TAMPERE UNIVERSITY OF TECHNOLOGY

ALEKSANDR DIMENT

SEMI-SUPERVISED MUSICAL INSTRUMENT RECOGNITION

Master's Thesis

Examiners: Adj.Prof. Tuomas Virtanen
M.Sc. Toni Heittola
Examiners and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 7.11.2012

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Programme in Information Technology
DIMENT, ALEKSANDR: Semi-Supervised Musical Instrument Recognition
Master of Science Thesis, 71 pages
June 2013
Major subject: Signal processing
Examiners: Adj.Prof. Tuomas Virtanen, M.Sc. Toni Heittola
Keywords: music information retrieval, musical instrument recognition, pattern recognition, semi-supervised learning

The application areas of music information retrieval have been gaining popularity over the last decades. Musical instrument recognition is an example of a specific research topic in the field. In this thesis, semi-supervised learning techniques are explored in the context of musical instrument recognition. The conventional approaches employed for musical instrument recognition rely on annotated data, i.e., example recordings of the target instruments with associated information about the target labels in order to perform training. This implies a highly laborious and tedious work of manually annotating the collected training data. The semi-supervised methods enable incorporating additional unannotated data into training. Such data consists of merely the recordings of the instruments and is therefore significantly easier to acquire. Hence, these methods allow keeping the overall development cost at the same level while notably improving the performance of a system.

The implemented musical instrument recognition system utilises the mixture model semi-supervised learning scheme in the form of two EM-based algorithms. Furthermore, upgraded versions, namely, the additional labelled data weighting and class-wise retraining, for the improved performance and convergence criteria in terms of the particular classification scenario are proposed. The evaluation is performed on sets consisting of four and ten instruments and yields the overall average recognition accuracy rates of 95.3 and 68.4%, respectively. These correspond to the absolute gains of 6.1 and 9.7% compared to the initial, purely supervised cases. Additional experiments are conducted in terms of the effects of the proposed modifications, as well as the investigation of the optimal relative labelled dataset size. In general, the obtained performance improvement is quite noteworthy, and future research directions suggest to subsequently investigate the behaviour of the implemented algorithms along with the proposed and further extended approaches.

PREFACE

This work has been conducted at the Department of Signal Processing of Tampere University of Technology.

I would like to primarily express my gratitude to the supervisors of the thesis Tuomas Virtanen and Toni Heittola for this opportunity. Their assuring disposition, enormous responsiveness and perpetual eagerness to help uphold the workflow by providing invaluable guidance, advice and suggestions for research directions are eminently appreciated.

Additionally, my thankfulness extends to the members of the Audio Research Team for their positive and cordial attitude. This made the working process flowing and highly satisfying. I would also like to express appreciation to Antti Eronen for the Audio Research Team HMM Toolbox (AHTO), which has been utilised during the implementation stage of this work. The funding provided by the Department of Signal Processing for conducting the work is appreciatively acknowledged.

Finally, I wish to say thank you to my mother and whole family. Their enormous understanding and spiritual support have been incredibly encouraging.

Aleksandr Diment
Tampere, May 2013

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

$\boldsymbol{\Sigma}_k^{(t+1)}$	covariance matrix of the kth mixture component at iteration $t+1$, page 32
$\boldsymbol{\Sigma}_y$	covariance matrix, page 21
$\boldsymbol{\mu}_k^{(t+1)}$	mean vector of the kth mixture component at iteration $t+1$, page 32
$\boldsymbol{\mu}_y$	mean vector, page 21
γ_{ik}	estimate of the probability that the ith training sample originates from the kth mixture component, page 31
$\hat{\boldsymbol{\theta}}$	model parameters estimate, page 31
$\mathbf{x}$	feature vector, page 4
$\mathcal{D}$	training data, page 22
$\mathcal{H}$	hidden data, page 23
$\omega(t)$	decreasing weighting function of iteration index t, page 39
c_j	a label associated with class of index j, page 33
D	dimension of a feature vector, page 4
$E(f)$	energy of the spectral component at frequency f, page 16
f	classifier, page 24
K	number of mixture components in a model, page 31
k	audio sample index, page 15
k	mixture component index, page 31
L	number of labelled samples, page 19
M	total number of classes, page 9
N	number of samples within a frame, page 15
n_k	total weight of the kth Gaussian across all samples, page 31
n_y	the number of labelled training data instances originating from the class y, page 21
s	values of a sampled audio signal within a frame, page 15
U	number of unlabelled samples, page 19
$w_k^{(t+1)}$	weight of the kth mixture component at iteration $t+1$, page 31
y_i	a label associated with a feature vector $\mathbf{x}_i$, page 19
Y_k	mel-spaced filterbank, page 17
z_{ij}	a hidden variable containing information about labelling of the data instances, page 33

DCT	discrete cosine transform, page 19
EM	expectation-maximisation, page 23
GMM	Gaussian mixture model, page 21
HMM	hidden Markov model, page 21
LCR	label change rate, page 37

MFCCs	mel-frequency cepstral coefficients, page 13
MLE	maximum likelihood estimate, page 22
NMF	non-negative matrix factorisation, page 14
SSL	semi-supervised learning, page 2
SVMs	support vector machines, page 25
TSVMs	transductive support vector machines, page 25

1. INTRODUCTION

Musical instrument recognition belongs to the the subject of music information retrieval. The latter, broadly speaking, refers to obtaining information of various kinds from music. It is a rather wide research area, which includes numerous topics; to mention a few: score following, query by singing/humming, melody extraction, chord estimation and beat tracking. This research area has been recently gaining interest among businesses and within academia. Its applications introduce such previously unknown concepts as situationally tailored playlisting, personalised radio, social music applications and so forth.

Examples of such applications include Musipedia [31], which enables retrieval of a song by whistling or playing the melody or tapping its rhythm. Another example is Shazam [56] — a fingerprint-based retrieval system (i.e., a system that utilises a compact identifying summary inferred from an arbitrarily large signal) tailored for a use on mobile phones, robust to noise and distortions caused by such conditions. These and many other systems employ various recent advancements made in the area of musical information retrieval, and the interest of the audience, which they have attracted, indicates the high potential of this topic.

1.1 Musical instrument recognition

One of the areas of application related to musical information retrieval and content analysis is automatic musical instrument recognition, which has attracted a growing research interest over the past two decades. It has made possible the development of various applications. These include, for example, automatic music database annotation for indexing and retrieval purposes. By introducing such content-dependent fields as the instruments present in the recordings to the structure of a database, this facilitates completely new ways of searching for material, thus providing a totally different user experience and increasing the acceptability of the service.

Another example is automatic music transcription applications, which could benefit from identifying the instruments present in the recording. Furthermore, one may think in broader terms and realise applicability of musical instrument recognition for other areas, such as musical genre classification, where instrumentation may serve as a feature [42].

Quite a lot of work has been done on the subject already. Depending on the complexity of the task (number of possible instruments to be recognised, the polyphonic or monophonic nature of the recordings etc.) classification accuracies between 60 and 92% have been achieved over the past three years [30, 53], which can be seen as a rather successful accomplishment. Instrument identification is a supervised classification problem, i.e., it requires annotated data in order to train a classifier, as opposed to unsupervised tasks, which operate on unannotated data. To obtain such annotated datasets is quite laborious: even though there are limitless possibilities to collect the audio, its annotation requires tedious and expensive human work. Therefore, a requirement of a technique that would overcome this complication is rather apparent.

1.2 Semi-supervised learning

Semi-supervised learning (SSL) is a technique that is meant to address the requirement of large datasets needed to train a classifier which would demonstrate a sufficient level of generalisation capability. Basically, the larger and more diverse the training dataset is, the better generalisation properties one may expect to achieve. In SSL, this dataset extension is approached by incorporating additional data that is not annotated. There exist several SSL schemes, which differ in the way they treat the annotated and unannotated data and from which conventional pattern recognition concept they originate.

Semi-supervised techniques have shown to be successful in numerous machine learning tasks, such as text classification [45], computer vision [59], network traffic classification [17], as well various audio-related problems [33, 60], including music information retrieval [38, 50, 58], to mention a few. However, they have not yet been applied to the musical instrument recognition problem. The related works within neighbouring areas deal with one of the SSL techniques for singing voice detection [36] and the idea of weak labelling (where a label indicates appearance or absence of an instrument in a mixture) for instrument recognition [39].

1.3 Objectives and main results of the thesis

The objectives of this thesis consist of studying techniques for musical instrument recognition as well as various SSL schemes. Furthermore, a subset of those is to be evaluated for two instrument classification scenarios in terms of applicability for the given problem.

The main result of the thesis is a developed pattern recognition system for musical instrument recognition that utilises a selected SSL scheme. Two alternative algorithms previously applied to different pattern recognition problems are implemented, and their performance is evaluated, compared and analysed.

1.4 Organisation of the thesis

The thesis is organised as follows. Chapter 2 studies briefly the broad concept of pattern recognition, extending towards examples of its realisation for the problem of musical instrument recognition, both conventional and state of the art. Subsequently, it reviews the basic terms of SSL and several common ways to approach it. It also includes examples of audio-related areas where SSL has already shown its advantage over supervised learning.

Chapter 3 introduces the details of implementation of the particular music instrument recognition system, which is developed based on one of the studied SSL schemes. Its performance is thereupon evaluated in Chapter 4 accompanied by a comparison of two versions of the algorithm that employ the selected scheme. Finally, several conclusions about the applicability of the implemented system are drawn along with suggestions for the future research directions in Chapter 5.

2. LITERATURE REVIEW

In this chapter, firstly, a review of the pattern recognition concept is given along with the description of the basic building blocks of a generic pattern recognition system. Secondly, an overview of the state of the art musical instrument classification systems is presented as well as a description of the commonly used feature extraction methods.

Then the discussion moves to the concept of SSL, which is introduced with several commonly used schemes and assumptions behind them. Finally, the recent areas of application of SSL for audio-related problems are presented.

2.1 Pattern recognition

The term *pattern recognition* refers to finding similarities between the objects and making certain decisions according to these similarities. It is the act of taking in raw data and performing an action based on the "category of the pattern" [16, Chapter 1, p. 3]. Pattern recognition systems are applicable in numerous areas, from junk mail filtering to cancer cell detection.

The objects (instances) are represented by *feature vectors*, each denoted as $\mathbf{x} = (x_1, ..., x_D)$, of dimension D. Feature vectors consist of information, relevant in terms of a particular application, that is inferred from the data with the purpose of efficiently distinguishing the objects while performing dimensionality reduction. The concept of features is described in more detail in Section 2.1.2. A pattern recognition system operates on the objects based on their features, so different objects with the same features are indistinguishable.

As a rule, a pattern recognition system should be capable of generalisation. In other words, it should be able to distinguish the true importance of variations in the input data from noise and to continue to perform accordingly when the input data stops resembling the data the system was trained on. The generalisation property is known as *induction* [43].

2.1.1 Learning scenarios

A pattern recognition system is developed with the use of data samples, i.e., the example instances of the feature vectors originating from the objects similar to the ones the system is expected to operate on. Depending on what is inferred by a pattern recognition system from a data sample, it is traditionally thought of as belonging to one of the paradigms: *unsupervised* and *supervised* learning.

In case of unsupervised learning, there is no knowledge on how to process the data samples, but just the samples on their own. Among the areas of application of unsupervised learning one could mention the following:

- Clustering — grouping the objects according to their somehow defined similarities. It attempts to split the data according to the objects' alikeness without the information about not only the class labels, but often even the number of possible classes.
- Anomaly detection — finding samples that are significantly different from the majority.
- Dimensionality reduction — comparably efficiently representing samples by a feature vector of a dimension, lower than the original one.
- Data visualisation, which utilises dimensionality reduction methods in order to represent a high-dimensional data on a two-dimensional plot, making it possible to visually analyse the data.

In supervised learning, a label is assigned to each data sample, referred to as a training sample, defining the class that has produced the object with such features. Supervised pattern recognition stands for taking certain actions on a new object based on the knowledge obtained during the training stage, where the training has been conducted by means of classified (labelled) data. A category label or cost for each pattern in a training set is provided, and an attempt to reduce the sum of the cost for these patterns is made [16, Chapter 1, p. 16], where cost is a measure that defines the impact of taking an undesirable action (or making a misclassification).

There are numerous examples where supervised learning is utilised: speech recognition, optical character recognition and medical diagnosis, to mention a few. The existing mechanisms implemented in such systems are quite efficient, however, for each particular application there is a need to collect large amounts of data that needs to be labelled or annotated. It is relatively easy to collect the data as such, but its annotation is quite costly since there is a requirement for human experts to manually conduct the annotation. For example, in the case of annotating a musical piece for the application of tempo recognition, an expert needs to listen to the piece several times, manually establishing the starting points of each measure (in a manual scenario) or correcting the wrongly detected starting points (in a semi-automated scenario). Given the huge amount of data needed for training, it renders it highly inefficient to rely on such scenario. Semi-supervised techniques (see Section 2.3) are one of the possible ways to overcome this issue.

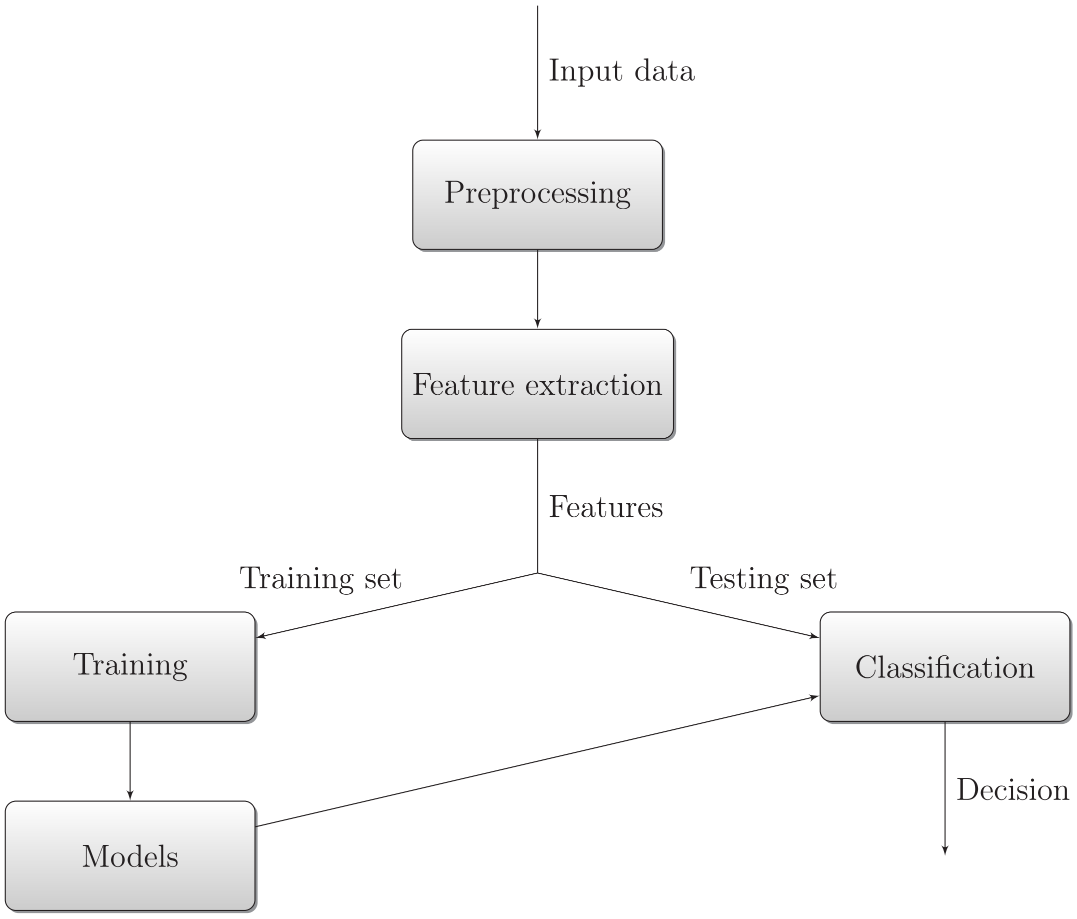


Figure 2.1. *A basic block diagram of a typical pattern recognition system.*

2.1.2 Structure of a pattern classification system

Typically, a pattern recognition system intended to solve classification problems consists of the building blocks that are presented in Figure 2.1. Their structure and functionality are described below.

In the stage of *preprocessing*, the original data is prepared in such a way that it is somehow optimised for the subsequent step of feature extraction. It often involves a segmentation operation. For example, in the case of optical character recognition, before the features of the characters are extracted, there is a requirement to find the location and boundaries of these symbols. In audio-related applications, a corresponding operation is applied quite often in a form of segmenting the original waveform into smaller blocks (frames), which are easier to handle and that are thought of as having somewhat stationary spectral characteristics.

The task of *feature extraction* is to preserve information, relevant in terms of a particular pattern recognition task, while significantly reducing the amount of data. The latter requirement is caused by the fact that in most pattern recognition scenarios the volume of original unprocessed raw data is quite high, which could result in the so-called *curse of dimensionality* — a limitation, due to the fact that high dimensionality can lead to complications, whose significance is harder to distinguish. [16, Chapter 4, p. 15].

In the case of face recognition, for example, one of such features could be the distance between the eyes, which significantly reduces the dimensionality (from the total number of pixels in the image to a single number) while maintaining much of the information relevant for the problem of interest. In the case of musical instrument recognition, such a feature is desirable that would extract the information about the timbre (i.e., the characteristic that uniquely describes musical instruments of a particular group, see Section 2.2.2) of the instrument, discarding the information irrelevant in terms of the problem of interest (such as fundamental frequency, level of reverberation, channel effects and so forth).

Training is the process of discovering a rule that explains the distribution of the data samples in the feature space. In the case of classification problems, it can be seen as finding decision boundaries in the feature space between classes. For example, assume a musical instrument classification problem, which is approached with a two-dimensional vector of such features as bandwidth and spectral centroid (i.e., center of gravity of the spectrum). Suppose also that such choice of features is sufficient for representing different instruments in the separate regions of the feature space. Then, during the training stage, continuous boundaries between these areas characterised by discrete training samples would be found. Consequently, these would be utilised for classification. It is worth mentioning, though, that such a scenario is artificial and in a more realistic case a much higher dimension of the feature space as well as a more complicated feature extraction mechanism are required.

Depending on whether the training aims to generate a prediction function defined on the whole feature space, there exist two different learning settings: transductive and inductive [62]. In the case of *transductive learning* prediction is performed only for the test points. The goal of *inductive learning* is to obtain a classifier defined on the entire feature space.

Generative algorithms are those that try to model class-conditional density $p(\mathbf{x}|y)$, where y denotes a class label associated with a feature vector, by some unsupervised learning procedure. *Discriminative algorithms* do not try to estimate how $\mathbf{x}_i$ have been generated, but instead concentrate on estimating $p(y|\mathbf{x})$. Generative models estimate the density of $\mathbf{x}$ as an intermediate step, while discriminative methods directly estimate the labels. A strength of the generative approach is that knowledge of the structure of the problem or the data can be naturally incorporated by modelling it [10].

In the case of clustering, the training samples are represented by feature vectors without any additional information, as a rule. Although this data set may be referred to as a "training set", the training stage *per se* does not necessary have to be present separately in the system. Instead, it may be conducted at a later stage in such a way that the whole data is grouped into several clusters depending on samples' positions in feature space.

In the supervised scenario, the training samples correspond to feature vectors with labels assigned to each of them. The training stage performs an estimation of the rule according to which samples belonging to different classes are located in their particular areas of feature space. As a result, models are obtained and consecutively used in order to classify previously unseen data.

Semi-supervised learning can be thought of as an extension of either supervised or unsupervised learning, and the training stage is where these concepts differ the most. Further description of these concepts is presented in Section 2.3.

In the *classification* stage the patterns learnt during training are applied to the new, unseen data in order to produce a classification decision. The new data has undergone the same preliminary stages as the training data with the purpose of matching the learnt patterns with the ones produced by the unseen data.

The final stage of the development procedure of any pattern recognition system is *evaluation*. Moreover, it is also included as an intermediate part when searching for possibilities to maximise the performance level of a system in development. The evaluation stage shows the level of accuracy which can be achieved with a current implementation. It is highly important for any pattern recognition system as it is meant to show the system's effectiveness in real-world applications. However, not all the methods are well suited to representation of real-world performance.

2.1.3 Methods of evaluation

There exist three basic approaches to evaluate the pattern recognition system performance: *resubstitution*, *leave-one-out* (or *cross-validation*) and *holdout* methods [52, p. 570]. The essential difference between them is in the way they utilise the labelled data for training and testing.

When the resubstitution method is applied, the same data is used for training and testing, which is most likely to produce an overoptimistic result. The fact that the classifier trained on a particular piece of data is able to recognise it quite accurately does not necessary imply that it will perform equally well when previously unseen data is introduced. Furthermore, trying to achieve a high recognition accuracy that is measured by means of the resubstitution method may actually result in worse generalisation properties of the system and, therefore, lower performance level. This highly undesirable phenomenon is called *overfitting*. Resubstitution is known to produce a biased estimate of the error rate, and in [23] this bias is shown to be dependent upon the ratio of the sample size per class to the feature size. Strictly speaking, the use of this method is considered ill-advised for a classification problem with insufficient training set size.

