 we have presented the validation results after feature selection.

Table 4.6.: CPB85104049, $\frac{1}{\gamma} = [0.5, 1, 2]$, $C = 300$, after feature selection

| | spk 1 | spk 2 | spk 3 | Recall |
|---|---|---|---|---|
| spk 1 | 26% | 7% | 1% | 77% |
| spk 2 | 3% | 30% | 0% | 90% |
| spk 3 | 1% | 2% | 30% | 90% |
| Precision | 86% | 77% | 96% | 86% |

These results are by far the best results obtained on the validation set and is thereby a first indication of that our multiple kernel learning approach is able to outperform standard support vector machine approaches to standard problems.

A short cross validation procedure was then applied to simulate an real life situation. As described earlier in this section we imagine an operator selecting a segment of speech from each speaker appearing in the show. Our approach can thus be considered as "semi-automatic". This is simulated by training a classifier on randomly selected segments

among the longest segments of speech from each speaker. The rest of the speech sections are used for validating the classifier. The parameters $\frac{1}{\gamma}$ and $C = 300$ were set for all experiments.

A 4 fold cross validation was performed on the various candidate speech segments to assess the validity of the results. We experienced an increase in accuracy of $3,7 \pm 4$ percentage points when using MKL feature selection as opposed to standard MKL.

### 4.2.2. Audio segmentation

Audio segmentation is a process that divides an audio file into its composite sounds. Each segment or clip should consist of a single type of sounds that is acoustically different from other parts of the audio file. An accurate segmentation process can identify appropriate boundaries for partitioning given audio streams into homogeneous regions. The problem of distinguishing speech signals from other audio signals has become increasingly important as automatic speech recognition systems are applied to more real-world multimedia domains e.g. internet data. Standard speech recognisers attempting to perform recognition on all input frames will naturally produce high error rates with mixed input signal. A common issue is that speech is confused with segments of music or background noise [Ajmera 03]. Therefore, a preprocessing stage that segments the signal into periods of speech and non-speech is invaluable in improving recognition accuracy. This also has the benefit of reducing overall computational load, as the full speech recognition system is only enabled for speech segments.

Figure 4.8.: music, speech and applause

Another application of audio segmentation is low bit-rate audio coding [Brandenburg 97]. Traditionally, separate codec designs are used to digitally encode speech and music signals. An effective speech to music discrimination decision will enable these to be merged in a universal coding scheme capable of reproducing well both speech and music. More generally, audio segmentation could allow the use of automatic speech recognition acoustic models trained on particular acoustic conditions, such as wide bandwidth speech (high quality microphone input) versus telephone narrow bandwidth speech, music, applause and noise, thus improving overall performance of the resulting system.

The goal of our audio segmentation is to classify the segments depending on whether there are music, applause, speech, etc. In figure 4.8 we have shown a screen-shot from such audio moments. The audio segmentation experiments have the intention to separate mid-level classes such as “Silence”, “Music” and “Speech”.

We first present some preliminary audio segmentation experiments with our system. These figures are not cross-validated, but simply the result of applying our approach to two different shows from “Grand Echiquier”.

We created a training-set by selecting the first half of the Grand Echiquier TV show named “CPB85104049”, the second half was used for testing. We then created a subset

from the training data, containing only the data labelled as "silence", "music" and "speech" and repeated this procedure for the validation data. This resulted in 59 minutes and 1 second of training data and 59 minutes and 6 seconds of validation data. Moreover, we saw that this show only have a total of 24 seconds of audio and video labelled as "silence", and knowing that silence normally is detected with a specific detector, we settled for the labels "music" and "speech" only.

We then extracted some frequently used audio features (MFCC's):

Audio: 13 Mel frequency cepstrum coefficients
Audio: 13 $\Delta$ Mel frequency cepstrum coefficients
Audio: 13 $\Delta\Delta$ Mel frequency cepstrum coefficients

After extracting the features we trained a classifier based on five Gaussian kernels with $\frac{1}{\gamma} = [2^{-2}, 2^{-1}, 3 * 2^{-2}, 2^{0}, 5 * 2^{-2}]$ and a slack-variable $C = 500$. After training, the Gaussian kernel with $\frac{1}{\gamma} = 2^{-2}$ got $\frac{2}{3}$ of the weight the Gaussian kernel with $\frac{1}{\gamma} = \frac{1}{2}$ got the remaining $\frac{1}{3}$ when creating the optimal classifier. The results of classifying the training data-set is displayed in the left matrix of table 4.7, while the matrix on the right has the values for classifying the test-set.

Table 4.7.: CPB85104049, $\frac{1}{\gamma} = [2^{-2}, 2^{-1}, 3 * 2^{-2}, 2^{0}, 5 * 2^{-2}]$, $C = 500$

| | music | speech | Recall |
|---|---|---|---|
| music | 41% | 7% | 85% |
| speech | 4% | 48% | 93% |
| Precision | 92% | 86% | 89% |

| | music | speech | Recall |
|---|---|---|---|
| music | 55% | 1% | 98% |
| speech | 40% | 4% | 10% |
| Precision | 57% | 83% | 61% |

After classifying the validation data we applied a median filter of length 500 instances. These results are presented in the left matrix of figure 4.7.

As can be seen from the matrix, most of (95%) the validation data has been classified as "music". We have a very high recall rate (98%) for this label, but a mediocre precision (57%). For the other label, "speech", we thus have a very low recall of only 10%, but it is encouraging that the precision is as high as 83%, meaning that most of what is found to be "speech" really is speech. A logical strategy to improve results like these is to increase the slack-variable, $C$, leaving less room for outliers.

In table 4.8 results are presented for an increased slack-variable $C$ to 1000. As the weighting suggested that the lower range kernels gave the optimal classifier, we chose new Gaussian kernels with $\frac{1}{\gamma} = [2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}]$. After training, the Gaussian kernel with $\frac{1}{\gamma} = 2^{-2}$ got, as before, most of the weight (73.96%) while the Gaussian kernel with $\frac{1}{\gamma} = \frac{1}{2}$ got the remaining 26.04% when creating the optimal classifier.

The results presented in table 4.8 are very good, with f-measure as high as 89% for the validation dataset. There still is a small section of "speech" data being classified as "music", but these are margins that are very small and neglectable as this is a single test without cross-validation. We have nevertheless used the multiple kernel approach to find

Table 4.8.: CPB85104049, $\frac{1}{\gamma} = [2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}]$, $C = 1000$

| | music | speech | Recall |
|---|---|---|---|
| music | 48% | 1% | 98% |
| speech | 0% | 51% | 99% |
| Precision | 99% | 98% | 99% |

| | music | speech | Recall |
|---|---|---|---|
| music | 52% | 3% | 94% |
| speech | 0% | 45% | 99% |
| Precision | 99% | 93% | 96% |

very well performing parameters on a validation set for the task of audio segmentation. The only features assessed where 39 mel frequency cepstrum coefficients.

To test the parameters, a full cross-validation procedure should be performed. The figures presented in figure 4.9 are a 3-fold validation procedure of the parameters deduced on the development. We chose the first half of one show, named "CPB84052346", and created a training-set by selecting only the data labelled as "music", "applause" and "speech". This resulted in 1 hour, 11 minutes and 43 seconds of training data. The second half of "CPB84052346" was used for testing. Similarly, the show named "CPB85104049" was cut in two, creating two test-sets.