The leave-one-out method is applied when the amount of labelled data is limited and it is desired to use all of it for training purposes. In that case, the evaluation is conducted in such a way that all the data except for one sample is used for training, and that one sample is classified during evaluation. Then another sample is picked for evaluation, and the classifier is retrained on the rest of data. This process is repeated until all the samples have been used once for testing, then their classification accuracy is calculated as a ratio between number of correct classification cases to the size of data. The advantage of this method is that it provides an adequate measure of performance while utilising the maximum available labelled data. The drawback is its computational complexity since it is required to conduct training as many times as there are samples in the dataset.

The holdout method is applied in such a way that the dataset is divided into non-intersecting sets of training and testing data. This method lacks the drawbacks of resubstitution method, as the test data is always previously unseen, and simultaneously it is much less computationally complex than the leave-one-out method. However, it requires sufficient amount of labelled data in order to function properly.

The above-mentioned methods describe different ways of handling available data in evaluation. The evaluation as such includes most commonly the average accuracy calculation:

$$\text{Accuracy, \%} = \frac{C}{T} \times 100\%, \tag{2.1}$$

where C is the number of correctly classified test instances and T is the total number of test instances. Alternatively, one may utilise the average classification error measure:

$$\text{Average error, \%} = \frac{T - C}{T} \times 100\%. \tag{2.2}$$

A logical extension of that measure is a *confusion matrix*, which is designed to illustrate to what extent models of one class affect models of another one. In other words, it shows how mutually confusable the classes are and is represented in the following form:

$$\begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1M} \\ s_{21} & s_{22} & \cdots & s_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ s_{M1} & s_{M2} & \cdots & s_{MM} \end{bmatrix},$$

where M is the total number of classes. A confusion matrix is constructed in such a way that each element of the matrix s_{ij} represents the number of cases when the data sample belonging to class with index i is classified as belonging to class with index j. Sometimes it is worthwhile to divide the matrix by the total number of tests in order to obtain a normalised characteristic of the confusability of the classes. The utilisation of confusion matrices allows a more thorough study of the effectiveness of the applied feature extraction, as well as classification and training methods.

2.1.4 Sources of error

Performing an evaluation with the above-described methods during the implementation of a pattern recognition system is highly important when aiming for a better performance. The latter is done by investigating the source of the test error and taking the measures accordingly.

There are three main sources where errors originate. Firstly, it is the insufficiently effective choice of features, which leads to overlaps of the areas occupied by the classes in the feature space. This error is referred to as the *Bayes error*, and, given a particular feature extraction procedure, it can never be reduced. Secondly, incorrect models for class-conditional densities are a source of a so-called *model error*, when the distribution is not in accordance with the actual data points.Such error can be naturally reduced by selecting a more appropriate model in terms of the problem of interest. Finally, one may observe an *estimation error*, which originates from the insufficient number of training samples. It can be diminished by increasing the data set size, however, having a larger data set may lead to a consideration to use more accurate models in order to minimise the model error, which reinstates the estimation error, whose minimisation requires even bigger dataset. In other words, there is always a trade-off between these two errors.

2.2 Musical instrument recognition

Quite a significant amount of research has been conducted on the subject of musical instrument classification. It has been most actively explored since the 1990's, when the systems aimed at handling small numbers of instruments represented by isolated notes were already reaching impressive performance scores of 98% [35] and 100% [32]. During the following years, various systems realising numerous methods and applied for different numbers of instruments have been developed. An extended summary of the early works can be found in [19].

In the upcoming sections, an overview of the state of the art works on the subject is presented. Consequently, it is followed by a summary of features that have attracted most interest in the area of musical instrument classification.

2.2.1 State of the art

The recent works on the subject of musical instrument classification are summarised in Table 2.1. Despite the presented values of maximum achieved accuracy (or F-measure) for each work, comparing them by such value would not be impartial due to significant differences between the problems those systems are intended to solve. To start with, some of them approach polyphonic cases, whereas the others confine themselves with solo recordings. Furthermore, some systems applying novel approaches do not necessary aim at outperforming the traditional methods but demonstrate the potential of the suggested techniques. Consequently, no comparative analysis of these works is presented but merely an overview of their novelties and particularities.

Among the recent works that address the problem of polyphonic musical instrument recognition, one could mention the three following. Burred *et al.* [8] treat timbre of the musical instruments based on the spectral envelope and its evolution in time. They suggest an approach of grouping sinusoidal trajectories according to common onsets and using a set of pre-trained templates, which contain information about the temporal evolution of the spectral envelopes, to compare these groups. Using a set of recordings of five instruments, they obtain best classification accuracies of 79.7%, 77.8% and 61.4% with polyphony of two, three and four voices respectively.

Another approach to polyphonic musical instrument recognition proposed by Heittola *et al.* [30] includes incorporating a source-filter model and an augmented non-negative matrix factorisation algorithm for sound separation. The essence of this novel approach is in decomposing the mixture signal into a sum of spectral bases modelled as a product of excitations and filters. As a result, a significant reduction of the interference of the instruments playing simultaneously is achieved, which in case of whole 19 instruments case results in F-measures of up to 60.2%, 58.0%, 57.9%, 55.9% and 59.1% when applied on the polyphonic cases of two, three, four, five and six voices respectively.

Barbedo and Tzanetakis [5] approach the problem of polyphonic musical instrument recognition by utilising the spectral disjointness amongst instruments. The proposed method consists of identifying isolated partials, i.e., partials, that do not collide with any others, which are expected to be present in a polyphonic recording at least at some point. While using these partials, the conventional features (based on spectrum, amplitude envelope and frequency trajectory) are extracted. The evaluation is performed on polyphonic signals synthesised from isolated notes as well as real recordings. Additionally, the system is tested under noisy conditions, leading to a conclusion that white noise is the most harmful to the performance. Two databases are utilised for the tests with 15 and 25 instruments, and the highest achieved accuracy in the synthesised scenario is 78.7%.

Considering again the monophonic scenarios, Joder *et al.* [34] demonstrate the usefulness of incorporating the mid-term temporal properties of the signal, i.e., the information carried by the temporal evolution of the features. They perform

Table 2.1. *State of the art works on musical instrument recognition, where* Acc. *refers to highest achieved classification accuracy or F-measure and* # ins. *is the number of instruments or instrument groups used as separate classes.*

Work	**Particularity**	**Polyphony**	**# ins.**	**Acc., %**
Burred *et al.* 2009 [8]	dynamic model of spectral envelope	yes	5	79.7
Heittola *et al.* 2009 [30]	source-filter model for separation	yes	19	60.2
Joder *et al.* 2009 [34]	temporal integration	no	9	84.5
Loughran *et al.* 2009 [41]	genetic algorithms	no	5	64.0
Sturm *et al.* 2010 [51]	multiscale MFCCs	no	7	84.7
Tjoa & Liu 2010 [53]	temporal information	no	24	92.3
Liu & Xie 2010 [40]	Chinese music	no	8	87.2
Barbedo & Tzanetakis 2011 [5]	individual partials	yes	25	78.7
Rui & Bao 2012 [49]	projective NMF	no	11	87.9

both early (over larger time *texture windows*) and late integration (with the aid of *alignment kernels*). The experiments show improvement of the performance as a result of utilising temporal integration, which, on the other hand, possibly increases the problem dimensionality. A maximum accuracy of 84.5% is shown in 9-instrument case, which outperforms the reference score of 81.6% achieved with the state of the art system.

Loughran *et al.* [41] approach musical instrument recognition with the idea of using genetic algorithms with the purpose of optimising feature selection. Those are a continuous extension of the binary genetic algorithms, which have previously shown somewhat encouraging results when utilised to reduce features in musical instrument classification studies. Out of 95 initial features, the proposed algorithm does indeed select the most optimal ones, followed by classification with a multi-layered perceptron. The successfulness of the choice of features produced by the genetic algorithm is justified by evaluating the performance with reduction of features, which has not degraded the performance.

A multiscale extension of the mel-frequency cepstral coefficients (MFCCs) feature has been suggested in [51] by Sturm *et al.*, i.e., evaluating MFCCs over several time scales. MFCCs are initially intended for speech-related applications, and speech is relatively well-behaved signal compared with other manners of sound production. The authors believe that the cepstrum of musical signals, when computed over a single time-resolution, cannot distinguish between different phenomena that occur over many time-scales in the mixtures that musical signals contain. The work proposes decomposing a signal by a greedy iterative descent method of sparse approximation using a multiresolution time-frequency dictionary of Gabor atoms; then finding the distribution of the energy in the signal as a function of atom scale and modulation frequency; and then reducing redundancy of the feature space with the aid of discrete cosine transform. The performance of the suggested approach is evaluated on monophonic signals, which are, however, derived from real musical settings: the recordings are not isolated single notes, but include extended performance techniques and non-traditional styles. Each class is represented by five instances, which differentiate by performers, instruments and so forth. A substantial improvement of performance of the multiscale decomposition is shown over the single time-resolution features. A classification accuracy of 84.7% is achieved with one of the multiscale approaches. This outperforms the traditional MFCCs with delta coefficients, which demonstrate the accuracy of 81.0%.

Considering spectral and temporal information equally important in the definition of timbre, Tjoa and Liu [53] combine advances in dictionary learning, auditory modelling and music information retrieval to propose a new timbral representation. They suggest a novel method of extracting temporal information using a multiresolution gamma filterbank which is computed from the temporal atoms extracted from spectrograms using nonnegative matrix factorisation. The evaluation is performed on 24 instruments and shows 72.9% accuracy when incorporating solely temporal information, 88.2% with spectral information and 92.3% in their combination.

Liu and Xie [40] apply support vector machine (SVM) to classify Chinese traditional and western classical music. This widely used machine learning technique has been proven to be an efficient classifier in various music information retrieval areas, and the novelty of this work is that it is applied to Chinese instruments classification. By utilising it in a combination with several common feature extraction methods on more than 26 instruments grouped into 8 families, a maximum accuracy of 87.2% is achieved. Regarding the Chinese instruments in particular, it has been shown that MFCCs, being an efficient feature in general, does not distinguish sufficiently well the Chinese and western percussion instruments, which suggests the necessity of employing some feature selection algorithm.

Rui and Bao [49] propose a novel learning algorithm for classifying instruments by utilising the projective non-negative matrix factorisation with Bregman divergence. As opposed to the conventional non-negative matrix factorisation (NMF) method, which has been shown to be unable to guarantee the sparsity and localisation of the matrix, the projective extension of the method has proven to have better sparsity and orthogonality of basis matrix. The evaluation is performed on separated notes belonging to 11 instruments and shows the average accuracy 87.9%, which outperforms the conventional non-negative matrix factorisation method by 1.2%.

2.2.2 Features applied in musical instrument classification

The task of the feature extraction stage in any pattern recognition system is to reduce the dimensionality of the data, preserving the most relevant information about the objects to be classified. A carefully designed feature extraction algorithm defines what is relevant for a particular classification task and is able to assign such values to the objects that would be as close to each other, as are the objects to be classified in a particular context.

In the context of musical instrument classification, the relevant information that helps separate the classes is within their *timbres*. A timbre is a broad term referring to the characteristics of a musical instrument that are unique to a particular group of instruments (strings), type of instruments (violin), or, even further, particular playing manner (pizzicato).

From a musician's point of view timbre is the set of tonal qualities which characterise a particular musical sound. Generally speaking, timbre may indicate any acoustic phenomena other than pitch, loudness, duration and spatial location [27]. Since the very notion of timbre has variety of definitions, there is not therefore any

Table 2.2. *Features commonly applied in selected works on musical instrument recognition.*

Feature	**Works that utilise the feature**
MFCCs	[12, 14, 18, 20, 21, 34, 51, 41, 40, 30]
Autocorellation coefficients	[20]
Zero-crossing rate	[1, 2, 20, 34, 41, 40]
Amplitude envelope	[18, 41]
Spectral centroid	[18, 1, 2, 20, 34, 41, 40]
Bandwidth	[1, 2, 20]
Spectral skewness	[1, 2, 20, 34, 41]
Inharmonicity	[1, 2]
Octave band signal intensities	[20, 34, 41]

certain fixed way of incorporating the complete information about timbre into a pattern recognition system. It is quite transparent how to compose features that can be used to extract information about pitch, loudness or duration — i.e., everything the timbre is not. Therefore, empirical approaches are utilised, incorporating the features that have proven to be effective in other audio-related fields.

An overview of most commonly used features for musical instrument classification along with a list of selected works that utilise them is presented in Table 2.2. The features are defined in the following paragraphs. It is worth noting that not only these features as such are applied but often along with the values of their standard deviations, which have shown to be informative in modelling changes of the audio attributes. For a more extensive overview of the features applied for musical instrument recognition, see [19].

Temporal features

Zero-crossing rate is the relative number of zero-value crossings within a frame:

$$\text{Zero-crossing rate} = \frac{1}{2N} \sum_{k=1}^{N-1} \left| \operatorname{sgn}\big(s(k) - s(k+1)\big) \right|, \tag{2.3}$$

where s represents values of a sampled audio signal within a frame, N is a number of samples within that frame, k is a sample index and

$$\operatorname{sgn}(y) = \begin{cases} -1 & \text{if } y < 0, \\ 0 & \text{if } y = 0, \\ 1 & \text{if } y > 0. \end{cases}$$

One of the examples of application of this feature is classification of percussive sounds [26].

Amplitude envelope contains information about the type of excitation, for example, whether a violin has been bowed or plucked. It can be calculated by half-wave rectification and low-pass filtering of the signal or by the means of short time root-mean-square energy of the signal. [18]

Spectral features

Among the spectral features, the ones that are related to the harmonic properties of a sound appear to play a crucial role in musical instrument classification. As shown in [1], by extracting just the harmonic properties, a simplified representation of data is obtained, so that the important characteristics of the musical instrument timbre are preserved. Additionally, a compact representation like this is a way to cope with the curse of dimensionality. A selected list of such features is presented below.

Spectral centroid is the mean frequency of the spectrum, and the verbal label associated with it in timbre perception studies is referred to as *brightness* [27]. It is defined as

$$\text{Centroid} = \frac{\sum_{f=f_{\min}}^{f_{\max}} f \cdot E(f)}{\sum_{f=f_{\min}}^{f_{\max}} E(f)}, \tag{2.4}$$

where $f_{\min} = 80$ Hz, $f_{\max} = 5000$ Hz and $E(f)$ is the energy of the spectral component at frequency f [2].

Bandwidth may be calculated as magnitude-weighted differences between the spectral components and the centroid [1]:

$$\text{Bandwidth} = \frac{\sum_{f=f_{\min}}^{f_{\max}} |\text{Centroid} - f| \cdot E(f)}{\sum_{f=f_{\min}}^{f_{\max}} E(f)}. \tag{2.5}$$

Inharmonicity is a cumulative distance between the estimated partials and their theoretic values [1]:

$$\text{Inharmonicity} = \sum_{i=1}^{4} \frac{|p_i - i \cdot f_0|}{i \cdot f_0}, \tag{2.6}$$

where p_i is the partial of index i and f_0 is the fundamental frequency. This feature may be relevant for such instruments as plucked strings or piano.

Harmonic energy skewness — sum of energy confined in the partials region, multiplied by the respective inharmonicities [1]:

$$\text{Harmonic energy skewness} = \sum_{i=1}^{4} \frac{|p_i - i \cdot f_0|}{i \cdot f_0} \cdot E_{p_i}, \tag{2.7}$$

where E_{p_i} is the energy of the i-th partial.

Octave band signal intensities — the log energy of the spectrum within each of the subbands, obtained using an octave filterbank with triangular frequency responses, whose edges are mapped to musical note frequencies starting from the lowest Piano note A1 [20]. This is also meant to represent the differences in harmonic structures of the spectra of different instruments.

Mel-frequency cepstral coefficients

Mel-frequency cepstral coefficients is a feature that has proven to be reliable in various audio applications, starting from speech recognition [13, 48] and then expanding its area of usage into other audio-related applications [44, 39]. It was introduced in [13] as a way to characterise syllables and is defined as

$$c_i = \sum_{k=1}^{N} Y_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{N}\right], i = 1, 2, \ldots, M, \tag{2.8}$$

where Y_k is the log energy within each mel band. The latter are obtained with the aid of a mel-spaced filterbank, i.e., triangular filters spaced uniformly across the mel-frequency scale, approximated by the following equation [47]:

$$\mathrm{Mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right). \tag{2.9}$$

The original idea behind computing MFCCs in speech recognition is that it enables extraction of the information about the spectral envelope. Its importance is considered quite high since with its aid it is possible detect formants, which characterise the speech content quite significantly in terms of speech recognition.

In the musical instrument signals, however, the presence of formants in the spectrum is rarely strong [32], or they are not a factor independent from fundamental frequency, in contrast to speech signals. For example, in the spectra of trombone or clarinet, due to the acoustical change of active volume of their body during the sound production, the resonances depend on pitch [37, 22]. Still, the level of performance of MFCCs is quite satisfactory when applied to the musical instrument classification problem as well, which can be explained by its correspondence to the way human auditory system recognises audio [12]. Consequently, MFCCs were one of the first features used in the early stages of the development of the systems for musical instrument recognition [12, 14] and proven to be effective amongst other features in this area as well [18].

The block diagram of an MFCCs calculation is presented in Figure 2.2. The functionality of its building blocks is discussed in the following paragraphs.

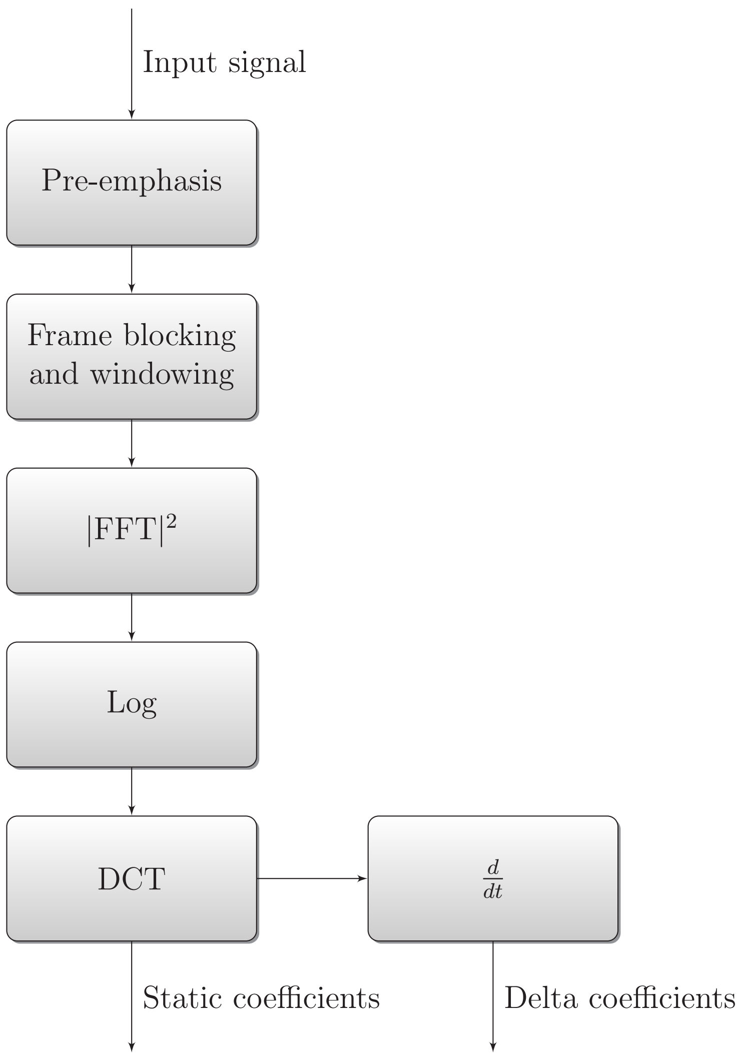


Figure 2.2. *A block diagram of MFCCs calculation.*

Firstly, the input signal is pre-emphasised, i.e., a high-pass filtering (typically, with a $1 - 0.97z^{-1}$ filter) is performed. The motivation behind this is that the spectrum of a musical audio signal, as well as many other types of audio signals, contains normally a lot of energy in its low-frequency region. On the other hand, the timbre information does not rely much on low frequencies. Moreover, when human ear analyses audio, the low frequencies are to some extent suppressed. Hence, the pre-emphasis stage is performed in order to flatten the spectrum and equalise the contribution of its components into the subsequent feature stage of feature extraction.

The next step is frame blocking and windowing. Frame blocking is a mechanism of preliminarily segmenting an audio signal in such a way that it is divided into frames of 10–20 milliseconds length with the aid of overlapping windows. This step is motivated by the fact that during the length of a window the spectral characteristics of a signal may be considered stationary.

Subsequently, the power spectrum of the signal is computed and decimated to the mel scale. This is done in order to simulate the phenomenon of critical bands, inherent in human auditory system, as well as the logarithmic property of frequency perception.

The next step is to compress the dynamic range of the spectrum by applying a logarithm to each mel-band energy. Both this and the previous steps also ensure that the result will match the auditory properties, namely, the logarithmic relation of the subjective sensation (in this case, the loudness and pitch) and stimulus intensity (the amplitude and the frequency of the spectral components), known also as the Webber–Fechner law.

Thereupon, a discrete cosine transform (DCT) is applied in order to decorrelate the coefficients and to be able to discard part of data, namely, the zeroth and the higher coefficients. The former characterises the gain component, usually irrelevant in classification problems, while the latter account for pitch and fine spectral structure. The remaining part is the lower coefficients, which are responsible for the spectral envelope and enable the generalisation of the timbre of the instrument.

Finally, the delta coefficients are often calculated as well, by computing the time derivative of the approximation of the recent coefficients trajectory. This helps preserve the dynamic properties of the spectral envelope [48, p. 116].

2.3 Semi-supervised learning

As noted in Section 2.1.1, generally, supervised learning approach poses a requirement of large annotated training sets. Semi-supervised learning is introduced in order to cope with such practical issue. As an extension of supervised learning, it is meant to overcome the difficulties caused by the need of large amounts of annotated data. By introducing the requirement of having only part of data annotated and utilising the easily obtainable unlabelled data to further train the classifier, the mentioned issues are expected to be resolved.

Such type of SSL that extends the supervised classification problem is called *semi-supervised classification*. Having a training set that consists of both labelled $\{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$ (where $\mathbf{x}_i$ is a feature vector, y_i is a label associated with it, and L is a number of labelled samples) and unlabelled data $\{\mathbf{x}_i\}_{i=L+1}^{L+U}$ (U — a number of unlabelled samples) used together to train a classifier has the purpose of achieving performance close to the one that could be reached in the case when all the data of the same size were labelled.

However, there are other types of SSL scenarios, among which one example is constrained clustering [55] — an extension of unsupervised learning problem where in addition to unlabelled data, a limited amount of information about the clusters is provided. Being intended to solve problems, other than classification, those are left behind the scope of this work.

There exist several SSL schemes, such as mixture models (e.g., [9, 46]), self-training (e.g., [57, 4]), co-training [7], graph-based schemes (e.g., [6, 24, 3, 61]) and semi-supervised support vector machines [11]. Each of them rely on its own assumptions about how the marginal distribution $p(\mathbf{x})$ is linked to conditional distribution $p(y|\mathbf{x})$ [62, p. 13]. A reasonable choice of a scheme based on an adequate assumption may lead to improvement in the system's performance, whereas a wrong choice will make it worse. Therefore, understanding the adequacy of the assumption for a given classification task is of crucial importance. The following subsections describe various SSL schemes, as they are presented in [62, pp. 15–55], with a relation to the assumptions behind them.

2.3.1 Self-training

Self-training means that the predictions of the learning process are used to teach it [62, p. 15]. Essentially, the algorithms for self-training are developed in such a way that the predictor, obtained from the labelled instances, is applied to classify the unlabelled part, and then part of the latter (normally the part that was classified most confidently) with the obtained labels is moved to the labelled set, after which the procedure repeats. This is a *wrapper* method, meaning that it wraps around an arbitrary prediction method, making it therefore easy to implement based on existing supervised solutions for the given problem.

The *assumption* of self-training is that the predictor is correct at least for the classifications it was most certain in. In other words, the classes form well-separated clusters for the particular dataset. [62, p. 16]

2.3.2 Mixture models

Mixture models for supervised learning

Mixture model is a way of representing an unknown $p(\mathbf{x})$ as a linear combination of density functions:

$$p(\mathbf{x}) = \sum_{y=1}^{M} p(\mathbf{x}|y)P_y, \tag{2.10}$$

where

$$\sum_{y=1}^{M} P_y = 1, \quad \int_{\mathbf{x}} p(\mathbf{x}|y)d\mathbf{x} = 1.$$

The idea behind mixture models is in decomposing the mixture into individual components as a means of representing a distribution to be estimated.

This implies that each feature vector may be drawn from any y of the M model distributions with probability P_y. Given an adequate number of mixtures, the model may arbitrarily closely represent any continuous density function. In order to obtain such a model, a set of density components $p(\mathbf{x}|y)$ needs to be chosen in parametric form $p(\mathbf{x}|y;\theta)$ and then the unknown parameters θ and P_y are computed based on the training set. [52, p. 44]

In the case of equal costs for correct and erroneous classifications, a straightforward approach to obtain labels is to maximise their conditional probability:

$$\hat{y} = \arg\max_{y} p(y|\mathbf{x}). \tag{2.11}$$

In order to compute $p(y|\mathbf{x})$, one could use the Bayes' rule

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)P_y}{\sum_{y'=1}^{M} p(\mathbf{x}|y')P(y')}, \tag{2.12}$$

thus turning the task of maximising the unknown distribution of all the possible labels for a particular feature vector $p(y|\mathbf{x})$ into the task of maximising the product of class-conditional probabilities ($p(\mathbf{x}|y)$), estimated from the training data, and prior probabilities P_y. The latter are either assumed to be distributed uniformly in the case of *separate sampling*, i.e., when training data is collected separately for each class, or are estimated from the training data as well in the case of *mixture sampling* in the following manner:

$$P_y = \frac{n_y}{\sum_{i=1}^{M} n_i}, \tag{2.13}$$

where n_y and n_i are the numbers of labelled training data instances originating from the classes y and i, respectively.

An example of such models is Gaussian mixture model (GMM), which utilises the following multivariate Gaussian distribution:

$$p(\mathbf{x}|y) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_y|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_y)^T \boldsymbol{\Sigma}_y^{-1}(\mathbf{x}-\boldsymbol{\mu}_y)\right), \tag{2.14}$$

where $\boldsymbol{\mu}_y$ is the mean vector and $\boldsymbol{\Sigma}_y$ is the covariance matrix. Such distributions may be combined with different weights, as well as with different values of the mean and covariance, to represent an arbitrary distribution with a certain level of accuracy (see Figure 2.3). These mixtures constitute an overall GMM, which is estimated for each class y.

Another example of such models is the hidden Markov model (HMM) — a doubly embedded stochastic process with an underlying stochastic process that is *not* directly observable, but can be observed only through another set of stochastic pro-

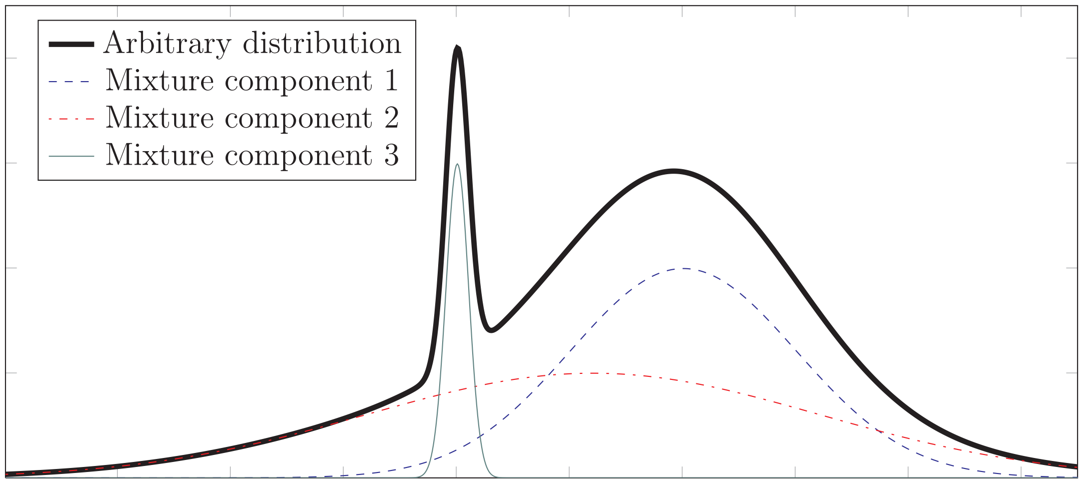


Figure 2.3. *An example of representing an arbitrary one-dimensional distribution as a mixture of three Gaussians.*

cesses that produce the sequence of observations [48, p. 326]. These are commonly used in applications where tracking the sequence of instances is important. HMMs define conditional distributions for its states (using, for instance, GMM), as well as probabilities of transitions between the states.

A straightforward approach to apply generative models is to find such parameters of the model that will maximise the likelihood $p(\mathcal{D}|\theta)$ of the training data $\mathcal{D}$ (or log-likelihood $\log p(\mathcal{D}|\theta)$, since logarithm is monotonic and yields simpler calculations):

$$\hat{\theta} = \arg\max_{\theta} p(\mathcal{D}|\theta) = \arg\max_{\theta} \log p(\mathcal{D}|\theta). \tag{2.15}$$

This criterion is known as the maximum likelihood estimate (MLE).

In supervised learning ($\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$) the optimisation problem of obtaining MLE reduces to finding the parameters of the model that would best fit the given training data features $\mathbf{x}_i|_{i=1}^{L}$ and the prior information $P(y_i)$ inferred from $y_i|_{i=1}^{L}$:

$$\hat{\theta} = \arg\max_{\theta} \log p(\mathcal{D}|\theta) = \arg\max_{\theta} \sum_{i=1}^{L} \log P(y_i)p(\mathbf{x}_i|y_i, \theta). \tag{2.16}$$

This can be done analytically, and in case of GMM the results are the following: MLE for each class mean is class's sample mean, and covariance matrix is the sample covariance for the instances of that class. [62, p. 25]

Mixture models for SSL

In the case of SSL ($\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_L, y_L), \mathbf{x}_{L+1}, \ldots, \mathbf{x}_{L+U}\}$) the analytical solution is not possible, since likelihood depends on unlabelled data as well [62, p. 25].

The log-likelihood function in this case is defined as

$$\log p(\mathcal{D}|\theta) = \sum_{i=1}^{L} \log P(y_i)p(\mathbf{x}_i|y_i,\theta) + \sum_{i=L+1}^{L+U} \log p(\mathbf{x}_i|\theta), \tag{2.17}$$

where $p(\mathbf{x}|\theta)$ is the *marginal probability* — the probability of generating $\mathbf{x}$ from any of the classes:

$$p(\mathbf{x}|\theta) = \sum_{y=1}^{M} p(\mathbf{x},y|\theta) = \sum_{y=1}^{M} P(y)p(\mathbf{x}|y,\theta). \tag{2.18}$$

The unobserved labels $y_{L+1} \dots y_U$ are referred to as *hidden variables*, which constitute hidden data $\mathcal{H}$.

Even though the analytical solution is not possible, a local maximum of the parameter estimate can be found with the aid of the expectation-maximisation (EM) algorithm [15]. It is commonly used for computing maximum likelihood estimates from incomplete data. The algorithm can be applied for SSL based on a mixture model in such a way that it estimates so-called "soft labels" $q^{(t)}(\mathcal{H}) = p(\mathcal{H}|\mathcal{D},\theta^{(t)})$ to the unlabelled data based on a current model and then finds a model that will maximise $\sum_{\mathcal{H}} q^{(t)}(\mathcal{H}) \log p(\mathcal{D},\mathcal{H}|\theta^{(t+1)})$. Two EM-based algorithms for SSL, which are used in this work, are presented in Section 3.4.2.

The EM algorithm can be thought of as a special case of self-training. The difference is that in this case the labels are "soft", i.e., all possible labels are assigned to each unlabelled instance with different weights, compared to the "hard" labelling for the most confident predictions in self-training [62, p. 28].

The *assumption* of SSL based on mixture models is the following: the samples are produced by the *correct* mixture model. In other words, the number of components, prior $P(y)$, and conditional $p(\mathbf{x}|y)$ are assumed to be correct.

2.3.3 Co-training

The essence of co-training, introduced in [7], is in having two separate classifiers, each tailored for a particular part of a feature vector, denoted with a term *view*. Each view is responsible for a particular "kind" of information, and these two classifiers operate in such a manner that classification made by one of them is used to train another one, and vice versa. As an example of such different "kinds" of information, on which these two classifiers could be trained, the authors of the method present contents of a web-page and hyperlinks that lead to this page in a web-page classification scenario.

Similarly to self-training, the unlabelled instances with the most confident predictions are moved into labelled set, but the difference is that one classifier gives these confident predictions, whereas another one uses the newly obtained data to give its own confident predictions to some of the remaining unlabelled instances.

Like self-training methods, co-training is also a wrapper method, allowing the use of arbitrary classifiers. A simplified version of the co-training algorithm [62, p. 37] is presented in Algorithm 2.1.

```
Input: labelled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$, unlabelled data $\{\mathbf{x}_j\}_{j=L+1}^{L+U}$, learning speed $k$.
       Each instance has two views $\mathbf{x}_i = [\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}]$.
Initially let the training sample be $L_1 = \{(\mathbf{x}_1^{(1)}, y_1), \dots, (\mathbf{x}_L^{(1)}, y_L)\}$,
$L_2 = \{(\mathbf{x}_1^{(2)}, y_1), \dots, (\mathbf{x}_L^{(2)}, y_L)\}$.
repeat
    Train a view-1 classifier $f^{(1)}$ from $L_1$, and a view-2 classifier $f^{(2)}$ from $L_2$.
    Classify the remaining unlabelled data with $f^{(1)}$ and $f^{(2)}$ separately.
    Add $f^{(1)}$'s top $k$ most-confident predictions $(\mathbf{x}, f^{(1)}(\mathbf{x}))$ to $L_2$.
    Add $f^{(2)}$'s top $k$ most-confident predictions $(\mathbf{x}, f^{(2)}(\mathbf{x}))$ to $L_1$.
    Remove these from the unlabelled data.
until labelled data is used up.
```

Algorithm 2.1. *Co-training.*

The *assumptions* of co-training are the following [62, p. 37]:

1. each view should be able to produce good classifications on its own given sufficient amount of data;

2. the views should be conditionally independent given the class label:

$$\begin{aligned} p(\mathbf{x}^{(1)}|y, \mathbf{x}^{(2)}) &= p(\mathbf{x}^{(1)}|y), \\ p(\mathbf{x}^{(2)}|y, \mathbf{x}^{(1)}) &= p(\mathbf{x}^{(2)}|y). \end{aligned} \tag{2.19}$$

The latter assumption means that the labelled instances that were obtained with the aid of the first classifier are required to be informative enough for the second classifier.

2.3.4 Graph-based schemes

Graph-based schemes are defined by such a graph where labelled and unlabelled data correspond to nodes and whose edges represent the pairwise distance between the nodes. The edges' weights are related to the similarity between the instances. In the case of an unweighted graph, the weights between connected vertices are set to one. Otherwise, such weight function as

$$w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \tag{2.20}$$

can be used to facilitate the weight's property of decreasing as the Euclidean distance increases.

The unlabelled instances acquire their label based on the weight of the path towards the closest labelled instance. This may contradict with the label obtained by directly measuring the Euclidean distance between instances.

There are two requirements for estimating a label function on the graph: it is expected to give values close to the real labels of the labelled instances (which is expressed by the loss function) and it should be smooth on the whole graph, in other words, the labels are expected to change slowly on the graph [62, p. 51] (the smoothness assumption). The existing methods for graph-based SSL are characterised by the way they meet these requirements.

2.3.5 Transductive support vector machines

The support vector machines (SVMs) algorithm is a non-probabilistic binary linear classifier that for each given input predicts one of two possible classes, finding a linear decision boundary between their regions so that the margin between these regions and the boundary is maximised. The widely used version of the algorithm was proposed in [11].

As an extension of SVMs for SSL, transductive support vector machines (TSVMs) have been introduced in [54]. The idea behind TSVMs is in placing both labelled and unlabelled instances outside the margin. Since the correct side of the decision boundary is unknown, one can incorporate the unlabelled instance into learning by treating the prediction as the putative label and applying the hinge loss function that does not need the real label but is completely determined by the learnt function. [62, p. 61]

2.4 Applications of SSL in audio processing areas

Due to the benefits introduced by semi-supervised techniques, along with the easiness of obtaining raw unannotated audio data, SSL has been applied for solving audio-related pattern recognition problems quite extensively. In this section, several specific selected areas of its application are presented with the example works that apply SSL in these areas.

In [33], SSL is utilised for automatic prosodic event detection. Prosodic phenomena include variations in fundamental frequency, timing variations, changes in intensity of particular syllables and so forth. This information is crucial when solving automatic spoken language processing tasks. However, to manually annotate prosodic information is expensive and time-consuming. The work proposes applying the co-training algorithm, where one view represents acoustic information and the other one is responsible for lexical-syntactic part. The following prosodic event detection tasks have been set: pitch accents, intonational phrase boundaries and break indices. The results show performance, similar to one that would be obtained if the same amount of data were fully labelled (see Table 2.3, where results are shown for pitch accents, intonational phrase boundaries and break indices, respectively).

Table 2.3. *Selected audio-related works that utilise SSL.*

Work	**Subject**	**Accuracy, %**		
		Supervised, full dataset	**Supervised, little data**	**SSL**
[33]	Prosodic event detection			
	pitch accents	82	69	80
	phrase boundaries	74	59	71
	break indices	77	62	75
[60]	Sound event classification	72	67	69
[59]	Unusual event detection	95	88	93

Semi-supervised techniques have been used for unusual event detection in [59], where an adapted HMM framework has been proposed. It utilises the available large amounts of annotated data corresponding to the usual events and trains for the unusual part in an unsupervised manner. The technique has been applied to both audio and visual data, and the results for audio data are presented in Table 2.3.

Another related area where SSL has been recently applied is sound event classification. In [60], the task of avoiding using prototypical sparse and small-scale databases for training a sound event classifier has been set.

Regarding the existent music-oriented applications of SSL, one could mention such work as [58], which deals with the problem of note onset detection. Bootstrapping techniques (having onset labels used as training data and the output — to further refine the alignment data) were used there to iteratively refine onset detection functions leading to improvement of the onset detection algorithm used with orchestral music.

Moreover, in [50], such SSL approach as manifold regularisation (which graph learning algorithms are a special case of) is applied for music genre classification. The results show improved accuracy when used with three timbre and rhythm features.

Additionally, SSL methods have been used for solving the problem of music artist style identification. The work [38] focuses on singer-songwriters and applies an iterative procedure that resembles co-training in such a way that two separate classifiers based on features related to acoustic sounds and lyrics are applied. Thereupon, the unlabelled data on whose classification they confidently agree is used along with the labelled samples to update the classifier.

2.5 Summary

This chapter has addressed several crucial concepts related to the pattern recognition theory in general, as well as SSL. Several schemes that enable semi-supervised techniques have been described. Furthermore, an overview of the area of musical instrument recognition has been presented both from a traditional and state of the art points of view, as well as several examples of SSL applied in audio-related areas.

With the presented background it is now possible to explore the specific methodology, which is applied to solve the problem of musical instrument classification by utilising the semi-supervised techniques. The implementation steps of such particular system are presented in the following chapter.

3. METHODOLOGY

This chapter discusses the details of the implementation of the building blocks for a particular semi-supervised pattern recognition system for musical instrument classification. It follows the steps of a generic pattern recognition system and consequently describes two particular algorithms utilised for semi-supervised training.

Several potential issues of the algorithms observed in the literature and during preliminary experiments are then described. The chapter concludes with the suggested ways of overcoming these complications.

3.1 Preprocessing

As a preprocessing step in the case of the described pattern recognition system, audio normalisation is applied:

$$s_{\text{out}}[n] = \frac{1}{\max_i |s_{\text{in}}[i]|} \cdot s_{\text{in}}[n] - \sum_{i=1}^{N} \frac{s_{\text{in}}[i]}{N}, \tag{3.1}$$

where s_{in} is the input digitised audio signal of length N samples, and s_{out} is the output signal. This operation normalises the amplitude of the signal as well as removes the DC offset. This eliminates the variations in the signal energy, resulting in a better generalisation property of the system. These variations might be introduced by numerous factors, such as difference in distances to the microphone, microphone gain etc.

It is worth mentioning, though, that in ideal case the following steps (namely, MFCCs extraction) are meant to be able to discard these variations (by discarding the zeroth coefficient of the cepstrum), in which case the current step would be obsolete. However, the normalisation by discarding the zeroth coefficient of the MFCCs is performed within a different time range, namely, within the frame. Audio normalisation is, nevertheless, applied in order to obtain a more consistent and standardised data and facilitate the possible additional features that may require this.

3.2 Feature extraction

As features, the static and delta MFCCs (see Section 2.2.2) are utilised for representing the timbre of the musical instruments. This choice is motivated by the fact that MFCCs are most commonly used and have shown to be a quite robust method of characterizing the amplitude spectrum [18], which corresponds to the way the human auditory system processes audio.

Firstly, a high-pass filter $1 - 0.97z^{-1}$ has been applied to the signals of sampling frequency 44.1 kHz for pre-emphasis purposes. Secondly, frame blocking is performed with the aid of Hamming window of length 20 ms with 50% overlap. The window values in the temporal domain, with which the audio signal $s(n),\ n = 1 \dots N$ within a frame of length N is multiplied, are described as

$$w(n) = 0.54 - 0.46 \cdot \cos \frac{2\pi n}{N-1}. \tag{3.2}$$

To the obtained frames all the subsequent operations of the MFCCs extraction procedure are applied. For the implementation details see Section 2.2.2. The parameters of the applied MFCCs extraction are presented in Table 3.1.

Table 3.1. *Parameters of MFCCs feature extraction.*

Parameter	**Value**
Window length	20 ms
Window overlap	50%
Number of MFCC coefficients	16 excluding the zeroth
Total number of mel filters	40

Consequently, a feature normalisation is performed. It is widely used in various practical situations [52, p. 263] in order to avoid having particular features dominating over the others if those tend to have large values. This way, each feature contributes proportionally to the training as well as classification.

Feature normalisation starts with computing the scaling factors (namely, mean and standard deviation) for every k-th element of the feature vector over the whole training set:

$$\bar{x}_k = \frac{1}{N} \sum_{i=1}^{L+U} x_{ik}, \tag{3.3}$$

$$\sigma_k^2 = \frac{1}{N-1} \sum_{i=1}^{L+U} (x_{ik} - \bar{x}_k)^2, \tag{3.4}$$

where x_k denotes the k-th element of a feature vector and N denotes the total number of feature vectors in the training dataset. In the subsequent stages of training and testing these values are used for scaling each particular coefficient in the following way:

$$\hat{x}_{ik} = \frac{x_{ik} - \bar{x}_k}{\sigma_k}. \tag{3.5}$$

The applied method is linear, which arises from the assumption that the data is evenly distributed around the mean. One could, however, utilise a nonlinear mapping function as well, such as logarithm, depending on the nature of data in a particular application.