Table 4.9.: CPB84052346, $\frac{1}{\gamma} = [2^{-3}, 2^{-2}, 2^{-1}, 0.75, 1]$, $C = 1000$, filtered

| | music | applause | speech | Recall |
|---|---|---|---|---|
| music | 32% | 0% | 2% | 95% |
| applause | 0% | 7% | 0% | 100% |
| speech | 0% | 0% | 59% | 100% |
| Precision | 100% | 99% | 98% | 98% |

### 4.2.3. High level concept detection

A scene is defined as one or more consecutive shots that they are semantically correlated, or they all share the same content for action, place and time. While shots are marked by physical boundaries, scenes are marked by semantic boundaries, so scene boundary detection is a far more difficult task compared with shot boundary detection. The content of a shot is essentially characterised by two factors: the underlying scene, and the camera work. As a result, the methods for automatic or semi-automatic shot characterisation have focused on generating attributes or features which capture shot content for one or both of the above factors.

Figure 4.9.: scene view, film extrait and audience view

The problem is important for several reasons: automatic scene segmentation is the first step towards greater semantic understanding of the film, breaking up the film into scenes will help in creating film summaries, thus enabling a nonlinear navigation of the film, and the determination of visual structure within each scene (e.g. dialogues), will help in the process of visualising each scene in the film summary. Finding semantically meaningful parts of a story (i.e., scene detection) is a necessary process in multimedia indexing, since such an analysis is a crucial step in reaching from low-level features, like colour, motion, voice pitch, to the high-level semantic descriptions of the content. This challenging goal can be simplified by defining a context in which the connections between low-level features and high-level descriptions are well-established in comparison to general cases.

The high level concept detection have the intention to separate high level classes such as “Scene View”, “Big Pan”, “Extract (film, photo, document)” and “Applause (Public)” , as shown in figure 4.9.

We have done some preliminary high level concept detection experiments with our system. These figures are not cross-validated, but simply the result of applying our approach to two different shows from “Grand Echiquier”.

We selected all data in the show named “CPB85104049” that has been annotated as “Scene View”, and and equally large data-set of data that comes from sections annotated as anything else than “Scene View”. To create a train data-set we selected 50% from each of the subsets, giving 1 hour 8 minutes and 35 seconds (102881 instances) of labelled audio and video. Both audio and video features where extracted:

Audio: 13 Mel frequency cepstrum coefficients
Audio: 13 $\Delta$ Mel frequency cepstrum coefficients
Audio: 13 $\Delta\Delta$ Mel frequency cepstrum coefficients
Video: 12 Colour Layout
Video: 32 Colour Structure

All features are described in section 3.1.

We then trained a classifier using 5 Gaussian kernels $\frac{1}{\gamma} = [2^{-4}\, 2^{-2}\, 2^{-1}\, 2^{0}\, 2^{1}]$ and a slack variable, $C = 50$. After training, the kernel weighting suggested clearly that the Gaussian kernel with $\frac{1}{\gamma} = 1$ was the one creating the best classifier. In table 4.10, to the left, the trained classifier is applied to the data-set used to train the classifier.

We then applied the classifier to the remaining 50% of the same show, giving 1 hour 8 minutes and 32 seconds (102819 instances) of labelled audio and video. The validation results are presented to the right in table 4.10. In the post processing we have added

Table 4.10.: CPB85104049, $\frac{1}{\gamma} = [2^{-4}\, 2^{-2}\, 2^{-1}\, 2^{0}\, 2^{1}]$, $C = 50$

| | other | Scene View | Recall |
|---|---|---|---|
| other | 51% | 2% | 95% |
| Scene View | 5% | 42% | 89% |
| Precision | 91% | 94% | 92% |

| | other | Scene View | Recall |
|---|---|---|---|
| other | 38% | 9% | 81% |
| Scene View | 8% | 45% | 83% |
| Precision | 81% | 83% | 82% |

a median filter with increasing length (see section 3.5). In figure 4.10 the blue line represents the filtered result of the experiment presented in table 4.10 (visualised as the red line). The best result can be found for a filter with length 29 seconds, giving a F-measure increase of 2 percentage points. The confusion matrices after filtering is given

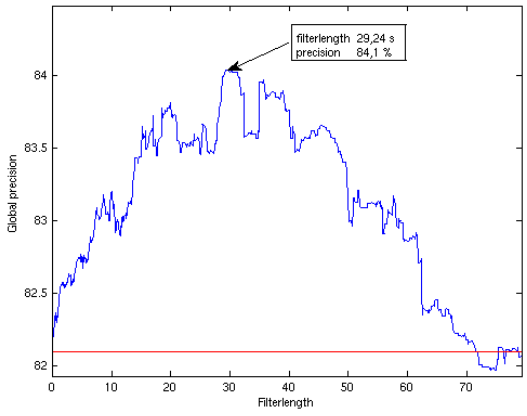



Figure 4.10.: CPB85104049 precision for increasing filter length

in table 4.11. The good results for long median filters (from 10 to 50 seconds) can be explained by the fact that we are segmenting events that stretch out in time. A typical

Table 4.11.: CPB85104049 $\frac{1}{\gamma} = [2^{-4}\, 2^{-2}\, 2^{-1}\, 2^{0}\, 2^{1}]$, $C = 50$, filtered

| | other | Scene View | Recall |
|---|---|---|---|
| other | 52% | 2% | 97% |
| Scene View | 1% | 45% | 97% |
| Precision | 97% | 96% | 96% |

| | other | Scene View | Recall |
|---|---|---|---|
| other | 39% | 8% | 83% |
| Scene View | 8% | 45% | 85% |
| Precision | 83% | 85% | 84% |

scene from Grand Echiquier is where the host is talking to the main guest or music is performed from the stage. In these two cases the scene may rest the same for several minutes.

We then moved on to testing the deduced parameters. We selected all data in the show named "CPB82055196" that has been annotated as "Scene View", and and equally large dat-set of data that comes from sections annotated as anything else than "Scene View". To create a train data-set we selected 50% from each of the sets, giving 1 hour 15 minutes and 30 seconds of labelled audio and video. We extracted the same features as for the validation phase. The result of applying the classifier to the data used for training gives the results in the left matrix in table 4.12. In the right matrix of table 4.12 we have

Table 4.12.: CPB82055196 $\frac{1}{\gamma} = [2^{-4}\, 2^{-2}\, 2^{-1}\, 2^{0}\, 2^{1}]$, $C = 50$, filtered

| | other | Scene View | Recall |
|---|---|---|---|
| other | 57% | 2% | 97% |
| Scene View | 22% | 19% | 46% |
| Precision | 72% | 90% | 76% |

given the results of classifying the selected data from the show named "CPB82055196" not used for training. This is 1 hour 15 minutes and 18 seconds of unlabelled audio and video. The trained classifier does indeed have a high precision (90%) for the chosen label, "Scene View". This means that the instances classified as "Scene View" almost always are "Scene View" according to the ground truth. Nevertheless, the classifier only managed to classify 46% of the "Scene View" instances available in the test set. This means that the trained classifier does not manage to generalise, and further hyper parameter investigation is needed.

The results for the high level concept detection segmenting do not look very promising, but are only preliminary as no cross validation procedure has been executed. At the time of writing this report, several experiments are still executing. Larger cross-validation multiple kernel learning experiments are time demanding processes, but we do expect to show more results at the masters thesis defence.

# 5. Conclusion

The objective of this thesis was to propose an automatic approach to infer a structure of television talk show. There is no common understanding of what this imposed structure should look like and thus encompasses issues as diverse as scene segmentation, speech noise detection, speaker diarisation, etc. We have presented a very general approach to TV-show classification problems. Our approach is oriented towards a hierarchical structure of semantic level.

In this report we have presented the current research on video segmentation (chapter 1.2), a thorough background on multiple kernel learning support vector machines (chapter 2), our classification system (chapter 3), the three selected evaluation tasks and our empirical results (chapter 4). The multiple kernel learning (MKL) support vector machines classification system, as described in chapter 3, has been successfully implemented and is able to assess a wide selection of audio and video features. Despite the restriction to one database, we have created a generic approach, replicable and exportable to other collections of such TV shows.