3.3 Recogniser

The implemented system performs supervised as well as semi-supervised learning, and both scenarios utilise mixture models, namely GMMs. For the theoretical details of this process see Section 2.3.2. For each class, the feature vectors obtained from the labelled data combined into a single matrix are used to train the GMM, i.e., to estimate the parameters of the mixture model that best explains these feature vectors. The EM algorithm is applied for this purpose. The parameters of the GMM calculation are presented in Table 3.2. The models obtained during the training stage are class-wise fit into each frame of each test instance to be classified producing log likelihoods, which are subsequently summed over the frames of the test instance. Thereupon, the label of the class whose model has produced the highest log-likelihood is assigned to that instance.

Table 3.2. *Parameters of GMM calculation.*

Parameter	**Value**
Number of mixture component densities	16
Maximum number of supervised EM-iterations	60

In the supervised case, the training ends at the point when the GMMs are obtained based on the labelled training data are consequently used to classify the unseen data from the testing set. The semi-supervised training scenario, however, continues by incorporating the unlabelled data and learning the new GMMs with the aid of EM-based algorithms. These are introduced in the upcoming sections.

3.4 Training algorithms

This section addresses the training stage of the developed system by describing the conventional supervised EM-algorithm, used for the supervised training scenario, as well as its semi-supervised versions. One of these algorithms is further extended by introducing the one-class-at-a-time training and labelled data weighting approaches for the simplified convergence criterion.

3.4.1 The EM algorithm

The EM algorithm [15] is a widely accepted tool of statistical analysis. The algorithm facilitates computing MLEs of the model parameters when treating observations as incomplete data. The term *incomplete data* refers to so-called *hidden*, or, more generally, *latent variables*, i.e., unobserved variables that are inferred from other, observable variables in a statistical model. In the case of mixture models, it is unknown which i of the L samples originates from which k of K Gaussians. The latent variables corresponding to the data points are incorporated in such a way that they specify the mixture components from which the data points originate [29].

The EM algorithm is used when obtaining direct equations for the model is impossible due to the requirement of both parameters and the values of the latent variable for the solution. It addresses the problem by assigning some initial values to the parameters and then iteratively performing two steps: expectation and maximisation. During the expectation step, the algorithm employs current estimates of the model parameters in order to find an expectation function for the log-likelihood, whereas the maximisation step recomputes the model parameters based on the obtained expected log-likelihood function.

An example of EM algorithm for learning parameters of GMM [28] is presented in Algorithm 3.1. It operates on a variable γ_{ik}, which is the estimate of the probability that the ith training sample originates from the kth mixture component:

$$\gamma_{ik} = \frac{w_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} w_{k'} \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})}, \tag{3.6}$$

where $i = 1, \ldots, L$, $k = 1, \ldots, K$ and K is the number of components in the mixture model. Additionally, for each Gaussian its total weight across all samples is defined as

$$n_k = \sum_{i=1}^{L} \gamma_{ik}. \tag{3.7}$$

In the presented case, the weights w_k of the mixture components are initialised to be equal, their means are assigned to the random samples from the training data and the covariances are initialised to identity matrices. However, a better initialisation can be achieved by performing some clustering beforehand to obtain an initial value for γ_{ik} [28]).

In the maximisation step, these variables are used to recompute for the next iteration of the algorithm the model parameters estimate $\hat{\boldsymbol{\theta}^{(t+1)}}$, consisting of weights

$$w_k^{(t+1)} = \frac{n_k^{(t)}}{\sum_{k'=1}^{K} n_{k'}^{(t)}}, \tag{3.8}$$

mean vectors

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{L} \gamma_{ik}^{(t)} \mathbf{x}_i \tag{3.9}$$

and covariance matrices

$$\boldsymbol{\Sigma}_k^{(t+1)} = \frac{1}{n_k^{(t)}} \sum_{i=1}^{L} \gamma_{ik}^{(t)} \left(\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)}\right)^T \tag{3.10}$$

for each k of the K mixture components.

The algorithm iterates until the convergence, which is often indicated by a sufficiently small change in the overall log-likelihood computed for all data points after each iteration:

$$\log p(\mathcal{D}|\hat{\boldsymbol{\theta}}^{(t+1)}) = \frac{1}{L} \sum_{i=1}^{L} \log \left(\sum_{k=1}^{K} w_k^{(t+1)} \mathcal{N}\left(\mathbf{x}_i \middle| \boldsymbol{\mu}_k^{(t+1)}, \boldsymbol{\Sigma}_k^{(t+1)}\right) \right). \tag{3.11}$$

```
Input: labelled data $S^l$.
Set $t = 0$.
[Initial M-step] for $k = 1, \dots, K$ do
    $w_k^{(0)} = \frac{1}{K}$,
    $\boldsymbol{\mu}_k^{(0)}$ as a random sample of $S^l$,
    $\boldsymbol{\Sigma}_k^{(0)} = \mathbf{I}_K$.
end
repeat
    [E-step] for $i = 1, \dots, L$ do
        for $k = 1, \dots, K$ do
            Compute $\gamma_{ik}^{(m)}$ (Eq. 3.6) and $n_k^{(m)}$ (Eq. 3.7).
        end
    end
    [M-step] for $k = 1, \dots, K$ do
        Based on obtained $\gamma_{ik}^{(m)}$ and $n_k^{(m)}$, compute $\hat{\boldsymbol{\theta}}^{(t+1)}$ consisting of
        $w_k^{(t+1)}$ (Eq. 3.8),
        $\boldsymbol{\mu}_k^{(t+1)}$ (Eq. 3.9) and
        $\boldsymbol{\Sigma}_k^{(t+1)}$ (Eq. 3.10).
    end
    Set $t = t + 1$.
until convergence (see Eq. 3.11).
Output: $\hat{\boldsymbol{\theta}}^{(t)}$.
```

Algorithm 3.1. *The EM algorithm for GMM.*

3.4.2 Extending EM for SSL

As examples of algorithms used for SSL with mixture models the *iterative* (Algorithm 3.2) and *incremental* (Algorithm 3.3) EM-based algorithms [44] are presented. They operate on the overall training dataset S consisting of the labelled S^l $(i = 1, \dots, L)$ and unlabelled S^u $(i = L+1, \dots, L+U)$ subsets, where L and U are the numbers of labelled and unlabelled samples, respectively. Both these algorithms incorporate the previously presented EM-algorithm by training based on the labelled instances only in order to obtain the initial model parameters estimate $\hat{\boldsymbol{\theta}}^{(0)}$. Thereupon, they incorporate unlabelled data in such a manner that the expected values of the hidden variable z_{ij} (see Equation 3.12) are used to estimate a "hard" labelling for the unlabelled examples at each step [44]. The hidden variable z_{ij} is defined for all $j = 1, \dots, M$ class indices (where M is the total number of classes) and for all training samples $\mathbf{x}_i$, $i = 1, \dots, L, L+1, \dots, L+U$ as

$$z_{ij} = \begin{cases} 1 & \text{if } y_i = c_j \\ 0 & \text{otherwise} \end{cases}, \tag{3.12}$$

where y_i is the actual label in the case of labelled data and the obtained classification result based on the models of a previous iteration in the case of unlabelled data. The label of a class j is represented by c_j.

The two presented algorithms differ in the way the hard labelling is handled. In the iterative version, the hard labels are assigned to each initially unlabelled instance based on the maximum value of the hidden variable amongst classes all at once, and can be further refined with additional iterations. In the incremental version, the expected values of the hidden variable that have a maximum value for a particular class are grouped together within this class, and the "most certain" one creates a hard label for the corresponding instance, relocating it from the unlabelled into labelled pool. During one iteration of the algorithm, one hard label for each class is assigned. For the graphical illustration of the training process within the iterative and incremental algorithms, see Figure 3.1 and Figure 3.2, respectively.

There is an apparent issue inherent in the incremental algorithm, namely is its computational inefficiency. At each iteration only one data sample (which produces the highest log-likelihood) is transferred from the unlabelled to the labelled pool. An obvious solution is to increase the number of samples moved at each iteration. To find an optimal size of such subset is an empirical problem, which could be also investigated during the experimental stage, provided the algorithm shows promising potential.

```
**Input**: labelled data $S^l$, unlabelled data $S^u$.
Set $t = 0$.
[Initial M-step] Initialise $\hat{\boldsymbol{\theta}}^{(0)} = \arg\max_{\boldsymbol{\theta}} P(S^l|\boldsymbol{\theta})$.
repeat
    [E-step] Set $\hat{\mathbf{z}}^{(t+1)} = E[\mathbf{z}|S; \hat{\boldsymbol{\theta}}^{(t)}]$.
    for $i = L+1, \ldots, L+U$ do
        Set $j^* = \arg\max_j \hat{z}_{ij}^{(t+1)}$.
        Set $\hat{z}_{ij}^{\text{hard}(t+1)} = \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases}, \; j = 1, \ldots, M$.
    end
    [M-step] Set $\hat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} P(S, \hat{\mathbf{z}}^{\text{hard}(t+1)}|\boldsymbol{\theta})$.
    Set $t = t + 1$.
until convergence.
Output: $\hat{\boldsymbol{\theta}}^{(t)}$.
```

Algorithm 3.2. *The iterative EM-based algorithm.*

```
Input: labelled data $S^l$, unlabelled data $S^u$.
Set $t = 0$.
[Initial M-step] Initialise $\hat{\boldsymbol{\theta}}^{(0)} = \arg\max_{\boldsymbol{\theta}} P(S^l|\boldsymbol{\theta})$.
while $S^u \neq \emptyset$ do
    [E-step] Set $\hat{\mathbf{z}}^{(t+1)} = E[\mathbf{z}|S; \hat{\boldsymbol{\theta}}^{(t)}]$.
    for $j = 1, \ldots, M$ do
        Set $S_j^u = \{\mathbf{x}_i \in S^u : \hat{z}_{ij}^{(t+1)} > \hat{z}_{ij'}^{(t+1)} \; \forall \; j' \neq j\}$.
        Set $i^* = \arg\max_{\{i : \mathbf{x}_i \in S_j^u\}} \hat{z}_{ij}^{(t+1)}$.
        Set $S^l = S^l \cup \{(\mathbf{x}_{i^*}, c_j)\}$.
        Set $S^u = S^u \setminus \{\mathbf{x}_{i^*}\}$.
    end
    [M-step] Set $\hat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} P(S^l|\boldsymbol{\theta})$.
    Set $t = t + 1$.
end
Output: $\hat{\boldsymbol{\theta}}^{(t)}$.
```

Algorithm 3.3. *The incremental EM-based algorithm.*

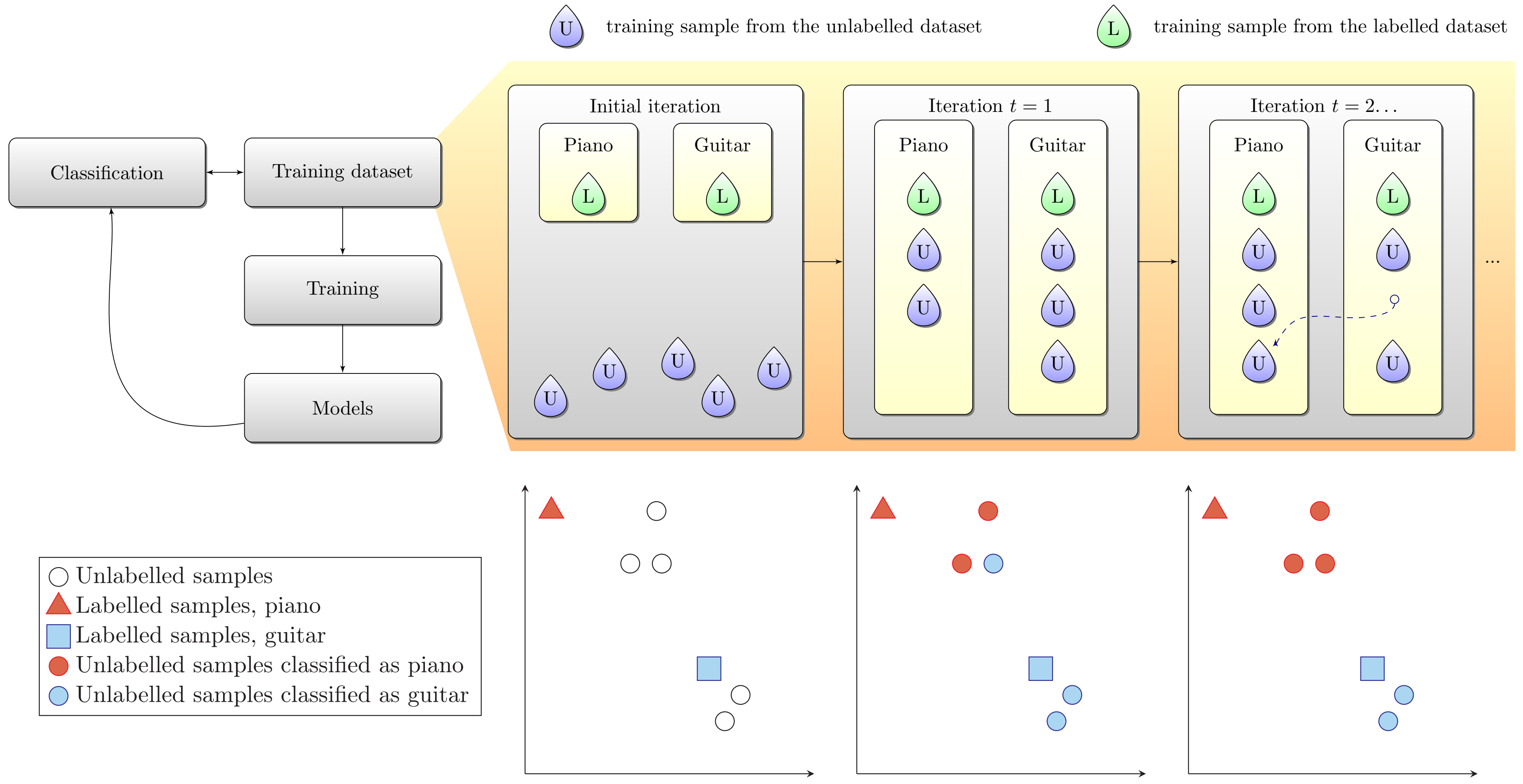


Figure 3.1. *Schematic and feature space representation of the training stage with three iterations of the iterative EM-based algorithm on an example of a two-instrument classification scenario with two-dimensional features.*

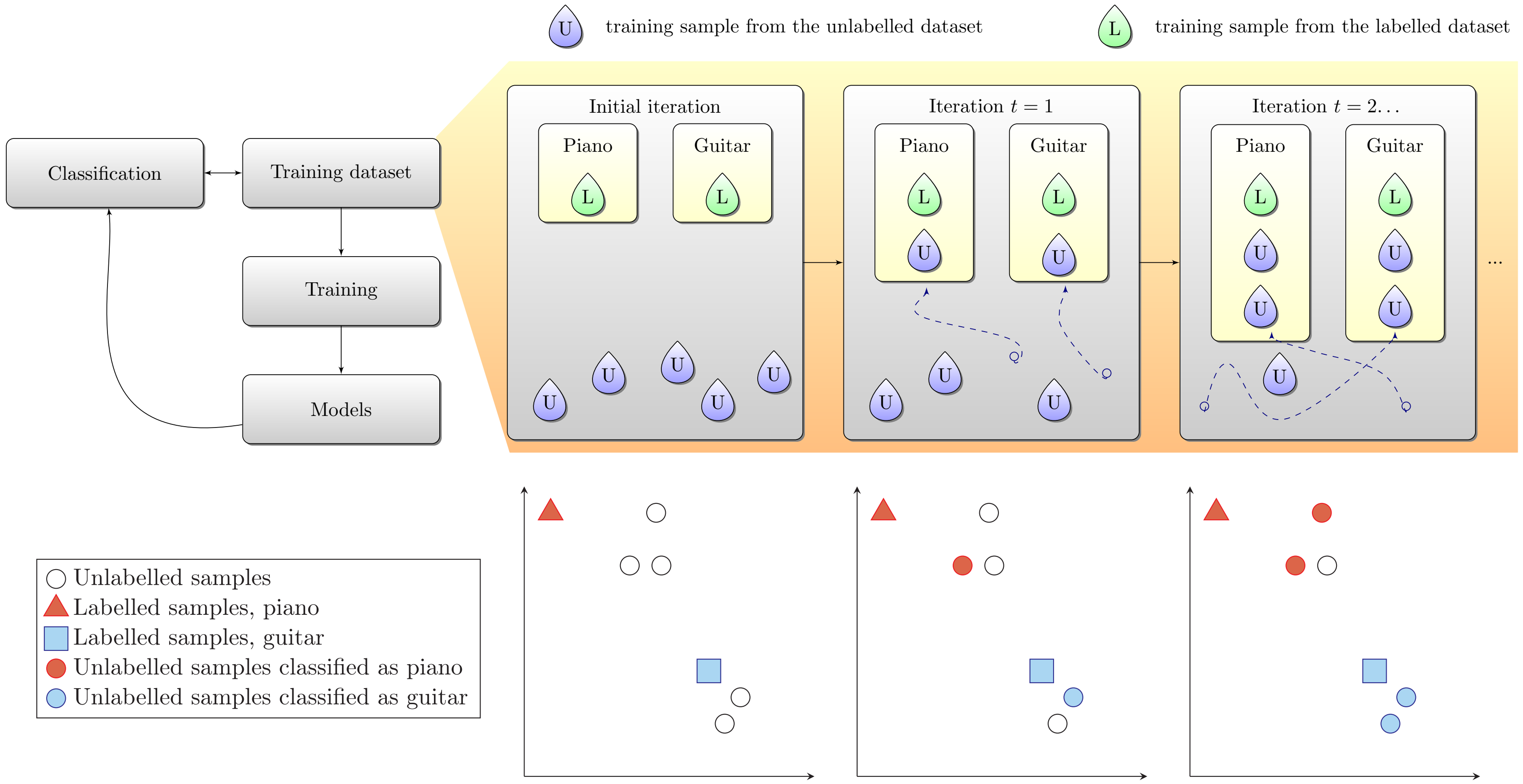


Figure 3.2. *Schematic and feature space representation of the training stage with three iterations of the incremental EM-based algorithm on an example of a two-instrument classification scenario with two-dimensional features.*

3.4.3 Decision on convergence

In [44], the experiments were conducted using the models obtained after a single iteration of the EM-based iterative algorithm. However, the iterations may continue, and preliminary experiments with the system described in the current work have shown that running the algorithm for several iterations positively contributes to the resulting accuracy. There is, however, a need of a rule that would facilitate a decision to terminate the algorithm.

Several parameters whose relative change of value could manifest the convergence of the algorithm are considered. The artificial way, which is not applicable in a real-world scenario but, rather, only in a development phase, is to set a threshold to the increase in classification accuracy of the validation data, which is assumed unlabelled during training, but the labels are nevertheless used to check the correctness of the intermediate labels obtained on the semi-supervised stage.

A more sensible way to check for the convergence would be to observe the value of the sum of the maximum log-likelihood values over all the unlabelled samples, i.e., the sum of the highest log-likelihoods across the classes for each unlabelled sample. The complication with such approach is the requirement of a threshold that is applied on the values of a rather unpredictable range depending on the number of instruments in the set, their mutual confusability, the labelled-unlabelled data size ratio and so forth. Alternatively, one could observe the relative change of this sum of the most certain log-likelihoods thus normalising the range of possible changes. However, the validity of normalisation in this case might be questionable due to the logarithmic nature of the values.

A third option, which appears most feasible, is to check the total number of labels that *change* at each iteration. One could utilise the matrix of the hidden variables $\hat{z}_{ij}^{\text{hard}}$ provided by the algorithm (see Equation 3.12) to check for a label change count (LCC):

$$\mathrm{LCC}^{(t+1)} = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\left|\hat{z}_{ij}^{\text{hard}(t+1)} - \hat{z}_{ij}^{\text{hard}(t)}\right|,\ t = 1, 2, \ldots, \tag{3.13}$$

where $\mathrm{LCC}^{(t+1)}$ — label change count at iteration $t+1$. By normalising this value by the first label change $\mathrm{LCC}^{(2)}$ (which happens on the second iteration since on the first iteration all the labels change from non-existent to some values), one can obtain a label change rate (LCR):

$$\mathrm{LCR}^{(t+1)} = \frac{\mathrm{LCC}^{(t+1)}}{\mathrm{LCC}^{(2)}} \cdot 100\%,\ t = 2, 3, \ldots. \tag{3.14}$$

The possible local minima in the LCR curve, which may occur at the earlier stages of the algorithm, can trigger the termination earlier than the actual convergence. These may be minimised with the aid of a moving average filter based on the M preceding values, where M equals the number of classes.

This approach appears reasonable as it is expected to yield low values of label change rate when the obtained models are sufficiently certain. In that case, they are not expected to be affected to a high extent in the following iterations, which suggests that the algorithm may be terminated.

Concerning the incremental algorithm, such decision is generally not required. The algorithm terminates when all data from the unlabelled pool is moved into the labelled one.