## 5.1. Our chosen approach

We have chosen to apply multiple kernel support vector machines (SVM) to create classifiers for the different tasks. There are several properties that distinguishes SVM from other approaches to these tasks. Typical pattern recognition methods are based on the minimisation of the empirical risk, that is on the attempt to minimise the misclassification errors on the training set. Instead, SVM minimise the structural risk, that is the probability of misclassifying a previously unseen data point drawn randomly from a fixed but unknown probability distribution.

SVM will, moreover, condense all the information contained in the training set relevant to classification in the support vectors. This reduces the size of the training set identifying the most important points, and makes it possible to efficiently perform classification.

SVM is quite naturally designed to perform classification in high dimensional spaces, even in the presence of a relatively small number of data points. The real limitation to the employment of SVMs in high dimensional spaces is computational efficiency. In practice, for each particular problem a tradeoff between computational efficiency and success rate must be established.

Our implementation uses the SimpleMKL approach to multiple kernel learning. One advantage of the MKL approach is the possibility to combine and select the most relevant representations. The set of kernels can be composed of different kernels built from

different types of representations. So instead of using a single representation, one can fuse different representations through the kernel.

One drawback of kernel methods is the need of tuning efficiently the classifier and kernel parameters. Thanks to multiple kernel approaches however, it is possible to address this issue by proposing a set of kernels, each of them being computed with a different set of kernel parameters. Another advantage of multiple kernel is the possibility of selecting only a subset of features. This approach could be assimilated as feature selection, since we can build a set of kernels from each feature and then selecting only the most relevant one.

We have used multiple kernel learning support vector machines as (1) a feature selection algorithm and then (2) built various content detectors and classifiers useful for television data structuring and indexing. Three main groups of experiments have been assessed to tests our classification approach. The first, low-level experiment, was speaker identification, the second, midlevel classification task, was to detect broad semantic classes such as applause, speech, noise, music, etc., and the third, high-level classification task, was to classify the audio segments depending on whether it is recorded on stage, in a studio, outdoors, etc.

## 5.2. Achievements

A first goal of our work was to prove multiple kernel learning as a well performing feature selector, replacing the separate feature selection algorithms. We start our procedure by extracting a set of carefully chosen features, corresponding to frequently used features in state of the art systems for the specific task. A wider approach, where all available features where to be assessed, would have been preferred, but this task is too time consuming for the time available for our work. The selected features where, nevertheless, integrated and normalised before partitioned into training and validation subsets. We use the embedded approach of multiple kernel learning, as described in section 2.3.2. This approach prefers the features that has the greatest absolute distances in the feature space for data points with different labels.

In section 4.2.1 we use the feature subset multiple kernel learning approach to select a subset of most discriminating features. These empirical results give a first indication on the behaviour of multiple kernel learning as feature selection algorithm and has a very positive impact on the result for the specific test presented in this section.

Another goal of our work was to prove that multiple kernel learning support vector machine classifiers are well performing for a wide set of audiovisual problems, exceeding standard support vector machine classifiers. A serious drawback of kernel methods, and Support Vector Machines (SVM) in particular, is the difficulty in choosing a suitable kernel function for a given data-set. One of the approaches proposed to address this problem is Multiple Kernel Learning (MKL) in which several kernels are combined adaptively for a given data-set. In sections 4.2.1, 4.2.2 and 4.2.3 we have empirical tests indicating a well performing MKL classifiers for speaker identification, audio segmenting and scene detection, and the results in section 4.2.1 show how the results of the mul-

tiple kernel approach exceeds the single kernel approach. The need for a set of known data diverges from most state of the art speaker diarisation systems and may hence be seen as a major drawback of this approach. Our approach can nevertheless be used "semi-supervised" fashion (as described in detail in section 4.2.1) or as an enhancement technique to existing diarisation approaches.

The three main groups of experiments can easily be combined into one large classification system, able to do automatic structuring of television talk show. What we have shown in our work, however, is the increase in performance when using a multiple kernel approach as opposed to a simple kernel approach, moreover, the increase in performance when using feature subset adapted multiple kernel learning as opposed to standard multiple kernel learning.

## 5.3. Future work

As stated several times, inferring a structure of television talk shows is a yet rarely talked about by the scientific community. Our studies have been indicating the possibility of a well performing classification system based on multiple kernel learning support vector machines. Nevertheless, there are several aspects that should be explored further.

For improved performance in speaker identification as well as for audio segmentation and scene detection it should be advantageous to explore more of the features we have extracted. This is especially true in the aspect of MKL as feature selection algorithm, able to select a subset of features that we have seen improves the overall result. There is a very large number of possible feature combinations and in the optimal case each feature should be tested separately. Event though this functionality is implemented, the limited period of a masters thesis, combined with the time demanding training process when the number of features gets high, all available features have not been explored. Further experiments, including large scale feature selection, is suggested to improve the results in each group of experiments. These experiments could be done in a generic manner or adapted to a specific application. Moreover, large fold cross validation procedures should be imposed on the experiments to ensure the statistical significance.

To explore and improve the results of feature subset adapted multiple kernel learning further validation of kernels is needed. This process is time demanding, but could potentially, as we have shown, give a significant increase in performance. To execute such experiments it would be desirable to have a systematic scheme for tuning the parameters such as the spread of the Gaussian and the strength of the slack-variable.

# A. Features

We have extracted over 800 different audio and video features from the "Grand Echiquier" database. These include common features such as variations of cepstrum coefficients and image histograms, but also more "experimental" features such as acceleration of a detected face. The next two subsections will provide a deeper understanding of these features.

## A.1. Audio features

### Temporal features

The *zero crossing rate* [Maragos 08] descriptor is the number of times the time-domain signal passes the zero-level.

The *temporal moments* descriptor [Maragos 08] is a description of the amplitudes of the signal and aims to describe motion more closely than just pure statistical techniques.

The *auto correlation* descriptor [Maragos 08] is simply coefficients of the autocorrelation function for a frame of the audio signal. The discrete autocorrelation $R$ at lag $j$ for a discrete signal $x_n$ is given as

$$R_{xx}(j) = \sum_n x_n \overline{x}_{n-j} \tag{A.1.1}$$

The *linear predictive coding* [Maragos 08] descriptor is representing the spectral envelope of the signal of speech in compressed form, using the information of a linear predictive model. LPC analyses the speech signal by estimating the formants, removing their effects from the speech signal, and estimating the intensity and frequency of the residual signal.

### Spectral features

The *line spectral frequencies* descriptor [Chu 03] is a representation of the linear prediction coefficients (LPC). LSFs have several properties (e.g. smaller sensitivity to quantisation noise) that make them superior to direct quantisation of LPC.

The *spectrum flatness* descriptor [Chu 03] is a function which is related to the tonality aspect of the audio signal and can therefore be used as a discriminating criterion between different audio signals. It is calculated as the ratio of geometric mean to arithmetic mean of spectral power within a band. The spectrum flatness can be used to describe the signal in different frequency bands.

The *spectral crest factor* [Chu 03] is somewhat similar to the spectrum flatness, but instead of calculating the mean value for the numerator the maximum is computed, i.e. the ratio between the maximum spectral power within a frequency band and its mean power is determined. In the same way as for the spectrum flatness, a multi-band version can be calculated.

The *spectral rolloff* d[Chu 03] escriptor is the frequency under which a given percentage of the power distribution is concentrated.

The *spectral moments* [Chu 03] describe the distribution of frequencies in a spectrum.

The *spectral slope* descriptor [Chu 03] measures how quickly the spectrum tails off towards the high frequencies and is calculated using a linear regression. The slope (spectral gradient) is defined as:

$$S = \frac{R_{F_1} - R_{F_0}}{\lambda_1 - \lambda_0} \tag{A.1.2}$$

where $R_{F_0}$, $R_{F_1}$ is the reflectance measured with filters $F0$, $F1$ having the central wavelengths $\lambda 0$ and $\lambda 1$, respectively.

The *spectral variation* descriptor is defined in [MPEG-7 02] as the average over the sound duration of the one's complement of the normalised correlation between the amplitude (linear scale) of the harmonic peaks of the spectrum of two adjacent frames.

The *spectral decrease* descriptor [MPEG-7 02] measures the change in spectral amplitude. The formulation comes from psycho-acoustical studies.

$$SD = \frac{1}{\sum_{k=2}^{K} a} \sum_{k=2}^{K} \frac{a_k - a_1}{k-1} \tag{A.1.3}$$

The *Mel-Frequency Cepstrum* [Chu 03] is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a non-linear Mel scale of frequency. One could say that MFC give a description of the envelope of the spectrum. Mel-Frequency Cepstrum Coefficients are coefficients that collectively make up an MFC. The descriptor used are the DC values of the MFCC coefficients.

The $\Delta$ *MFCC* descriptor is calculated to catch the dynamic changes in the MFCC's. It is defined as:

$$\Delta c_i(n) = \frac{1}{2}(c_{i+1}(n) - c_{i-1}(n)) \tag{A.1.4}$$

The $\Delta\Delta$ *MFCC* descriptor is calculated to catch the dynamic changes in the $\Delta$MFCC's. It is defined as:

$$\Delta\Delta c_i(n) = \frac{1}{2}(\Delta c_{i+1}(n) - \Delta c_{i-1}(n)) \tag{A.1.5}$$

The *bandwidth 1* and *bandwidth 2* descriptors are estimated bandwidths in Hertz, extracted with Praat (see section D).

### Harmonic features

The *F0* descriptor is the fundamental frequency. In terms of a superposition of sinusoids (for example, Fourier series), the fundamental frequency is the lowest frequency sinusoidal in the sum.

The *entropy feature 1* and *entropy feature 2* descriptors are entropies calculated with Praat (see section D).

### Psycho-acoustic features

The *loudness* is a psycho-acoustic feature, that tries to simulate the perception of signal strength, extracted with Praat (see section D)

The *loudness spread* is a psycho-acoustic feature that tries to simulate the difference between the peak signal level and the noise floor or minimum signal level, the dynamic spread. It can be commuted as the average distance from the central per-frame level. It is extracted with Praat (see section D)

The *loudness sharpness* is a psycho-acoustic feature that tries to simulate the perception related to the spectral density and the relative strength of high-frequency energy. It is extracted with Praat (see section D)

## A.2. Video features

### Colour features

The *image histogram* descriptor [MPEG-7 02] is simply a type of histogram which acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value.

The *colour layout* descriptor [MPEG-7 02] is a very compact and resolution-invariant representation of colour designed to efficiently represent the spatial distribution of colours. The descriptor is obtained in four stages: partitioning, representative colour detection, DCT transform and non-linear quantisation of the zigzag-scanned coefficients. It is recommended to use a total of 12 coefficients, 6 for luminance and 3 for each chrominance for most images.

The *colour structure* descriptor [MPEG-7 02] represents an image by both the colour distribution of the image and the local spatial structure of the colour. The descriptor is obtained in three stages: HMMD (Hue, Max, Min, Diff) colour space histogram extraction, resampling to reach 256 bins and quantisation.

The *main dominant colour* descriptor [Vallet 10] is based on the association of the two main dominant colours i.e. the two colours with the highest fraction $P_i$ of pixels, corresponding to a given dominant colour $i$ in the image face and suit. In case only one dominant colour is detected, this one is duplicated.

The *mean dominant colour* descriptor [Vallet 10] is the average dominant colour in the face and suit which is a weighted average of the $n$ colours covering at least 40% of

the bounding box according to:

$$x_{C_{avg}} = \frac{\sum_{i=1}^{n} P_i x_{C_i}}{\sum_{i=1}^{n} P_i} \tag{A.2.1}$$

with $\sum_{i=1}^{n} P_i \geq 40\%$ and $C \in [R, G, B]$ is an RGB colour space component vector.

The *scalable colour* descriptor [MPEG-7 02] can be interpreted as a Haar transform-based encoding scheme applied across values of a colour histogram in the HSV colour space. The HSV colour space is a colour representation consisting of hue, surration and brightness. The histogram values are extracted, normalised, quantised (non linearly), transformed (Haar transform) and quantised (linearly). The output representation is scalable in terms of number of bins, by varying the number of coefficients used.

The *face colour layout* [Vallet 10] is based on the MPEG7 colour layout descriptor, but applied to the area recognised as the face.

The *face scalable colour* [Vallet 10] is based on the MPEG7 scalable colour descriptor, but applied to the area recognised as the face.

The *face main dominant colours* [Vallet 10] is based on the main dominant colour descriptor, but applied to the area recognised as the face.

The *face mean dominant colour over 40%* [Vallet 10] is based on the mean dominant colour descriptor, but applied to the area recognised as the face.

The *face mean dominant colour over 60%* [Vallet 10] is based on the mean dominant colour descriptor. As the name suggests, the descriptor is calculated as a weighted average of the $n$ colours covering at least 60% of the the area recognised as the face.

The *suit colour layout* [Vallet 10] is based on the MPEG7 colour layout descriptor, but applied to an area below the recognised face (intended to be the body of the speaker).

The *suit scalable colour*[Vallet 10] is based on the MPEG7 scalable colour descriptor, but applied to an area below the recognised face (intended to be the body of the speaker).

The *suit main dominant colours* [Vallet 10] is based on the main dominant colour descriptor, but applied to an area below the recognised face (intended to be the body of the speaker).

The *suit mean dominant colour over 40%* [Vallet 10] is based on the mean dominant colour descriptor, but applied to an area below the recognised face (intended to be the body of the speaker).

The *suit mean dominant colour over 60%* [Vallet 10] is based on the mean dominant colour descriptor. As the name suggests, the descriptor is calculated as a weighted average of the $n$ colours covering at least 60% of the the area below the recognised face (intended to be the body of the speaker).

### Object features

The *region shape* descriptor [MPEG-7 02] expresses the pixel distribution within a 2D object or region and describes properties of multiple disjoint regions simultaneously. It uses a complex 2D angular radial transformation defined on a unit disk in polar co-ordinates.

### Texture features

The *edge histogram* descriptor [MPEG-7 02] represents a local edge distribution in the image. To obtain the descriptor one divide the image space into 16 sub images. The local-edge distribution for each sub image can be represented as a histogram. To generate the histogram, edges in the sub-images are categorised into five types; vertical, horizontal, 45 degree diagonal, 135 degree diagonal and non directional edges. A total of 80 $(5 * 16)$ bins are included in the descriptor.

The *Gabor texture* descriptor [Clausi 00] is created with a Gabor filter, a linear filter used for edge detection. Gabor filters are a group of wavelets, with each wavelet capturing energy at a specific frequency and a specific direction. Expanding a signal using this basis provides a localised frequency description, therefore capturing local features of the signal. Texture features can then be extracted from this group of energy distributions.

The *Haralick texture* descriptor [Chetverikov 95] capture information about the patterns that emerge in patterns of texture. The descriptor is obtained by calculating the distribution of co-occurring values at a given offset (creating a co-occurrence matrix). Once the co-occurrence matrix has been constructed, calculations of 13 features begin. Some of these features include angular second moment, contrast, correlation, as well as a variety of entropy measures.

### Movement features

The *global movement intensity* [MPEG-7 02] is computed as the movement for points of interest in the images. This is expressed as an integer in the range $[1 - 5]$ where a high value indicates high activity while a low value indicates low activity.