3.4.4 Labelled data weighting

It has been noted in [44] that the presented EM-based algorithms for SSL improve the performance in case the initial labelled data size is relatively low. However, the difference in the advantages of incorporating large and small amounts of unlabelled data has been reported relatively insignificant [44], which is explained by the fact that with increasing unlabelled data the parameter estimates depend very little on the labelled data and reliable class information.

Such issue appears to be a general property of these algorithms and is therefore worth addressing in this work. As stated in [46], when the mixture model assumptions are not true, the natural clustering of the unlabelled data may produce mixture components that are not in correspondence with the class labels, which is particularly apparent when the size of the labelled set is large enough to obtain reasonably good parameter estimates for the classifier, yet more unlabelled samples still overwhelm parameter estimation.

To overcome this issue, it has been suggested to de-weight the contribution of the unlabelled data in order to control the extent of unsupervised clustering. It can be achieved by scaling the contribution from unlabelled data in the log-likelihood (see Equation 2.17) by a weight $\lambda \in [0, 1]$ [46], [62, pp. 29–30]:

$$\log p(\mathcal{D}|\theta) = \sum_{i=1}^{L} \log p(y_i|\theta) p(\mathbf{x}_i|y_i, \theta) + \lambda \sum_{i=L+1}^{L+U} \log p(\mathbf{x}_i|\theta). \qquad (3.15)$$

The obtained log-likelihood would be then used in estimating the new labels. Such general method enables a rather controllable de-weighting. However, there exists a somewhat coarser but simplified way of de-weighting the contribution of the unlabelled data.

The essence of the simplified de-weighting is in replicating the labelled data several times so that the labelled and unlabelled data set sizes are initially equal. During the subsequent iterations the replication factor is gradually decreased. This operation is referred to in these extended algorithms (see Algorithm 3.4 and Algorithm 3.5) as $S^l = \omega(t)\Diamond S^l$, where S^l represents the labelled training dataset and $\omega(t)$ is a decreasing weighting function of iteration index t. This way, it is expected that more significance is given to the parameters obtained from the data with a priori correct labels, simultaneously incorporating the advantage of diversity of the unlabelled data to tailor the models on a meticulous level. Although a degree of flexibility and control of the weighting factors is limited, the implementation of such approach does not require complicated modifications of the algorithms as such and is thus more straightforward. As a result, the certainty of the models that are obtained from labelled data is emphasised, and the algorithm is more likely to rely on the certainly labelled data. This way, smoother transition between the models and reduced oscillations of the accuracy curve are expected.

3.4.5 One-class-at-a-time training

Another issue of the iterative algorithm observed during the preliminary tests is that the resulting classification accuracy is oscillating along the axis that represents the number of iterations (see Figure 3.3). The basic version of the algorithm (Algorithm 3.2) contemplates retraining of all the models at each iteration. It appears apparent that at any retraining stage there exists a classification error. If some data point were previously classified correctly leading to a somewhat correct model of its actual class, the classification error introduced at a later point would mean degradation of the models of two classes: the actual origin of the data point and the misclassification result.

To overcome this issue, it is proposed to apply a so-called *one-class-at-a time* approach, which enforces the previously obtained models of one class not to change while training another class. The essence of the approach is that at each iteration the models of only one class are retrained, while the others remain fixed. This way, a smoother transition between the models is expected, thus resulting in fewer local peaks in the accuracy curve, which is expected to become smoother. This benefits also to the decision on convergence problem: since the oscillations are minimised it is then safer to chose an arbitrary iteration index with some degree of certainty that it does not yield a local minimum. The iterative algorithm with the proposed extensions is presented in Algorithm 3.4 and the obtained preliminary results are compared in Figure 3.4, where the term *macroiteration* stands for a set of iterations required to retrain the models of all classes once.

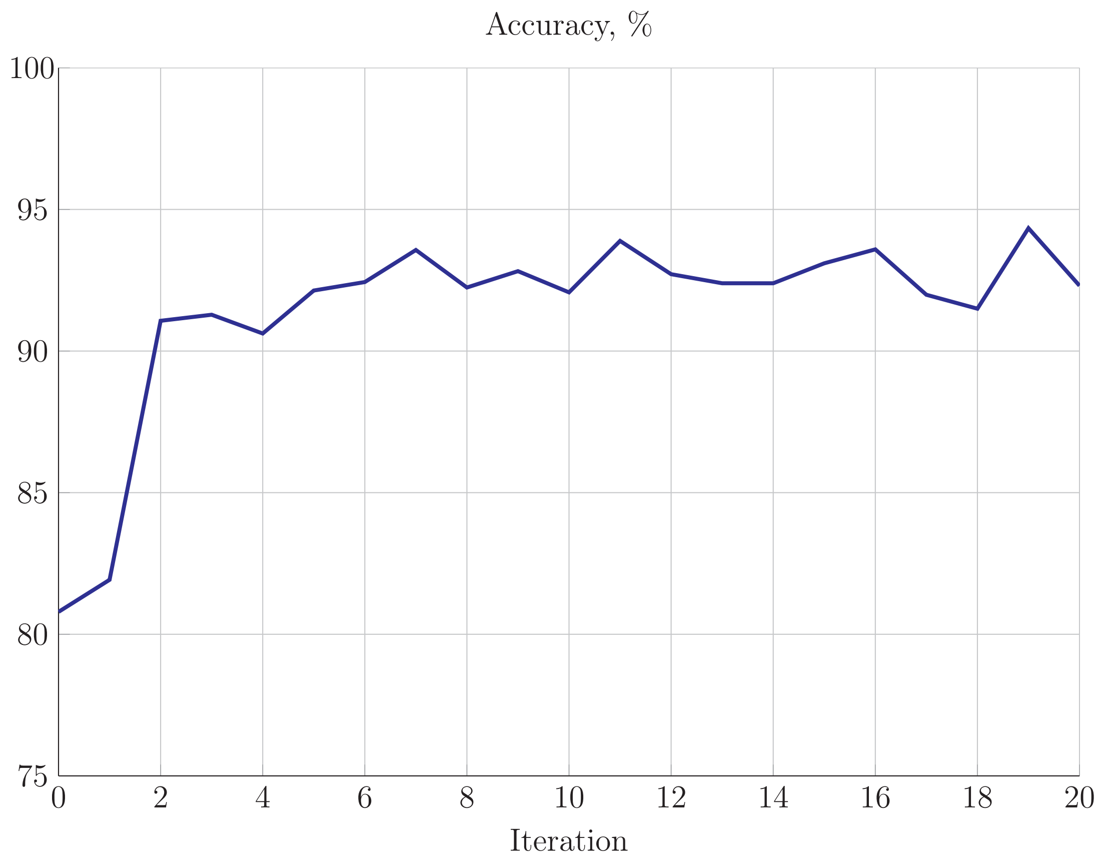


Figure 3.3. *An oscillating behaviour of the accuracy curve as a function of iteration index observed during the preliminary tests.*

An analogous one-class-at-a-time training is applied to the incremental EM-based algorithm for the same purpose. Namely, only a fragment of the unlabelled data, which obtains labels of a specific class at the current iteration with the highest confidence is used to retrain the models of that class, whereas the rest of the models remain fixed. Furthermore, the preliminary tests have not shown any improvement of the initial accuracy whatsoever when utilising the initial incremental algorithm. Therefore, in the evaluation stage, the one-class-at-a-time modification is included in the algorithm by default.

The validity of such decision is justified by the fact that ideally the algorithm is to be performed with as many iterations as there are training instances in the unlabelled dataset. Therefore, only one class would obtain new data at each iteration. As it does not appear computationally realistic to perform that many iterations, a higher number of training instances is moved from the unlabelled to the labelled set at each iteration (as suggested in Section 3.4.2). Employing the one-class-at-a-time is a way of partially simulating the idealistic scenario.

```
**Input**: labelled data $S^l$, unlabelled data $S^u$.
Set $t = 0,\ t^* = 0$.
[Initial M] Initialise $\hat{\boldsymbol{\theta}}^{(0)} = \arg\max_{\boldsymbol{\theta}} P(S^l|\boldsymbol{\theta})$.
repeat
    Set $S^l = \omega(t^*)\diamondsuit S^l$.
    for $j^\dagger = 1, \ldots, M$ do
        [E] Set $\hat{\mathbf{z}}^{(t+1)} = E[\mathbf{z}|S; \hat{\boldsymbol{\theta}}^{(t)}]$.
        for $i = L+1, \ldots, L+U$ do
            Set $j^* = \arg\max_j \hat{z}_{ij}^{(t+1)}$.
            if $j^\dagger = j^*$ then
                Set $\hat{z}_{ij,j=1,\ldots,M}^{\text{hard}(t+1)} = \begin{cases} 1 & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases}$.
                Set $\text{LCC}^{(t+1)} = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\left|\hat{z}_{ij}^{\text{hard}(t+1)} - \hat{z}_{ij}^{\text{hard}(t)}\right|$.
                Set $\text{LCR}^{(t+1)} = \frac{\text{LCC}^{(t+1)}}{\text{LCC}^{(1)}}$.
            else
                Set $\hat{z}_{ij,j=1,\ldots,M}^{\text{hard}(t+1)} = \hat{z}_{ij,j=1,\ldots,M}^{\text{hard}(t)}$.
            end
        end
        [M] Set $\hat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} P(S, \hat{\mathbf{z}}^{\text{hard}(t+1)}|\boldsymbol{\theta})$.
        Set $t = t + 1$.
    end
    Set $t^* = t^* + 1$.
until $LCR^{(t)} < threshold$.
**Output**: $\hat{\boldsymbol{\theta}}^{(t)}$.
```

Algorithm 3.4*. *The iterative EM-based algorithm for SSL with the proposed extensions highlighted.

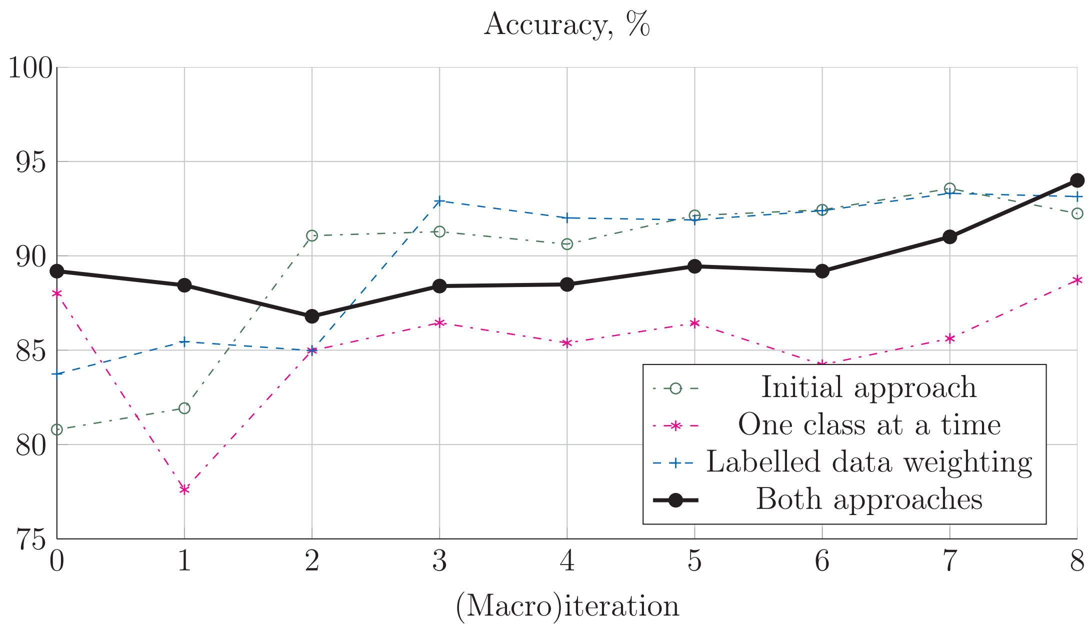


Figure 3.4. A preliminary comparison of the proposed extensions to the iterative EM-based algorithm, where macroiteration refers to the set of iterations performed to once retrain models of one class.

```
**Input**: labelled data $S^l$, unlabelled data $S^u$.
Set $t = 0$.
[Initial M-step] Initialise $\hat{\boldsymbol{\theta}}^{(0)} = \arg\max_{\boldsymbol{\theta}} P(S^l|\boldsymbol{\theta})$.
**while** $S^u \neq \emptyset$ **do**
    Set $S^l = \omega(t)\Diamond S^l$.
    [E-step] Set $\hat{\mathbf{z}}^{(t+1)} = E[\mathbf{z}|S;\hat{\boldsymbol{\theta}}^{(t)}]$.
    **for** $j = 1, \ldots, M$ **do**
        Set $S^u_j = \{\mathbf{x}_i \in S^u : \hat{z}^{(t+1)}_{ij} > \hat{z}^{(t+1)}_{ij'} \;\forall\; j' \neq j\}$.
        Set $i^* = \arg\max_{\{i:\mathbf{x}_i \in S^u_j\}} \hat{z}^{(t+1)}_{ij}$.
        Set $S^l = S^l \cup \{(\mathbf{x}_{i^*}, c_j)\}$.
        Set $S^u = S^u \setminus \{\mathbf{x}_{i^*}\}$.
        [M-step] Set $\hat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} P(S^l|\boldsymbol{\theta})$.
        Set $t = t + 1$.
    **end**
**end**
**Output**: $\hat{\boldsymbol{\theta}}^{(t)}$.
```

Algorithm 3.5. *The incremental EM-based algorithm for SSL with the proposed extensions highlighted.*

4. EVALUATION AND DISCUSSION

In this chapter, firstly, the details of the setup for evaluating the effectiveness of the proposed methods are described. It starts with presenting the database and the instruments used for evaluation, followed by a set of conditions specified for a particular test session.

Consecutively, the results obtained under these conditions are presented along with a discussion. The observed problems are addressed by employing the proposed modifications to the algorithms, whose contribution into the resulting performance is also evaluated and discussed.

4.1 Database

This work utilises the Musical Instrument Sound Database of the RWC Music Database [25] consisting of 50 musical instruments, each represented by three variations, or instrument instances. The variations stand for different instrument manufacturers and musicians.

Within each variation there is a subdivision into different playing styles (i.e., different manners of producing sound on a particular instrument, for instance, staccato and pedal on a piano), which are in their turn divided into a set of different dynamic levels (piano, mezzo, forte). Finally, each playing style is represented by a recording of separate notes, occupying the whole instrument range with a step of a semitone. All the recordings are made with sampling frequency 44.1 kHz and bit depth 16 bit.

4.2 Choice of instruments

Two sets of instruments are chosen for evaluating the systems performance. The first set consists of four instruments (Table 4.1). It is intended to represent an “easy” scenario, where quite a limited number of distinguishable classes is used. Therefore, rather certain models are expected to be produced already at the initial training, where only a small part of database is used.

The second set consists of ten instruments (Table 4.2) and is intended to more closely resemble a real-life application scenario. The choice of instruments is influenced by the requirement of a sufficiently high number of examples and their adequately consistent representation in the database. That is, at least a certain number of note recordings (in this case, about 270 notes, i.e. at least 90 notes of each

Table 4.1. *List of instruments and number of recordings of the notes used in the smaller set.*

Instrument	**# notes**
Acoustic Guitar	702
Electric Guitar	702
Tuba	270
Bassoon	360
Total	2 212

Table 4.2. *List of instruments and number of recordings of the notes used in the larger set.*

Instrument	**# notes**
Pianoforte	792
Classic Guitar	702
Electric Guitar	702
Electric Bass	507
Trombone	278
Tuba	270
Horn	288
Bassoon	360
Clarinet	360
Banjo	941
Total	5 200

instance) should be present in the database for the class to be presumed consistently represented. Amongst the selected instruments, some are mutually confusable. This might introduce complications, which the semi-supervised technique is expected to be able to overcome.

4.3 Training and test datasets acquisition

The datasets are divided into the three following groups: the training subset with labelled and unlabelled data and the testing subset. The details of such division are described bellow.

The effectiveness of suggested methods is evaluated by synthesising a scenario with the labelled and unlabelled datasets. Since the available databases are fully annotated, an artificial separation of their part into an “unlabelled” set is conducted by ignoring its labels. Because of the available freedom to choose the desired set sizes, there is an interest of observing the dependency between the labelled-to-unlabelled ratio and the resulting accuracy. This ratio is initially set to 10/90 for all the experiments if not stated otherwise.

Such artificial division, however, may introduce undesired overoptimistic results. In particular, the fact that the sets produced by the same instrument instance resemble each other much more, than the unlabelled and labelled data is expected to resemble each other in a real-life scenario, would lead to unrealistic evaluation results. In order to eliminate such effect, the labelled and unlabelled datasets are always acquired from different instrument instances. For the testing set, a separate instance is used for the similar reasons: to simulate the real-life scenario where one should not expect the data to be classified to resemble too much the data the classifier had been trained on.

Since the selected instruments are represented by at least three instances, a natural conclusion is to utilise each of these for a separate set, and in case there are more instances available, the extraneous data is used for the unlabelled set. However, the proportion between the labelled and unlabelled sets is always preserved such, that the set whose length (in notes) causes the desired proportion to exceed the given value is automatically truncated. The notes are mostly recorded in chromatic order, which may render the truncated datasets biased towards lower notes. For the purpose of eliminating such effect, the notes within each set are preliminarily randomised. Randomisation is also performed on the sequence of instrument instances within each class so that possible patterns following the ordering choice of the instances are also eliminated.

The performed randomisation, however, may introduce unstable results when a fixed case is needed to compare the effect of a particular parameter on the overall performance. Therefore, after all the randomisations have been done, the obtained sequences are saved for the future experiments.

4.4 Baseline results

The most significant benefit of applying semi-supervised techniques is that such performance is expected to be achieved that is close to the one obtained if all the data were labelled. In order to demonstrate the effectiveness of the suggested algorithms, there is a need therefore to be able to compare it to the fully supervised case. For the purpose of obtaining an upper bound for the accuracy, which is possible to achieve given the characteristics common to all evaluation scenarios, a baseline supervised test is conducted.

The baseline scenario includes the same preprocessing and feature extraction blocks, as well as the same mixture models for training, as those used in the semi-supervised approaches. It utilises the available datasets completely by incorporating the available labels for the data that is treated as unlabelled in the semi-supervised

Table 4.3. *Average classification accuracy across all classes in fully-supervised case.*

Instrument set size	**Recognition accuracy, %**
4 instruments	92.1
10 instruments	82.9

scenarios. The training, however, naturally stops after the first models are obtained, and those are utilised for the baseline evaluation. The baseline results obtained under the described conditions are displayed in Table 4.3.

The confusion matrices for the smaller and larger instrument sets are calculated as well. For their role in evaluation and calculating procedure see Section 2.1.3. The matrices are presented in Table 4.4 and Table 4.5 respectively.

Table 4.4. *Confusion matrix for the smaller instrument set in the fully-supervised case.*

	AcG	**ElG**	**Tu**	**Ba**
Acoustic Guitar	**97**	3	-	-
Electric Guitar	3	**97**	-	-
Tuba	-	-	**100**	-
Bassoon	32	-	-	**68**

Table 4.5. *Confusion matrix for the larger instrument set in the fully-supervised case.*

	Pi	**ClG**	**ElG**	**ElB**	**Tr**	**Tu**	**Ho**	**Bas**	**Cl**	**Ban**
Pianoforte	**92**	-	-	-	-	-	-	-	-	8
Classic Guitar	7	**74**	-	12	-	-	-	-	-	7
Electric Guitar	8	-	**91**	-	-	-	-	-	-	-
Electric Bass	-	24	-	**76**	-	-	-	-	-	-
Trombone	-	-	-	-	**96**	-	-	-	-	4
Tuba	-	-	-	4	-	**94**	-	1	-	-
Horn	1	-	-	5	1	-	**75**	-	2	15
Bassoon	3	2	1	1	5	-	3	**74**	3	8
Clarinet	-	1	-	1	1	-	-	-	**97**	-
Banjo	1	36	-	4	-	-	-	1	-	**58**

The fully-supervised four-instrument case shows quite satisfactory performance, thus justifying the adequacy of the selected feature extraction and mixture model parameters. The confusion matrix indicates that most of the instruments are well-separated with the exception of a bassoon, which is occasionally erroneously classified as the acoustic guitar, however, the opposite does not hold, and the models for the acoustic guitar are rather accurate. Such well-separated case is expected to require a small ratio of labelled-to-unlabelled data sizes when proceeding towards semi-supervised scenarios.

When applied to the larger instrument set, the described supervised system shows a somewhat decreased performance, which is supposedly caused by the fact that some of the instruments in the set are mutually confusable to some extent. This can be seen in the confusion matrix (Table 4.5). It demonstrates that in given feature extraction and training settings the models obtained for the instrument Electric Guitar, for instance, occasionally produce high likelihood values for the features extracted from the samples belonging to the instrument Electric Bass. Another example is the instrument Banjo, which is considered similar to Classic Guitar. Some unexpected similarities are also observed, for instance, between the instruments Horn and Banjo. Nevertheless, with the exception of the mentioned confusable classes, the overall performance is still rather satisfactory.

The supervised scenario incorporated in each of the evaluated semi-supervised algorithms is similar to the above-described baseline one with the exception of the size of the labelled dataset, which is reduced to the 15% of the whole training set. In order to facilitate a fair comparability of the approaches, all of which include this initial supervised step, it appears most logical to have this starting point fixed. However, even when the datasets are identical within the experiment sessions, the initial models have some degree of uncertainty.

The presence of such effect is apparent on an example of a set of 10 training and evaluation experiments performed with the same datasets of the 10-instrument case. The resulting average recognition accuracy has shown a noticeable level of variation: from 71.4% to 79.0% with the average value 75.2%. The reason behind these variations is the randomisation that occurs within the EM-algorithm (see Algorithm 3.1). This phenomenon is to be acknowledged when performing a comparative analysis of the presented algorithms.

4.5 Results with the semi-supervised algorithms

This section reveals the resulting performance evaluated for the two semi-supervised algorithms, namely, EM-based iterative and incremental algorithms. The details of their implementation were presented in Section 3.4.2.

Additionally, the section comprises of a discussion of certain undesired characteristics of their performance in general and in the particular classification task of interest, which were revealed during preliminary experiments and addressed in Section 3.4.5 and Section 3.4.4. Several proposed approaches to overcome those issues are evaluated.

4.5.1 Iterative EM-based algorithm

Initial approach

As a preliminary criterion for terminating the iterations the following rule of thumb is proposed:

$$\text{Number of iterations} = 5 \times \text{number of classes}. \tag{4.1}$$

Such dependency accounts for the situations with variable number of classes. It appears somehow apparent, that the more classes there are considered in a particular scenario, the more iterations it would require to acquire sufficiently stable models. The factor of 5 is obtained empirically, based on preliminary experiments on the both instrument sets, which have suggested that increasing its value does not produce significant difference in the resulting performance.

However, this rule is not sufficiently motivated to be applicable as a final convergence criterion, which is to be measured based on an estimate of the level of models' certainty. A refined criterion to be ultimately applied is to set a threshold to the label change rate (LCR, see 3.4.3), whereas the current coarse rule is used solely as an upper bound in order to ensure that no unexpected behaviour is present at the points after the estimated convergence.

The results of the evaluation on the smaller instrument set with regard to the number of iterations, which is performed three times, are presented in Figure 4.1. Such multiple evaluation is conducted with the purpose of exploring the effect of randomisation on the resulting performance level. In all three cases, the same choice of instrument instances for labelled, unlabelled and evaluation sets, respectively, is used, whereas the randomisation occurs when selecting the notes within each instance that are to be utilised. Furthermore, even the same feature vector sets utilised for a multiple training are not expected to yield identical performance due to the randomisation that occurs within the EM-algorithm (see Algorithm 3.1).

The obtained curves demonstrate a rather similar behaviour of the system's performance depending on the number of iterations. In all the cases, it improves quite noticeably during the first iterations, reaching a somewhat upper limit after 8 iterations, which corresponds to a double number of instruments in the model set. Then the average accuracy starts oscillating somewhat randomly, however, the changes are within a reasonable range of 1–3%.

The reason for these oscillations is most likely due to the models' uncertainty. It manifests itself in the fact that improving the models of one instrument influences the correctness of the models of another instrument due to the fact that some training instances are somewhat equally likely to have been produced by either of the classes. If such instance is located in the area of the feature space where these classes

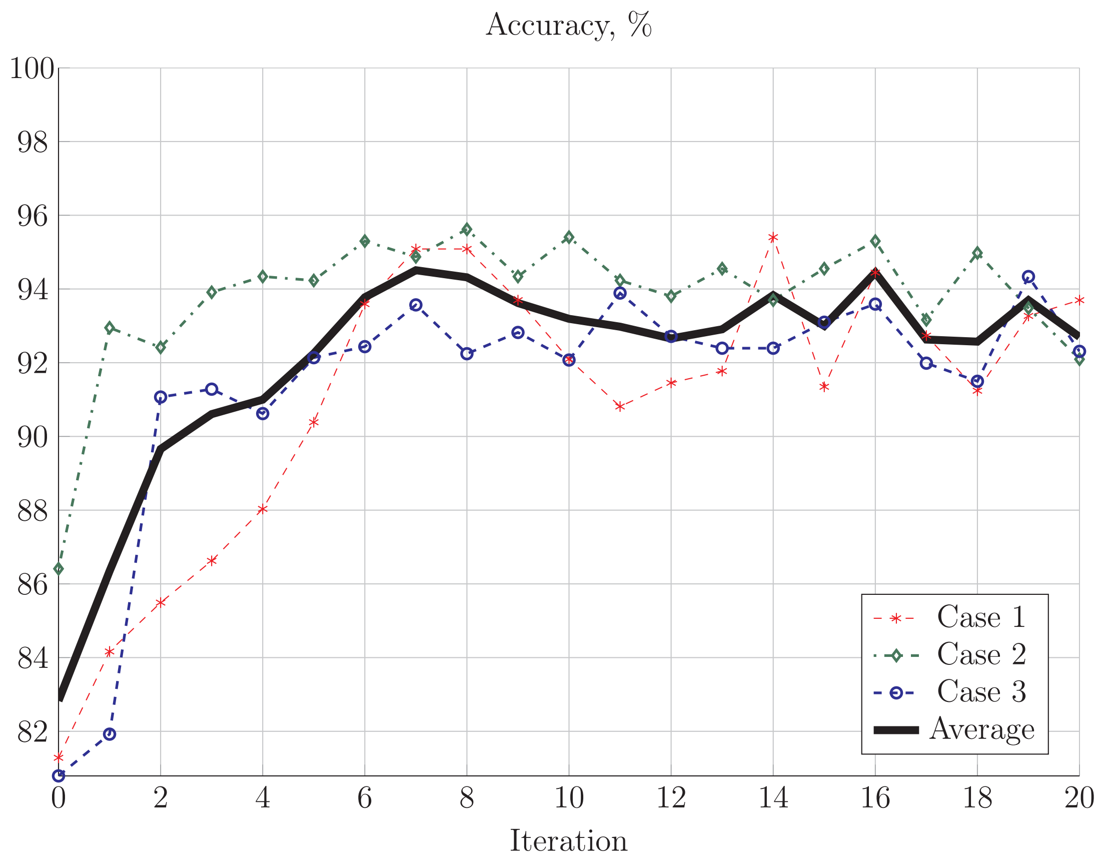
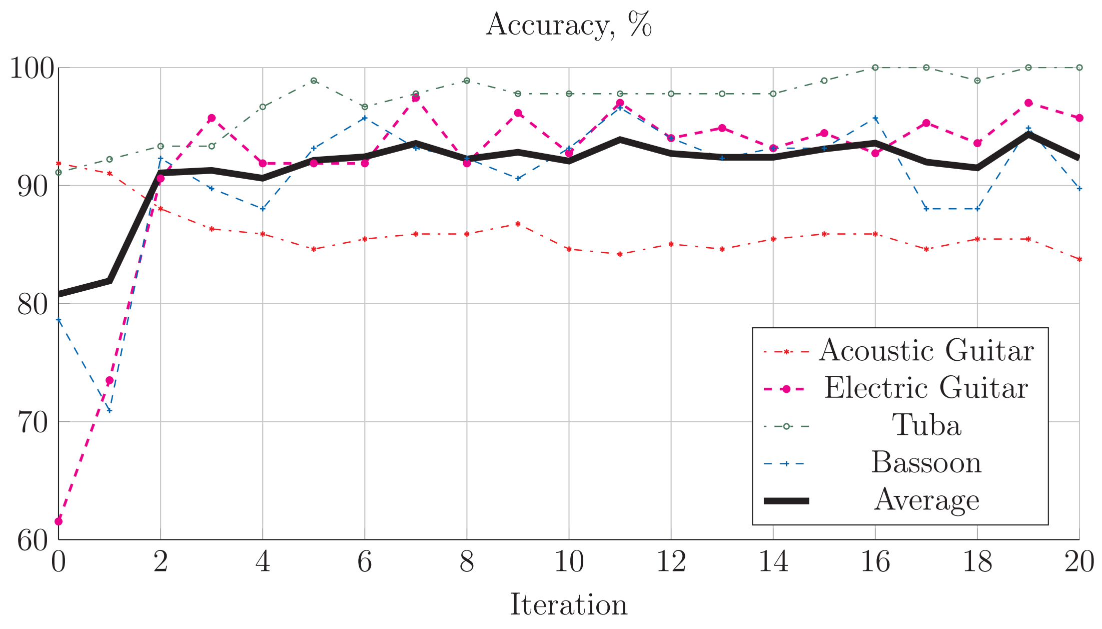


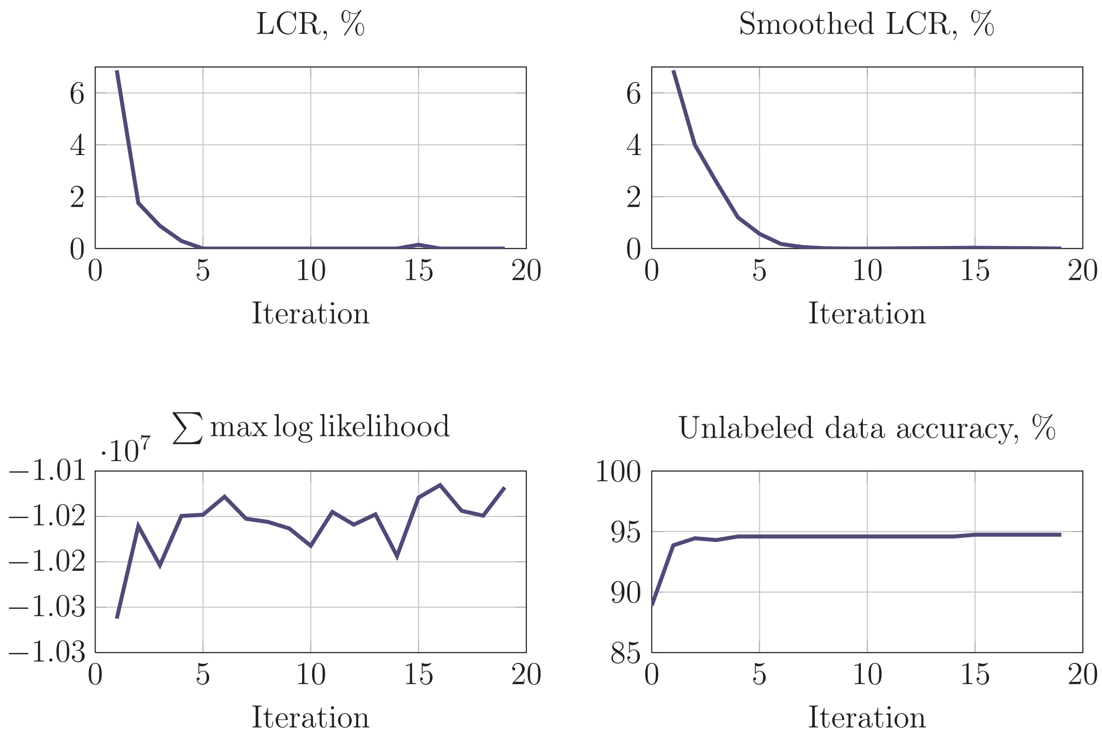
Figure 4.1. *Three randomisation cases and a mean value of the recognition rates with the smaller instrument set as a function of number of iterations of the original iterative EM-based SSL algorithm with labelled-to-unlabelled ratio 10/90.*

intersect, each iteration would slightly move the decision boundary, thus causing the change in the obtained class label. Such explanation can be ascertained by analysing Figure 4.2(a), in which the class-wise accuracy depending on the iteration index is presented. Indeed, the local maxima and minima in the resulting accuracy of the models of one instrument (for example, Bassoon at iterations 6 and 9 respectively) do occasionally correspond to the opposite peaks in the accuracy curve of another instrument (for example, Electric Guitar on the same iteration indices). If such overlap of the decision regions actually takes place, this manifests itself the Bayes error, which can not be eliminated with the given feature set (see Section 2.1.4). Therefore, one could expect that such oscillating behaviour may continue infinitely, and hence executing the iterations for arbitrarily long time with anticipation of reaching a stable point appears to be somewhat futile.

All the subsequent evaluation scenarios are characterised by a similar behaviour: the initial models are not as reliable as the ones obtained after several iterations, and the dependency retains its oscillating characteristic within a tolerably small range.



(a) Instrument-wise accuracies as functions of the number of iterations.



(b) Auxiliary information on the training progress as functions of the number of iterations.

Figure 4.2. *Instrument-wise accuracies (a) and auxiliary information on the training (b) of the original* iterative *EM-based SSL algorithm with a* smaller *instrument set.*

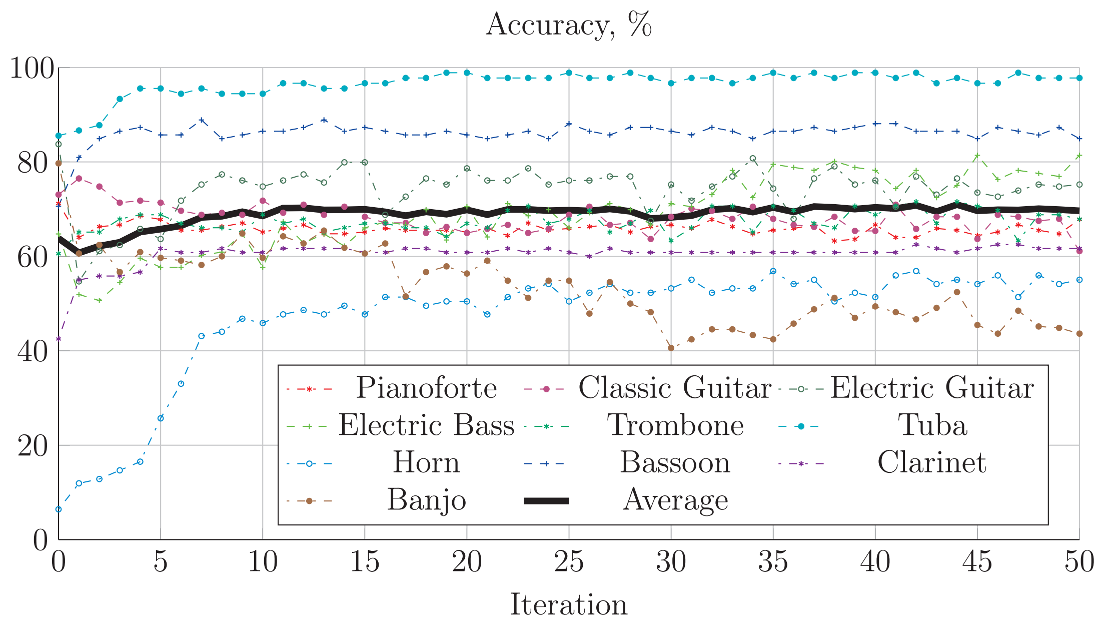
Such three-fold analysis is therefore omitted in the consecutive tests and the results are presented on single evaluation round cases.

The noticeable growth of the accuracy of the models obtained after several iterations of the algorithm on the smaller instrument set is present in each randomisation case (Figure 4.1), which could suggest an overly optimistic prognosis of the applied SSL technique's successfulness. However, one should bear in mind the simplicity of the considered instrument set: all the classes are rather distinguishable (see baseline tests, Section 4.4) and well-represented. Adding even one instrument that is not as well represented in the labelled training set (such as Harpsichord with only feature vectors inferred from 6 notes that can be utilised while maintaining the labelled-to-unlabelled ratio 10/90) has shown to degrade the final average performance of the system by some 5% units of accuracy. The initial models of such instruments are highly unreliable and show a class-wise accuracy of 20%, dropping to the total incorrectness with several iterations of the algorithm. Such undesirable behaviour, however, is still improved by the algorithm, which after 20 iterations produces models for such instrument that perform with up to 80% accuracy. Therefore, the effectiveness of the algorithm when applied to smaller instrument sets still holds, however, it is suggested to have most of the classes sufficiently represented in the labelled training data set.

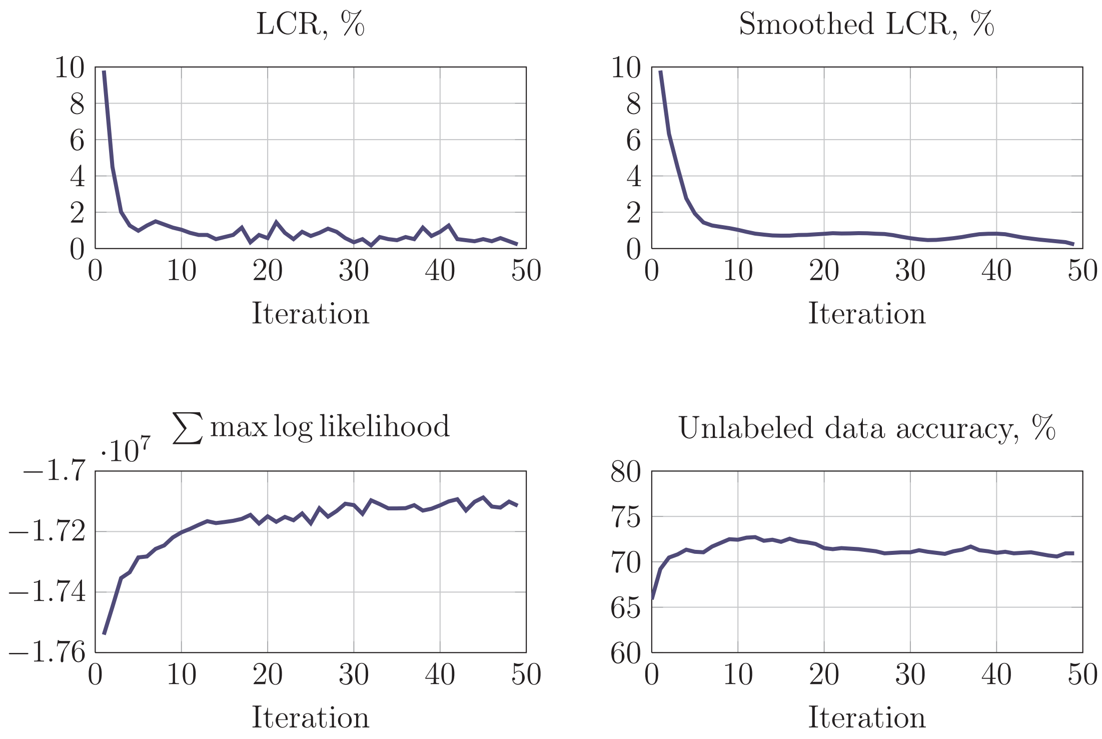
Several auxiliary curves related to the iterative EM-based algorithm used for training the smaller instrument set are presented in Figure 4.2(b). Label change rate (LCR) and particularly its smoothed version (for the smoothing procedure see Section 3.4.3) indicate that the algorithm converges after some 10 iterations, which is confirmed by the resulting accuracy curve (Figure 4.2(a)), where no significant change in the performance is observed after that point. The sums of log-likelihoods that produced labelling for the unlabelled data is rather unsteady and not highly informative, and therefore the suggested approach of terminating the algorithm by setting a threshold to a smoothed version of the LCR curve is decided to be utilised instead. The curve that represents "unlabelled" data classification accuracy is almost monotonically increasing, which suggests that the algorithm does indeed gradually incorporate the true information about the classes of the unlabelled data.

A similar behaviour of the discussed parameters of the training stage of the iterative EM-based algorithm has been observed when several modifications to the latter have been introduced in the subsequent experiment. Therefore, these are generally omitted from the further presentation with the exception of the cases with significant deviation from the demonstrated behaviour.

Figure 4.3(a) presents analogous instrument-wise accuracy as a function of number of iterations of the initial iterative EM-based algorithm applied on a larger instrument set. Such setting resembles better the realistic case of the application of the system



(a) Instrument-wise accuracies as functions of the number of iterations.



(b) Auxiliary information on the training progress as functions of the number of iterations.

Figure 4.3. *Instrument-wise accuracies (a) and auxiliary information on the training (b) of the original* iterative *EM-based SSL algorithm with a* larger *instrument set.*

Table 4.6. *Classification results with the initial iterative EM-based SSL algorithm.*

Instrument set size	**Recognition accuracy, %**			
	initial	maximum	absolute gain	relative gain
4 instruments	80.79	94.34	13.55	16.77
10 instruments	63.82	70.96	7.14	11.19

and, naturally, the expected performance is not as high as in the previous artificial example. Indeed, the initial models perform rather poorly. However, introducing unlabelled data tends to improve the performance by some 7.1% units, reaching as far as 71.0%. Although the value of maximum accuracy may be considered insufficiently high due to the complexity of the task, the increase in performance compared to the initial models is of the same order as in the easier case. The overall accuracy curve demonstrates an oscillating behaviour, an issue that is to be addressed in the upcoming sections. The instrument-wise curves demonstrate that, like with the smaller instrument case, such oscillations are caused by mutual confusability of the classes, which manifests itself in the fact that improving the models of one instrument affects the models of another one, which appears to share a part of the feature space with the former one.

The auxiliary plots (see Figure 4.3(b)) demonstrate a similar behaviour. The smoothed LCR curve tends to decrease almost monotonically, suggesting applying some thresholding based on its value in order to define the termination point.

The results of the evaluation of the initial iterative EM-based algorithm are summarised in Table 4.6. By comparing the maximum achieved accuracies with the initial models' performance one may observe a somewhat indicative improvement of the models' performance. Such results are quite close to the fully supervised case, suggesting that the implemented SSL technique is, indeed, applicable to the problem of musical instrument classification.

Nevertheless, a similar uncertainty of the models along the number of iterations is observed with both instrument sets. The indeterminacy of the iteration number, at which the algorithm needs to be terminated so that one would not end up in a local minimum, poses a requirement of a smoother transition between the models along the iterations. The effects of the proposed approaches to confront this issue are investigated in the following sections.