The *global speed intensity* [MPEG-7 02] is computed as the first derivative of the movement for points of interest in the images.

The *global acceleration intensity* [MPEG-7 02] is computed as the second derivative of the movement for points of interest in the images.

The *face movement intensity* [Vallet 10] is computed as the movement for points of interest in the area detected as the face.

The *face movement orientation* [Vallet 10] is computed as the r and $\theta$ co-ordinates in a polar system for the dominant direction of the points of interest in the area detected as the face. It is expressed in [MPEG-7 02] by a three-bit integer that has a value corresponding to any of eight equally spaced directions.

The *face speed intensity* [Vallet 10] is computed as the first derivative of movement of points of interest in the area detected as the face.

The *face speed orientation* [Vallet 10] is computed as the r and $\theta$ co-ordinates in a polar system for the dominant direction of first derivative of the points of interest in the area detected as the face. It is expressed [MPEG-7 02] by a three-bit integer that has a value corresponding to any of eight equally spaced directions.

The *face acceleration intensity* [Vallet 10] is computed as the second derivative of movement of points of interest in the area detected as the face.

The *face acceleration orientation* [Vallet 10] is computed as the r and $\theta$ co-ordinates in a polar system for the dominant direction of the second derivative of the points of interest in the area detected as the face. It is expressed [MPEG-7 02] by a three-bit integer that has a value corresponding to any of eight equally spaced directions.

The *suit movement intensity* [Vallet 10] is computed as the movement for points of interest in the the area below the recognised face (intended to be the body of the speaker).

The *suit movement orientation* [Vallet 10] is computed as the r and $\theta$ co ordinates in a polar system for the dominant direction of the points of interest in the area below the recognised face (intended to be the body of the speaker).

The *suit speed intensity* [Vallet 10] is computed as the first derivative of movement of points of interest in the area below the recognised face (intended to be the body of the speaker).

The *suit speed orientation* [Vallet 10] is computed as the r and $\theta$ co-ordinates in a polar system for the movement of first derivative of the points of interest in the area below the recognised face (intended to be the body of the speaker). It is expressed [MPEG-7 02] by a three-bit integer that has a value corresponding to any of eight equally spaced directions.

The *suit acceleration intensity* [Vallet 10] is computed as the second derivative of movement of points of interest in the area below the recognised face (intended to be the body of the speaker).

The *suit acceleration orientation* [Vallet 10] is computed as the r and $\theta$ co-ordinates in a polar system for the second derivative of the points of interest in the area below the recognised face (intended to be the body of the speaker). It is expressed [MPEG-7 02] by a three-bit integer that has a value corresponding to any of eight equally spaced directions.

The *head minus body movement intensity* [Vallet 10] is computed as the movement of the points of interest in the recognised face minus the movement of the points of interest the area below (intended to be the body of the speaker).

The *head to body movement intensity* [Vallet 10] is computed as the movement between the points of interest in the recognised face and the area below (intended to be the body of the speaker).

The *head minus body speed intensity* [Vallet 10] is computed as the first derivative of movement of the points of interest in the recognised face minus the movement of the points of interest the area below (intended to be the body of the speaker).

The *head to body speed intensity* [Vallet 10] is computed as the first derivative of movement between the points of interest in the recognised face and the area below (intended to be the body of the speaker).

The *head minus body acceleration intensity* [Vallet 10] is computed as the second derivative of movement of the points of interest in the recognised face minus the movement of the points of interest the area below (intended to be the body of the speaker).

The *head to body acceleration intensity* [Vallet 10] is computed as the second derivative of movement between the points of interest in the recognised face and the area below (intended to be the body of the speaker).

# B. Memory handling

When handling large data-sets you need to be careful when choosing data architecture and strategies. Even though several multiple kernel toolboxes are freely available on the internet, none of them seems to combine the functionality needed in our experiments with graceful memory handling. See appendix D for further details.

Most computer platforms today are 32-bit architectures, meaning that the length of a pointer or the size of a processor instruction can be at most 32 bits long. This limitation on the sizes of pointers implies that the memory addresses can be a maximum of 32 bits long, which results in a maximum of $2^{32}$ possible memory addresses. Since modern operating systems are byte addressable, this translates to 4 gigabytes (GB) of addressable memory. Therefore, under 32 bit architectures, the seemingly unlimited virtual memory space is actually limited to approximately 4 GB.

Several 32-bit UNIX operating systems also reserve the upper 2 GB of address space, thereby limiting the virtual memory available to 2 GB. When you launch MATLAB, approximately 1 GB of virtual address space is used to load the heap, stack, DLLs, and other operating system services. This leaves only 1 GB.

In August 2010, TELECOM ParisTech bought licences for 64-bit matlab, to take full advantage of the 64-bit machines. We decided to move the classification system to these machines. In addition to that we have optimised the memory handling in several ways. First, by writing all variables that are unnecessary for the current function to disc. A simple Matlab code, wrapping all major functions, is shown below.

```
save allVar
save mklsvmlsVar K training_lab_current C options verbose
clear all
load mklsvmlsVar
[beta,w,b,posw,story,obj] = mklsvmls(K,training_lab_current, ...
    C,options, verbose);
save mklsvmlsOut beta w b posw story obj
clear all
load allVar
load mklsvmlsOut
```

Nevertheless, further optimisation was needed. We modified the SimpleMKL toolbox in several places. For instance, the train data distance computations in equation 2.2.22 (see theory section 2.2.3) was cut and done in batches.

# C. Theorems

## Fourier

The Fourier transform is an operation that transforms one complex-valued function of a real variable into another. While there are several conventions for defining the Fourier transform of an integrable function, a common definition is:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\ e^{-2\pi i x \xi}\,dx \tag{C.0.1}$$

for every real number $\xi$.

When the independent variable $x$ represents time, the transform variable $\xi$ represents frequency (in hertz). Under suitable conditions, $f$ can be reconstructed from $\hat{f}$ by the inverse transform:

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\ e^{2\pi i x \xi}\,d\xi \tag{C.0.2}$$

for every real number $x$.

## Mercer

Mercer's theorem is a representation of a symmetric positive-definite function on a square as a sum of a convergent sequence of product functions. It is an important theoretical tool in the theory of integral equations; it is used in the Hilbert space theory of stochastic processes, for example the Karhunen-Loève theorem; and it is also used to characterise a symmetric positive semi-definite kernel. If K is a symmetric non-negative definite kernel on $(X, M, \lambda)$, then

$$K(y, x) = \sum_{i \in \mathbb{N}} \lambda_i e_i(y) e_i(x) \tag{C.0.3}$$

where the convergence in the $L2$ norm. Note that when continuity of the kernel is not assumed, the expansion no longer converges uniformly.

## Hilbert

The mathematical concept of a Hilbert space generalises the notion of Euclidean space. It extends the methods of vector algebra and calculus from the two-dimensional Euclidean plane and three-dimensional space to spaces with any finite or infinite number of dimensions. A Hilbert space, $H$, is a real or complex inner product space that is also a complete metric space with respect to the distance function induced by the inner

product. To say that $H$ is a complex inner product space means that $H$ is a complex vector space on which there is an inner product $\langle x, y\rangle$ associating a complex number to each pair of elements $x, y$ of $H$, that satisfies three properties. First that $\langle y, x\rangle$ is the complex conjugate of $\langle x, y\rangle$:

$$\langle y, x\rangle = \overline{\langle x, y\rangle}. \tag{C.0.4}$$

Secondly $\langle x, y\rangle$ is linear in its first argument. For all complex numbers $a$ and $b$

$$\langle ax_1 + bx_2, y\rangle = a\langle x_1, y\rangle + b\langle x_2, y\rangle. \tag{C.0.5}$$

And finally that the inner product is positive definite:

$$\langle x, x\rangle \geq 0 \tag{C.0.6}$$

where the case of equality holds precisely when $x = 0$.