One-class-at-a-time training

The results obtained with the one-class-at-a-time training approach (see Section 3.4.5) on the smaller and larger instrument sets are presented in Figure 4.4(a) and Figure 4.4(b), respectively. The horizontal axis represents *macroiterations*, i.e., a set of iterations required to retrain the models of all classes once. Its size depends naturally on the number of classes in the given problem. This way, a comparison of such approach with the initial version of the algorithm appears more consistent.

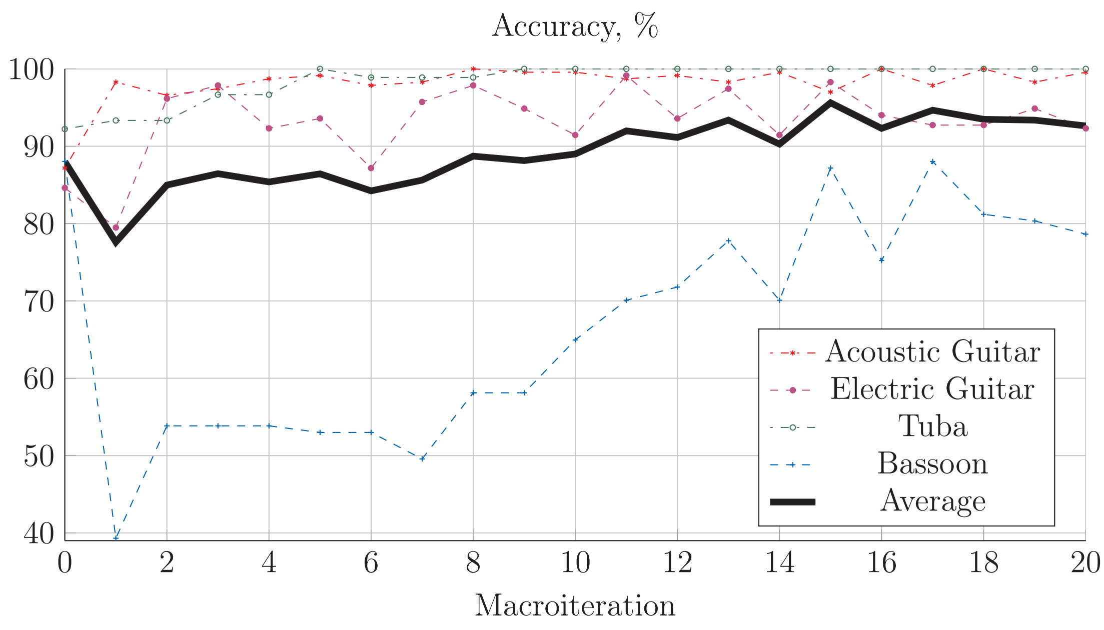
One may notice that the initial models' accuracies differ from the ones obtained with the original algorithm. The attempts to enforce consistently comparable cases were made by keeping the labelled, unlabelled and testing sets the same for all the evaluated algorithm versions. Despite this, some variations of the results, which are expected to be the same, are observed. The possible reason for this is the randomisation that occurs within the EM-algorithm in its supervised form, which is incorporated in each of the evaluated algorithms. This draws a conclusion that in order to obtain a system that would constantly produce a stable and predictable level of performance one may need to train several classifiers and utilise the majority vote principle in the classification stage. This way, another undesirable effect is additionally minimised, namely, the fact that in some cases the improvement compared to the initial models was not as apparent. The approach utilising a majority vote across several classifiers is expected to overcome this complication, too.

The results of the evaluation on both instrument sets demonstrate that the suggested approach does indeed overcome the models' "swapping" issue to some extent, especially with the smaller instrument set: the number of local maxima in the accuracy curve of one class corresponding to local minima of the curves of another one is significantly smaller, and the range of such oscillations appears to be reduced. Moreover, the models of almost each class appear to generally improve along the iterations in the long run (see, for instance, the accuracy curves of Acoustic Guitar in the smaller instrument set with and without one-class-at-a-time training). However, an undesired effect of the decrease of the overall accuracy during the first several iterations, which was observed mostly with the larger instrument set, is now enforced even on the smaller set. Nevertheless, such initial drop is soon obviated and its effect may be neglected.

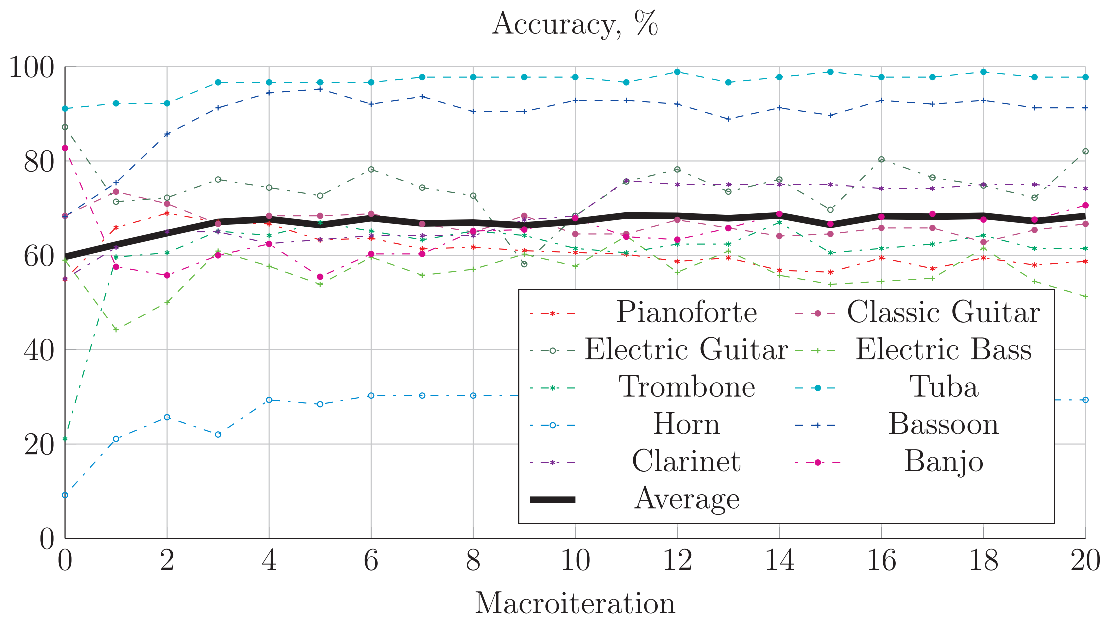
The maximum achieved accuracies with the smaller and larger instrument sets are compared to the initial models' accuracies in Table 4.7. The results do not appear to differ much from the ones obtained with the initial algorithm version. The suggested technique appears to demonstrate the advantage not in the overall performance per se, but rather in the improved straightforwardness of the convergence decision due to the increased smoothness of the accuracy values along the iterations.

Labelled data weighting approach

The weghting of the contribution of the labelled data to the training process (see Section 3.4.4) was done with a decreasing function ω of a macroiteration index t^*



(a) Smaller instrument set.



(b) Larger instrument set.

Figure 4.4. *Instrument-wise accuracies as functions of the number of macroiterations (set of iterations needed to retrain all the instruments once) of the* iterative *EM-based SSL algorithm with* one-class-at-a-time training*.*

Table 4.7. *Classification results with the modified iterative EM-based SSL algorithm with one-class-at-a-time training.*

Instrument set size	**Recognition accuracy, %**			
	initial	maximum	absolute gain	relative gain
4 instruments	88.01	95.62	7.61	8.65
10 instruments	59.68	68.46	8.78	14.71

Table 4.8. *Classification results with the modified iterative EM-based SSL algorithm with labelled data weighting.*

Instrument set size	**Recognition accuracy, %**			
	initial	maximum	absolute gain	relative gain
4 instruments	83.74	94.42	10.68	12.75
10 instruments	60.03	68.13	8.10	13.49

(i.e., an index corresponding to a single retraining stage of all the classes):

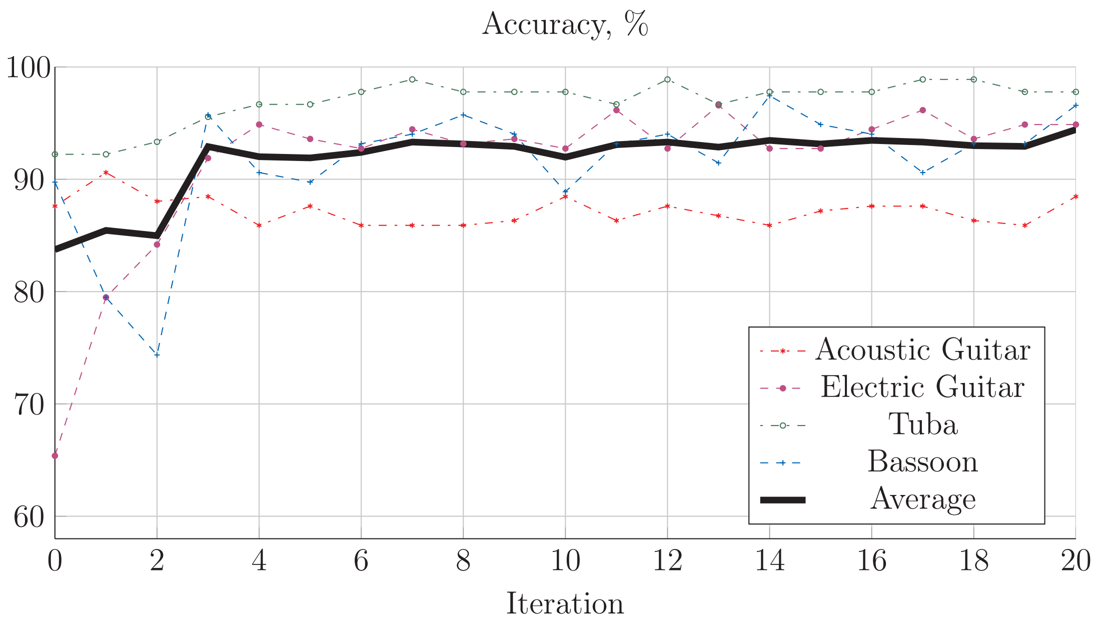
$$\omega(t^*) = [8, 6, 4, 2, 1]. \tag{4.2}$$

The effects of the labelled data weighting approach on the smaller and larger instrument sets are reflected in Figure 4.5. The average accuracy curve appears, indeed, smoother than in the case with the initial version of the algorithm. Regarding the maximum values of the overall accuracy, the improvement of the models compared to the initial ones is, again, apparent (see Table 4.8), although the deviations from the initial algorithm's results, as well as from the one-class-at-a-time approach, are not significant enough to be an evidence of advantage of this approach, but are more likely to be a consequence of the randomisation effects.

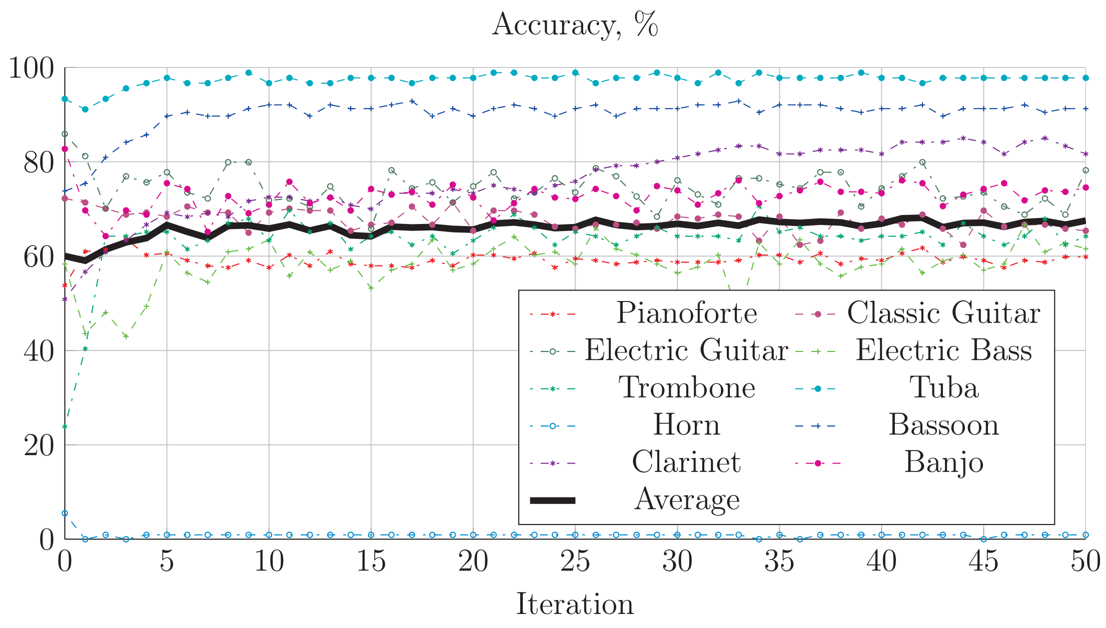
An analogous behaviour of the peaks in the instrument-wise accuracies is, again, noticeable. Therefore, it appears worthwhile to combine this approach with the previously suggested "one-class-at-a-time training".

Comparison of the approaches

The average accuracies produced by each approach and their combination are additionally compared in Figure 4.6. The benefits that the modifications to the algorithm provide in the simpler classification scenarios are most apparent in combination, leading to a rather steadily increasing average accuracy curve. Furthermore, the combined approach leads to the highest final accuracy value in this case. The detailed summary of the performance of this combination is reflected in Table 4.9. The gain obtained with the smaller instrument set is lower than in the previous cases,



(a) Smaller instrument set.



(b) Larger instrument set.

Figure 4.5. *Instrument-wise accuracies as functions of the number of iterations of the* iterative *EM-based SSL algorithm with* labelled data weighting.

Table 4.9. *Classification results with the iterative EM-based SSL algorithm when incorporating both modifications.*

Instrument set size	**Recognition accuracy, %**			
	initial	maximum	absolute gain	relative gain
4 instruments	89.19	95.30	6.11	6.85
10 instruments	58.73	68.43	9.70	14.18

however, this is more likely due to the fortunate initial models, and the advantages of the combined approach are nevertheless apparent.

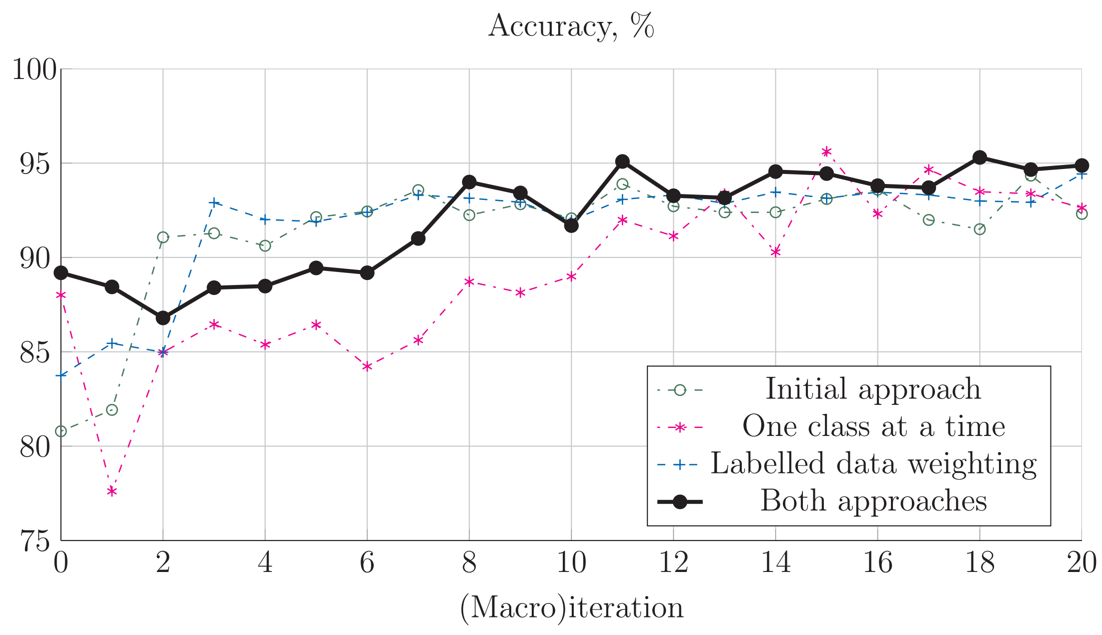
On the other hand, once the number of instrument increases, the ultimate effect of the modifications to the algorithm does not appear explicitly beneficial. Even though these are successful in eliminating the initial decrease present in the original setup while maintaining a relatively stable increase of the accuracy, the final accuracy value does not significantly differ from the one produced by the original algorithm. Therefore, the major benefit of the proposed modifications is seen in that they enable a simplified convergence decision due to the almost monotonous increase of the accuracy function.

Dependency of the accuracy on the labelled-to-unlabelled ratio

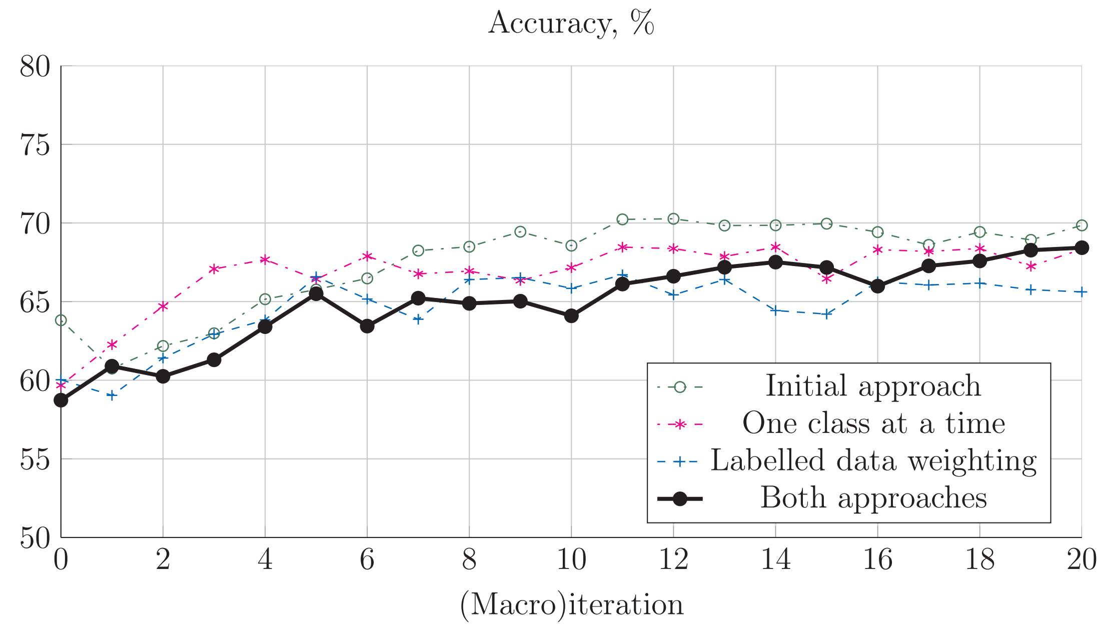
In this section, the effect of the relative size of the labelled data in the training dataset on the final accuracies and on the accuracy change compared to the initial values is investigated. The results of this experiment conducted in the four-instrument case are presented in Figure 4.7.

The results demonstrate i.a. that there are cases when the labelled data size can be insufficiently high for the iterative algorithm to be able to introduce improvement. For instance, in the case of the smaller instrument set its value of 5%, which corresponds roughly to only several labelled training instances of the insufficiently represented instrument Tuba, results in actual degradation of the system's performance compared to the supervised initial models. However, already 10% of the relative labelled dataset size yields positive absolute gain of 5%, which grows up to 20% in the case of 15% of labelled data. The subsequent decrease of this gain appears not to be a shortcoming of the algorithm as such. It is more likely due to the improved initial models, which do not leave as much opportunity for the semi-supervised technique to take effect.

Due to extensive computational requirements of this experiment, such parameter sweep is not performed on the larger instrument set. However, a similar behaviour is expected to be observed. For instance, setting the relative labelled dataset size to 15% has shown a lower, but still notable absolute accuracy gain of 9.0%. The performance



(a) Smaller instrument set.



(b) Larger instrument set.

Figure 4.6. *Comparison of the modifications to the* iterative *algorithm with their combination and the initial version.*

is expected to be similarly, almost monotonically improving when increasing the size of the labelled dataset up to a certain point. However, since the choice of such optimal value is not theoretically motivated, a simpler approach would be to aim at utilising as much labelled data as there is available, while taking into consideration the capabilities of the algorithm to improve the performance with almost any sufficiently high size of the labelled dataset.

The final accuracy value occasionally exceeds the baseline one, i.e., the accuracy of the fully supervised case with all available data treated as labelled. Such effect appears to be influenced by the fact that some training samples negatively contribute to the generalisation properties of the models, and, therefore, their misclassification during semi-supervised training is, in fact, beneficial. To study, whether this is always the case or just a fortunate coincidence, could be an appealing research problem.