## Lagrange

The method provides a strategy for finding the maximum/minimum of a function subject to constraints. If you want to minimise function $f(x, y)$ with constraint $g(x, y) = c$ you can imagine "walking along" the contour line with $g = c$. You may intersect or cross $f(x, y)$ several times, but you are interested in the point(s) where $g = c$ meets contour lines of $f(x, y)$ tangentially. Because the gradient of a function is perpendicular to the contour lines, another way of representing this is to say that we want points $(x, y)$ where $g(x, y) = c$ and

$$\nabla_{x,y} f = -\lambda \nabla_{x,y} g \tag{C.0.7}$$

where $\nabla_{x,y} f$ and $\nabla_{x,y} g$ are the respective gradients. $\lambda$ is needed because the magnitudes of the gradient vectors can be unequal. This can be incorporated into an auxiliary function

$$\Lambda(x, y, \lambda) = f(x, y) + \lambda * (g(x, y) - c) \tag{C.0.8}$$

where the solution is given as

$$\nabla_{x,y,\lambda} \Lambda(x, y, \lambda) = 0 \tag{C.0.9}$$

## Karush-Kuhn-Tucker

Consider the following nonlinear optimization problem:

$$\text{Minimize } f(x) \text{ subject to: } g_i(x) \leq 0, h_j(x) = 0 \tag{C.0.10}$$

where $f(\cdot)$ is the function to be minimized, where $g_i(\cdot)$ $(i = 1, \ldots, m)$ are the functions of the inequality constraints and $h_j(\cdot)$ $(j = 1, \ldots, l)$ are the functions of the equality constraints, and where $m$ and $l$ are the number of inequality and equality constraints,

respectively. Allowing inequality constraints, the KKT approach to nonlinear programming generalizes the method of Lagrange multipliers, which have allowed only equality constraints.

Suppose that the function to be minimized, is $f : \mathbb{R}^n \to \mathbb{R}$ and the constraint functions are $g_i : \mathbb{R}^n \to \mathbb{R}$ and $h_j : \mathbb{R}^n \to \mathbb{R}$. Further, suppose they are continuously differentiable at a point $x^*$. If $x^*$ is a local minimum that satisfies some regularity conditions, then there exist constants $\mu_i$ $(i = 1, \ldots, m)$ and $\lambda_j$ $(j = 1, \ldots, l)$ such that Stationarity

$$\nabla f(x^*) + \sum_{i=1}^{m} \mu_i \nabla g_i(x^*) + \sum_{j=1}^{l} \lambda_j \nabla h_j(x^*) = 0, \tag{C.0.11}$$

Primal feasibility

$$g_i(x^*) \leq 0, \text{ for all } i = 1, \ldots, mh_j(x^*) = 0, \text{ for all } j = 1, \ldots, l \tag{C.0.12}$$

Dual feasibility

$$\mu_i \geq 0, \text{ for all } i = 1, \ldots, m \tag{C.0.13}$$

Complementary slackness

$$\mu_i g_i(x^*) = 0, \text{for all } i = 1, \ldots, m. \tag{C.0.14}$$

## Wiener-Khinchin

The Wiener-Khinchin theorem states that the power spectral density of a wide-sense-stationary random process is the Fourier transform of the corresponding autocorrelation function. In the discrete case that means that:

$$S_{xx}(f) = \sum_{k=-\infty}^{\infty} r_{xx}[k] e^{-j2\pi kf} \tag{C.0.15}$$

where $r_{xx}[k] = \mathrm{E}\left[\, x[n]x^*[n-k] \,\right]$ and where $S_{xx}(f)$ is the power spectral density of the function with discrete values $x[n]$. Being a sampled and discrete-time sequence, the spectral density is periodic in the frequency domain.

# D. Softwares and languages

The main software used in this project was Matlab version 7.4.0.336 (R2007a) 32-bit and Matlab version 7.10.0.449 (R2010a) 64-bit. In order to verify and compare our results, we have assessed three different SVM libraries: LIBSVM[1], SimpleMKL[2] and SHOGUN[3]. LIBSVM is a standard support vector machine library [Chang 01] that does one kernel SVM classification only [Hsu 09], while SHOGUN [S.Sonnenburg 06] and SimpleMKL are multiple kernel learning toolboxes [Rakotomamonjy 08]. The SHOGUN toolbox provides implementations of several state of the art SVM algorithms, such as [Franc 08], [Fan 08], [Chang 01] and [Joachims 99] and can be used as a framework to do multiple kernel learning based on them. Moreover, it has implementations of common feature selection algorithms. The advantage of this toolbox is that it is written in Python and hereby handles large data sets more gracefully than Matlab.

The Praat program[4] [Weenink 01] is a phonetics tool that is commonly used for research purposes. Praat version 5.1.25 along with a variety of Praat scripts was proven useful in the feature extraction.

Yaafe[5] is a state of the art feature extraction software developed at TELECOM ParisTech that automatically identifies common intermediate representations such as spectrum, envelope, autocorrelation, etc., and computes them only once. Extraction is processed block per block so that arbitrarily long files can be processed, and memory occupation is low.

Python and Perl were used as file handeling and protocol adjustment languages. The C-code was compiled under Linux Debian and executed at 64-bit machines 'Claude', 'Caligula' and 'Trajan', provided at AAO TSI TELECOM ParisTech.

[1]LIBSVM: http://www.csie.ntu.edu.tw/c̃jlin/libsvm/
[2]SimpleSVM: http://asi.insa-rouen.fr/enseignants/ arakotom/code/mklindex.html
[3]SHOGUN: http://www.shogun-toolbox.org/
[4]Praat: http://www.praat.org/
[5]Yaafe: http://yaafe.sourceforge.net/

# Index

# List of Figures

# List of Tables

# Bibliography


[Abbasnejad 10] M. Ehsan Abbasnejad, Dhanesh Ramachandram & Rajeswari Mandava. *An unsupervised approach to learn the kernel functions: from global influence to local similarity.* Neural Comput & Applic, 2010.

[Achlioptas 02] D. Achlioptas, F. McSherry & B. Scholkopf. *Sampling techniques for kernel methods.* Advances in Neural Information Processing Systems, vol. 14, 2002.

[Aflalo 10] J. Aflalo, A. Ben-Tal, C. Bhattacharyya, J. Saketha Nath & S. Raman. *Variable Sparsity Kernel Learning.* Journal of Machine Learning Research (submitted), 2010.

[Ajmera 03] Jitendra Ajmera, Iain McCowan & Herve Bourlard. *Speech/music segmentation using entropy and dynamism features in a HMM classification framework.* Speech Communication, vol. 40, 2003.

[Assfalg 02] Jurgen Assfalg, Alberto del Bimbo, Walter Nunziati & Pietro Pala. *Soccer highlights detection and recognition using HMMs.* International Conference on Multimedia and Expo, 2002.

[Athanasiadis 09] Thanos Athanasiadis, Nikolaos Simou, Georgios Papadopoulos, Rachid Benmokhtar, Krishna Chandramouli, Vassilis Tzouvaras, Vasileios Mezaris, Marios Phiniketos, Yannis Avrithis, Yannis Kompatsiaris, Benoit Huet & Ebroul Izquierdo. *Integrating image segmentation and classification for fuzzy knowledge-based multimedia indexing.* International MultiMedia Modeling Conference, 2009.

[Bach 04] F. Bach, G. Lanckriet & M. Jordan. *Multiple kernel learning, conic duality and the SMO algorithm.* Proceedings of the 21st International Conference on Machine Learning, pages 41–48, 2004.

[Baillie 04] Mark Baillie & Joemon M. Jose. *An audio-based sports video segmentation and event detection algorithm.* Conference on Computer Vision and Pattern Recognition Workshop, 2004.

[Barrington 08] Luke Barrington, Mehrdad Yazdani, Douglas Turnbull & Gert Lanckriet. *Combining feature kernels for semantic music retreival,.* ISMIR, 2008.

[Benmokhtar 07] Rachid Benmokhtar, Eric Galmar & Benoit Huet. *Eurecom at trecvid 2007: Extraction of high level features.* In International Conference on Multimedia and Expo, 2007.

[Benmokhtar 09] Rachid Benmokhtar & Benoit Huet. *Hierarchical ontology-based robust video shots indexing using global mpeg-7 visual descriptors.* International Workshop on Content-Based Multimedia Indexing, 2009.

[Boujema 04] Nozha Boujema, Francois Fleuret, Valerie Gouet & Hichem Sahbi. *Automatic textual annotation of video news based on semantic visual object extraction.* In Conference on Storage and Retrieval Methods and Applications for Multimedia, 2004.

[Bozonnet 10] Simon Bozonnet, Felicien Vallet, Nicholas Evans, Slim Essid, Gael Richard & Jean Carrive. *A multimodal approach to initialization for top-down speaker diarization of television shows.* submitted to EUSIPCO, 2010.

[Brandenburg 97] Karlheinz Brandenburg & Marina Bosi. *Overview of MPEG Audio: Current and Future Standards for Low Bit-Rate Audio Coding.* The Journal of the Audio Engineering Society, 1997.

[Chang 01] Chih-Chung Chang & Chih-Jen Lin. *LIBSVM: a library for support vector machines.* http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[Chen 03] Shu Ching Chen, Mei Ling Shyu, Chengcui Zhang, Lin Luo & Min Chen. *Detection of soccer goal shots using joint multimedia features and classification rules.* International Workshop on Multimedia Data Mining, 2003.

[Cheng 92] Yong-Qing Cheng, Yong-Ming Zhuang & Jling-Yu Yang. *Optimal fisher discriminant analysis using the rank decomposition.* Pattern Recognition Society, vol. 25, pages 101–111, 1992.

[Chetverikov 95] D. Chetverikov & R.M. Haralick. *Texture Anisotropy, Symmetry, Regularity: Recovering Structure from Interaction Maps.* Proceedings of British Machine Vision Conference, 1995.

[Chu 03] Wai C. Chu. Speech coding algorithms: foundation and evolution of standardized coders. Wiley-Interscience, 2003.

[Clausi 00] D. A. Clausi & M. E. Jernigan. *Designing Gabor filters for optimal texture separability.* Pattern Recognition, vol. 33, 2000.

[Cristianini 00] Nello Cristianini & John Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, 2000.

[Dileep 09] A. D. Dileep & C. Chandra Sekhar. *Representation and Feature Selection using Multiple Kernel Learning*. International Joint Conference on Neural Networks, 2009.

[Duda 73] Richard Duda & Peter Hart. Pattern classification. Wiley-Interscience, 1973.

[Dumont 08] Emilie Dumont & Bernard Merialdo. *Sequence alignment for redundancy removal in videorushes summarization.* International Conference on Multimedia Information Retrieval, 2008.

[Essid 05] Slim Essid. *Classification automatique des signaux audio-frequences: reconnaisanse des instruments de musique*. PhD thesis, Universite Pierre & Marie Curie, 2005.

[Fan 08] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang & C.-J. Lin. *LIBLINEAR: A library for large linear classification*. Journal of Machine Learning Research, vol. 9, 2008.

[Fine 01] S. Fine & K. Scheinberg. *Efficient SVM training using low-rank kernel representations*. Journal of Machine Learning Research, vol. 2, pages 243–264, 2001.

[Franc 08] V. Franc & S. Sonnenburg. *OCAS optimized cutting plane algorithm for Support Vector Machines*. Proceedings of ICML, 2008.

[Fredouille 09] Corinne Fredouille, Simon Bozonnet & Nicholas Evans. *The LIA-EURECOM RT'09 Speaker Diarization System*. NIST Rich Transcription, 2009.

[Friedland 09] Gerald Friedland, Hayley Hung & Chuohao Yeo. *Multi-modal speaker diarization of real-world meeting using compressed domain video features*. ICASSP, 2009.

[Guyon 03] Isabelle Guyon & Andre Elisseeff. *An Introduction to Variable and Feature Selection*. Journal of Machine Learning Research, vol. 3, 2003.

[Hanjalic 99a] Alan Hanjalic, Reginald Lagendijk & Jan Biemond. *Automated high-level movie segmentation for advanced video-retrieval systems*. IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, pages 580–588, 1999.

[Hanjalic 99b] Alan Hanjalic, Reginald L. Lagendijk & Jan Biemond. *An integrated scheme for automated video abstraction based on unsupervised cluster-validity analy sis.* IEEE Transactions on Circuits and Systems for Video Technology, 1999.

[Hsu 02] Chih-Wei Hsu & Chih-Jen Lin. *A Comparison of Methods for Multiclass Support Vector Machines.* IEEE Transactions on neural networks, vol. 13, 2002.

[Hsu 09] Chih-Wei Hsu, Chih-Chung Chang & Chih-Jen Lin. *A Practical Guide to Support Vector Classifcation.* Rapport technique, National Taiwan University, 2009.

[Jaffré 04] Gael Jaffré. *Le costume: une nouvelle caracteristique pour l'indexation de contenus vidéo.* In colloque de l'EDIT, 2004.

[Joachims 99] T. Joachims. *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning, B. Scholkopf and C. Burges and A. Smola, MIT-Press, 1999.

[Joder 09] Cyril Joder, Slim Essid & Gael Richard. *Temporal integration for audio classification with application to musical instrument classification.* IEEE Transactions on audio, speech and language processing, vol. 17, 2009.

[Kim 04] In-Jung Kim. *Multi-Window Binarization of Camera Image for Document Recognition.* Intel Workshop on Frontiers in Handwriting Recognition, vol. 9, 2004.

[Kloft 09] Marius Kloft, Ulf Brefeld, Soren Sonnenburg, Pavel Laskov, Klaus-Robert Muller & Alexander Zien. *Efficient and Accurate lp-Norm Multiple Kernel Learning.* Advances in Neural Information Processing Systems, vol. 22, 2009.

[Kloft 10] Marius Kloft, Ulrich Ruckert & Peter Bartlett. *A Unifying View of Multiple Kernel Learning.* Rapport technique, University of California at Berkeley, Technical Report No. UCB/EECS-2010-49, 2010.

[Kohavi 98] R. Kohavi & F. Provost. *Glossary of terms.* Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, 1998.

[Kotsiantis 06] S. B. Kotsiantis, D. Kanellopoulos & P. E. Pintelas. *Data Preprocessing for Supervised LeaningData Preprocessing for Supervised Leaning.* International Journal of Computer Science, 2006.

[Lanckriet 04] G. Lanckriet, N. Christianini, L. EL Ghaouri, P. Bartlett & M. Jordan. *Learning the kernel matrix with semi-definate programming.* Journal of Machine Learning Research, 2004.

[Leens 09] Jerome Leens, Sebastien Pierard, Olivier Barnich, Marc Van Droogenbroeck & Jean-Marc Wagner. *Combining Color, Depth, and Motion for Video Segmentation.* International Conference on Computer Vision Systems, 2009.

[Liu 98] Huan Liu & Hiroshi Motoda. Feature selection. Kluwer academic publisher, 1998.

[Liu 03] Tianming Liu, Hong-Jiang Zhang & Feihu Qi. *A novel video key-frame extraction algorithm based on perceived motion energy model.* IEEE Transactions on Circuits and Systems for Video Technology, 2003.

[Loosli 05] G. Loosli, S. Canu, S. Vishwanathan, A. Smola & M. Chattopadhyay. *Boite a outils SVM simple et rapide.* Revenue d'Intelligence Artificielle, pages 741–767, 2005.

[Lucas 81] Bruce Lucas & Takeo Kanade. *An iterative image registration technique with an application to stereo vision.* International Joint Conference on Artificial Intelligence, 1981.

[Luo 08] Hangzai Luo, Yuli Gao, Xiangyang Xue, Jinye Peng & Jianping Fan. *Incorporating feature hierarchy and boosting to achieve more effective classifier training and concept-oriented video summarization and skimming.* ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 4, pages 1–25, 2008.

[Maragos 08] Petros Maragos. Multimodal processing and interaction: audio, video, text. Numeéro 0387763155. Springer Science+Buisness Media, 2008.

[Michie 94] D. Michie & D.J. Spiegelhalter. Machine learning, neural and statistical classification. Ellis Horwood, 1994.

[MPEG-7 02] MPEG-7. Introduction to mpeg-7. Wiley, 2002.

[Naturel 05] Xavier Naturel, Guillaume Gravier & Patrick Gros. *Etiquetage automatique de programmes de television.* Compression et Representation des Signaux Audiovisuels, 2005.

[Noy 01] Natalya F. Noy & Deborah L. McGuinness. *A guide to creating your first ontology.* Rapport technique, Stanford University, 2001.

[Platt 99] John C. Platt. *Probabilstic outputs for support vector machines and comparisons to regularized likelihod methods.* Advances in large margin classifiers, 1999.

[Pontil 98] Massimiliano Pontil & Alessandro Verri. *Support Vector Machines for 3D Object Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, 1998.

[Rakotomamonjy 08] Alain Rakotomamonjy, Francis R. Bach, Stephane Canu & Yves Grandvalet. *SimpleMKL.* Journal of Machine Learning Research, vol. 1-34, 2008.

[Ramona 10] Mathieu Ramona. *Classification automatique de flux radiophoniques par Machines a Vecteurs de Support.* PhD thesis, TELECOM ParisTech, 2010.

[Ripley 96] B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[Rubner 00] Yossi Rubner, Carlo Tomasi, & Leonidas J. Guibas. *The Earth Mover's Distance as a Metric for Image Retrieval.* International Journal of Computer Vision, vol. 40, 2000.

[Smeaton 09] Alan Smeaton, Paul Over & Aiden Dohery. *Video shot boundary detection.* Computer Vision and Image Understanding, pages 1–25, 2009.

[Smola 04] A. Smola & B. Scholkopf. *A tutorial on support vector regression.* Statistics and Computing, vol. 14, 2004.

[S.Sonnenburg 06] S.Sonnenburg, G.Raetsch, C.Schaefer & B.Schoelkopf. *Large Scale Multiple Kernel Learning.* Journal of Machine Learning Research, vol. 7, pages 1531–1565, 2006.

[Struc 09] V. Struc & N. Pavesic. *A comparison of feature normalization techniques for PCA-based palmprint recognition.* Proceedings of the international conference MATHMOD, 2009.

[Theodoridis 99] Sergios Theodoridis & Konstantionos Koutroumbas. Pattern recognition. Academic Press, 1999.

[Tomioka 10] Ryota Tomioka & Taiji Suzuki. *Sparsity-accuracy trade-off in MKL.* ARXIV, 2010.

[Tranter 06] S.E. Tranter & D.A. Reynolds. *An overview of automatic speaker diarization systems.* IEEE Transactions on Audio, Speech, and Language Processing„ vol. 14, no. 5, 2006.

[Trecvid 10] Trecvid. *Trec video retrieval evaluation.* http ://www-nlpir.nist.gov/projects/trecvid/, 2010.

[Tuia 09] Devis Tuia, Giona Matasci, Gustavo Champs-Valls & Mikhail Kanevski. *Learning the relevent image features with multiple kernels.* IEEE International Geoscience and Remote Sensing Symposium, 2009.

[Vajaria 06] H. Vajaria, T. Islam, S. Sarkar, R. Sankar & R. Kasturi. *Audio Segmentation and Speaker Localization in Meeting Videos.* International Conference on Pattern Recognition, 2006.

[Vallet 10] Felicien Vallet, Slim Essid, Jean Carrive & Gael Richard. *Robust visual features for the multimodal identification of unregistered speakers in TV talk-shows.* submitted to ICIP, 2010.

[Viola 01] Paul Viola & Michael Jones. *Robust real time object detection.* International Workshop on Statistical and Comutational Theories of Vision Modeling, Learning, Computing and Sampling, 2001.

[Vishwanathan 03] S. V. N. Vishwanathan, A. J. Smola & M. Murty. *SimpleSVM.* International Conference on Machine Learning, 2003.

[Wang 09] Feng Wang & Bernard Merialdo. *Multi-document video summarization.* International Conference on Multimedia & Expo, 2009.

[Weenink 01] Paul Boersma & David Weenink. *Praat, a system for doing phonetics by computer.* Glot International, vol. 5, pages 341–345, 2001.

[Williams 01] C. K. I. Williams & M. Seeger. *Using the Nystrom method to speed up kernel machines.* Advances in Neural Information Processing Systems, 2001.

[Xiong 03] Ziyou Xiong, Regunathan Radhakrishnan & Ajay Divakaran. *Generation of sports highlights using motion activities in combination with a common audio feature extraction framework.* International Conference on Image Processing, 2003.

[Xu 08] Dong Xu & Shih-Fu Chang. *Visual Event Recognition in News Video using Kernel Methods with Multi-Level Temporal Alignment.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, 2008.

[Yeung 96] Minerva Yeung & Boon-Lock Yeo. *Time-constrained clustering for segmentation of video into story units.* International Conference on Pattern Recognition, 1996.

[Zamolotskikh 07] Anton Zamolotskikh & Padraig Cunningham. *An Assessment of Alternative Strategies for Constructing EMD-Based Kernel Functions for Use in an SVM for Image Classification.* Rapport technique, Trinity College Dublin and University College Dublin, 2007.

[Zhang 06] J. Zhang, M. Marsza lek, S. Lazebnik & C. Schmid. *Local features and kernels for classification of texture and object categories: A comprehensive study.* International Journal of Computer Vision, 2006.

[Zhang 09] Jessie Xin Zhang, Jacqueline Whalley & Stephen Brooks. *A two phase method for general audio segmentation.* International Conference on Multimedia and Expo, 2009.

[Zhou 02] Junyu Zhou & Wallapak Tavanapong. *Shotweave : A shot clustering technique for story browsing for large video databases.* Lecture Notes In Computer Science, 2002.

[Zien 07] A. Zien & C.S. Ong. *Multiclass multiple kernel learning.* Proceedings of the 24th International Conference on Machine Learning, pages 1191–1198, 2007.

IN COOPERATION WITH NTNU AND TELECOM PARISTECH

# EURECOM PRESENTS

# AUDIOVISUAL CONTENT SEGMENTATION

## MASTER THESIS DEFENSE

## SEPTEMBER 21ST AT 10 AM EURECOM EC03

**MASTER CANDIDATE: KRISTOFFER SUNDØY**

The objective of the thesis is to detect high level semantic ideas to help to impose a DVD-like structure on television talk shows. Structuring of TV-shows is a subject that is rarely talked about in the scientific community. We propose a generic approach based on state of the art multiple kernel support vector machines that is able to assess all a large variety of common structuring problems. We have assessed a wide selection of audio and video features, used MKL as a feature selection algorithm and then built various content classifiers useful for imposing broad semantic classes.

**SUPERVISORS: FELICIEN VALLET, SLIM ESSID, NICKOLAS EVANS, MAGNE HALLSTEIN, TORBJØRN SVENDSEN**