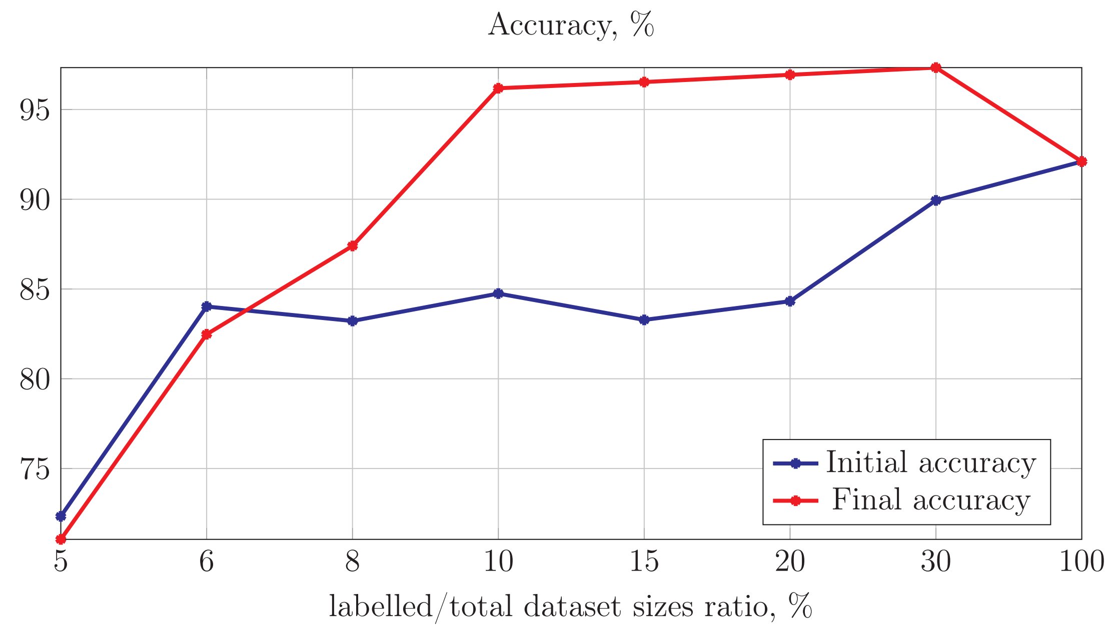
4.5.2 Incremental EM-based algorithm

The resulting accuracies obtained on both instrument sets with the second, incremental EM-based algorithm are presented in Figure 4.8. As mentioned in Section 3.4.5, the one-class-at-a-time training is always incorporated in the algorithm as a result of the preliminary tests, which have shown that otherwise the algorithm does not yield any improvement of the initial models whatsoever.

The evaluation has been performed with the 2.5% of unlabelled data that is used to retrain the model of a current class at each iteration. Such number appears to be a reasonable trade-off between the effectiveness of the algorithm (which ideally performs such retraining with only one most confident newly obtained labelled instance) and computational feasibility.

Despite the undertaken attempts, no significant improvement is observed in the evaluations of the incremental algorithm. The accuracy curve shows some peaks that are higher than the initial value, however, there is no apparent way of detecting them in a real-life situation. Moreover, the algorithm is meant to be terminated once all the unlabelled data is moved to the labelled pool, i.e., at the last iteration. The accuracy in this case is not improved in either of the instrument scenarios.

Generally, the accuracy fluctuates around the initial value produced by the purely labelled data. This suggests an assumption that the small portion of unlabelled data introduced to retrain the existing rather certain models, is unable to affect them to a high extent due to its insufficiency. To further explore this issues, one could modify the size of this unlabelled data portion added to the labelled set at each iteration. Supposedly, there could be observed some improvement when this data portion size is significantly higher than the evaluated 2.5%. However, by doing so, one would eventually end up with the one-iteration version of the iterative algorithm



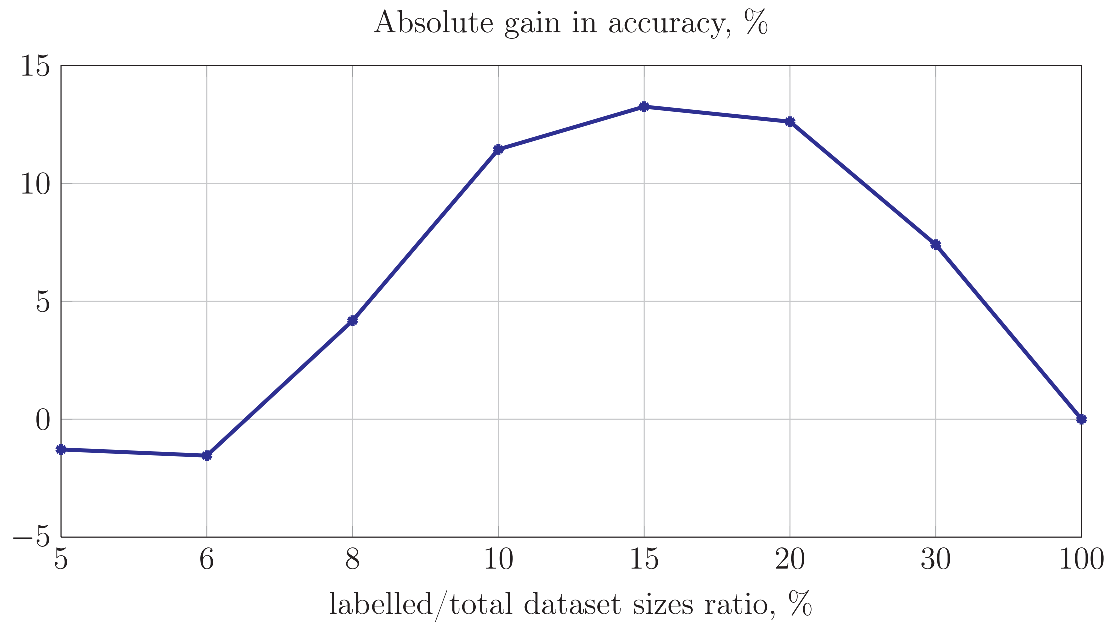


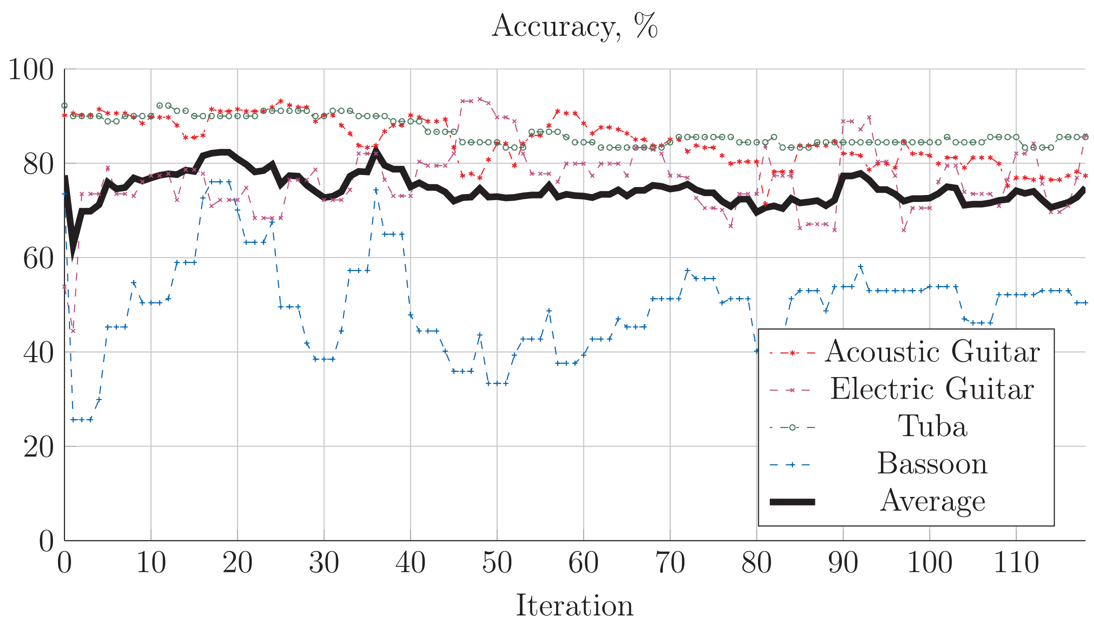
Figure 4.7. *Comparison of the accuracies of the initial models and the final iteration of the* incremental *EM-based SSL algorithm as a function of the labelled dataset size relative to the total size of the training dataset with the* smaller *instrument set. The training and evaluation are performed three times for each value with randomisation across the notes chosen for the labelled set from a fixed instrument instance, and the accuracies are averaged.*

discussed previously. Furthermore, such approach contradicts the original essence of the algorithm, i.e., to operate on one most certainly classified training instance at an iteration. Therefore, it appears most logical to utilise the iterative algorithm instead, given its rather successful performance and computational efficiency.

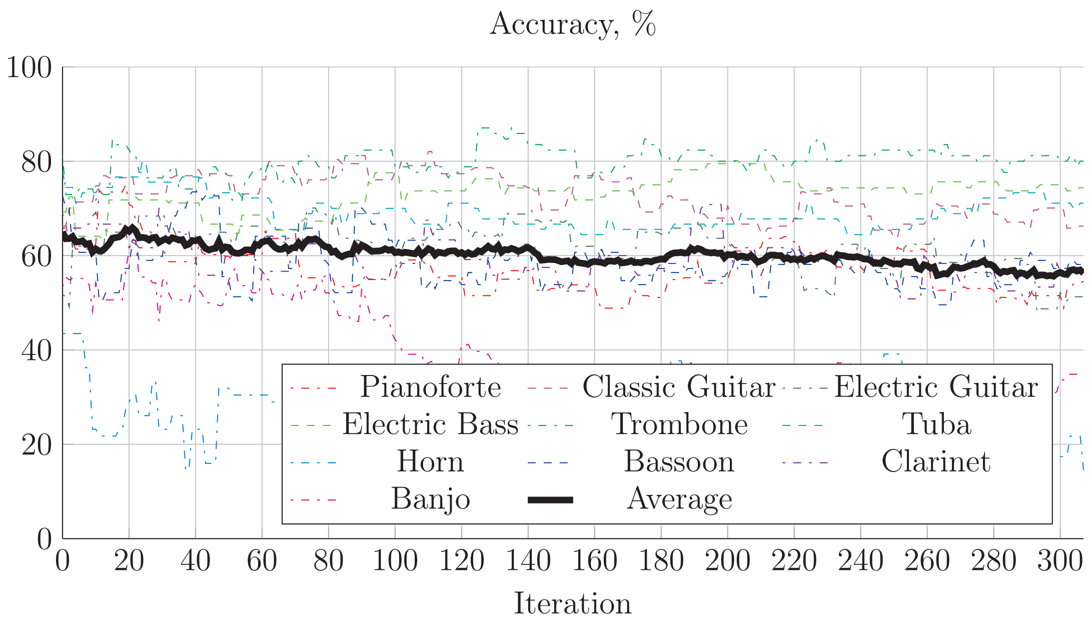
4.5.3 Discussion

Out of the two implemented algorithms one has shown a rather noteworthy performance in both simplified and sophisticated classification scenarios. Namely, the iterative EM-based algorithm with the two proposed modifications has managed to demonstrate the absolute improvement of the initial models' accuracy by up to 6.1% and 9.7% for the smaller and larger instrument sets, respectively. The advantages of the semi-supervised techniques introduced with the aid of the algorithm are present even when the labelled set is as small as 8% in relation to the unlabelled set size. The introduced modifications assist also in finding a convergence point. One could either observe the changes of the labelling of the data at each iteration (LCR) or coarsely terminate the execution after sufficiently many iterations, knowing that the modifications reduce to a great extent the possibility of local minima as well as their range.

The incremental EM-based algorithm has not shown to be effective for the given classification problem. It is likely to produce improvements if the conditions are such that the algorithm resembles the iterative one (i.e., when the size of the unlabelled data portion moved to the labelled pool is relatively high), but even in such case its computational efficiency is lower than the one of the iterative algorithm. Therefore, a logical conclusion is to presume the incremental algorithm, as it is presented, insufficiently applicable for this classification problem with the given feature extraction method, and to utilise the iterative algorithm instead.



(a) Smaller instrument set.



(b) Larger instrument set.

Figure 4.8. *Instrument-wise accuracies as functions of the number of iterations of the* incremental *EM-based SSL algorithm with adding 2.5% of the unlabelled data at each iteration.*

5. CONCLUSIONS AND FUTURE WORK

In this work, various approaches to semi-supervised learning have been studied with an emphasis to music information retrieval, particularly musical instrument recognition. In addition to the semi-supervised schemes, the common techniques applied for musical instrument recognition have been reviewed, as well as other audio-related areas of pattern recognition that utilise semi-supervised techniques.

Furthermore, based on the two algorithms that realise the mixture model SSL scheme, a musical instrument recognition system has been developed with various modifications aimed at improving the overall accuracy and simplification of the convergence criteria. The system's performance has been evaluated on two distinct scenarios, which differ in the number of instruments to be classified, i.e., classification scenarios of different level of complexity.

The evaluation has shown notable results in one of the implemented algorithms with the proposed modifications, namely, *the iterative EM-based algorithm with one-class-at-a-time training and labelled data weighting*. In the case of the extended algorithm with as little as 15% initially labelled training data, the absolute accuracy increase was 6.1% in the simple and 9.7% in the complex classification scenario.

The resulting overall classification accuracy has shown to behave with a somewhat noticeable level of instability along different experiments given the same instrument sets, presumably caused by the randomisation effects occurring within the incorporated supervised EM algorithm. In order to overcome such undesirable instability, it is suggested to train several classifiers and perform a majority vote for the ultimate classification.

The other implemented algorithm, *the incremental EM-based algorithm*, has shown somewhat poor performance. The supposed reason behind this is the overdominating nature of the initial models when the size of the unlabelled data portion moved to the labelled pool is relatively low.

As a suggestion for the future investigation, a more sophisticated feature extraction method, such as the ones reviewed in Section 2.2.2, could be incorporated into the developed system in order to study the performance of the algorithms when the Bayes error is lower than in the current implementation. Such a case would be an easier scenario for a classifier to work on due to the reduced overlapping area between the classes in the feature space, and the improvement introduced by the semi-supervised techniques could be also higher.

Additionally, it is suggested to study the observed phenomenon of the overall performance of the implemented SSL-based scheme exceeding the performance of the baseline supervised classifier given the same amount of training data. Namely, it appears worthwhile to explore whether such phenomenon occurs in other scenarios and to investigate the possible reasons behind this behaviour.

Another area of further investigation are the possible measures to overcome the issue of overdominating initial models in the case of the incremental algorithm. A possible approach would be to de-weight their contribution (as opposed to increasing the weight of the labelled data in case of the modified iterative algorithm) with a gradual increase while moving the unlabelled data portions to the labelled set. However, such an approach is not expected to significantly outperform the iterative algorithm with the proposed modifications.

Furthermore, an additional investigation of the system's performance could be conducted in more complex scenarios by increasing the number of instruments in the set or by introducing noise, reverberation and distortions to the datasets in order to mimic the real-world application scenario. Finally, the implemented and evaluated algorithms, having demonstrated their utility for the musical instrument recognition problem as such, may find their use in neighbouring pattern recognition problems as well. For instance, to investigate the effect of introducing SSL of the instruments in a musical genre classification system would be an interesting suggestion for the future work.

REFERENCES

[1] G. Agostini, M. Longari, and E. Pollastri. Musical instrument timbres classification with spectral features. In *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, pages 97–102, 2001.

[2] G. Agostini, M. Longari, and E. Pollastri. Musical instrument timbres classification with spectral features. *EURASIP Journal on Advances in Signal Processing*, 2003(1):943279, 2003.

[3] A. Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th international conference on Machine learning*, pages 49–56. ACM, 2007.

[4] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 26–33, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.

[5] J. Barbedo and G. Tzanetakis. Musical instrument classification using individual partials. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(1):111–122, 2011.

[6] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 19–26, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[7] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.

[8] J. Burred, A. Robel, and T. Sikora. Polyphonic musical instrument recognition based on a dynamic model of the spectral envelope. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 173–176. IEEE, 2009.

[9] V. Castelli and T. M. Cover. On the exponential value of labeled samples. *Pattern Recogn. Lett.*, 16(1):105–111, Jan. 1995.

[10] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, USA, 2006.

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sept. 1995.

[12] P. Cosi, G. De Poli, and P. Prandoni. Timbre characterization with mel-cepstrum and neural nets. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 42–45, 1994.

[13] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

[14] G. De Poli and P. Prandoni. Sonological models for timbre characterization. *Journal of New Music Research*, 26(2):170–197, 1997.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38, 1977.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, USA, second edition, 2001.

[17] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Performance Evaluation*, 64(9-12):1194–1213, 2007.

[18] A. Eronen. Comparison of features for musical instrument recognition. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.

[19] A. Eronen. *Signal Processing Methods for Audio Classification and Music Content Analysis*. PhD thesis, Tampere University of Technology, Tampere, Finland, June 2009.

[20] S. Essid, G. Richard, and B. David. Musical instrument recognition based on class pairwise feature selection. In *International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, Oct. 2004.

[21] S. Essid, G. Richard, and B. David. Musical instrument recognition on solo performances. In *European Signal Processing Conference (EUSIPCO)*, Vienna, Austria, Sept. 2004.

[22] N. H. Fletcher and T. D. Rossing. *The Physics of Musical Instruments*. Springer, 1998.

[23] D. Foley. Considerations of sample and feature size. *Information Theory, IEEE Transactions on*, 18(5):618–626, sep 1972.

[24] G. Getz, N. Shental, and E. Domany. Semi-supervised learning–a statistical physics approach. *arXiv preprint cs/0604011*, 2006.

[25] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Music genre database and musical instrument sound database. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR)*, pages 229–230, 2003.

[26] F. Gouyon, F. Pachet, O. Delerue, et al. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*, 2000.

[27] J. M. Grey. *An Exploration of Musical Timbre.* PhD thesis, Stanford University, Stanford, California, 1975.

[28] M. Gupta and Y. Chen. *Theory and Use of the EM Algorithm*, volume 3. Now Pub, 2011.

[29] T. Hastie, R. Tibshirani, and J. Friedman. 8.5 the em algorithm. *The Elements of Statistical Learning. New York: Springer*, pages 236–243, 2001.

[30] T. Heittola, A. Klapuri, and T. Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 327–332, 2009.

[31] K. Irwin. Musipedia: The open music encyclopedia. *Reference Reviews*, 22(4):45–46, 2008.

[32] K. Jensen. *Timbre Models of Musical Sounds: From the Model of One Sound to the Model of One Instrument.* Rapport. Københavns Universitet, Datalogisk Institut, 1999.

[33] J. H. Jeon and Y. Liu. Semi-supervised learning for automatic prosodic event detection using co-training algorithm. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 540–548, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[34] C. Joder, S. Essid, and G. Richard. Temporal integration for audio classification with application to musical instrument classification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(1):174–186, 2009.

[35] I. Kaminsky and A. Materka. Automatic source identification of monophonic musical instrument sounds. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 1, pages 189–194. IEEE, 1995.

[36] S. Z. K. Khine, T. L. Nwe, and H. Li. Singing voice detection in pop songs using co-training algorithm. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE Int. Conference on*, pages 1629–1632. IEEE, 2008.

[37] B. Kostek and A. Czyzewski. Representing musical instrument sounds for their automatic classification. *J. Audio Eng. Soc.*, 49(9):768–785, 2001.

[38] T. Li and M. Ogihara. Semi-supervised learning for music artists style identification. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, CIKM '04, pages 152–153, New York, NY, USA, 2004. ACM.

[39] D. Little and B. Pardo. Learning musical instruments from mixtures of audio with weak labels. In *ISMIR'08*, pages 127–132, 2008.

[40] J. Liu and L. Xie. Svm-based automatic classification of musical instruments. In *Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on*, volume 3, pages 669–673. IEEE, 2010.

[41] R. Loughran, J. Walker, and M. O'Neill. An exploration of genetic algorithms for efficient musical instrument identification. In *Signals and Systems Conference (ISSC 2009), IET Irish*, pages 1–6. IET, 2009.

[42] C. McKay and I. Fujinaga. Automatic music classification and the importance of instrument identification. In *Proceedings of the Conference on Interdisciplinary Musicology*, 2005.

[43] R. Michalski. Pattern recognition as rule-guided inductive inference. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4):349–361, 1980.

[44] P. J. Moreno and S. Agarwal. An experimental study of EM-based algorithms for semi-supervised learning in audio classification. In *Proceedings of the ICML-2003 Workshop on the Continuum from Labeled to Unlabeled Data*, 2003.

[45] I. Muslea, S. Minton, and C. Knoblock. Active + Semi-supervised Learning = Robust Multi-view Learning. In *Proceedings of ICML-2002*, pages 435–442, 2002.

[46] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2):103–134, 2000.

[47] D. O'Shaughnessy. *Speech Communications: Human and Machine*. Wiley-IEEE Press, 2 edition, 1999.

[48] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall Signal Processing Series. PTR Prentice Hall, 1993.

[49] R. Rui and C. chun Bao. Projective non-negative matrix factorization with bregman divergence for musical instrument classification. In *Signal Processing, Communication and Computing (ICSPCC), 2012 IEEE International Conference on*, pages 415 –418, aug. 2012.

[50] Y. Song, C. Zhang, and S. Xiang. Semi-supervised music genre classification. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–729. IEEE, 2007.

[51] B. Sturm, M. Morvidone, and L. Daudet. Musical instrument identification using multiscale mel-frequency cepstral coefficients. *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pages 477–481, 2010.

[52] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, 2008.

[53] S. Tjoa and K. Liu. Musical instrument recognition using biologically inspired filtering of temporal dictionary atoms. In *Proc. Int. Soc. Music Information Retrieval Conf., Utrecht, Netherlands*, pages 435–440, 2010.

[54] V. Vapnik. *Statistical learning theory.* Adaptive and learning systems for signal processing, communications, and control. Wiley, 1998.

[55] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *In ICML*, pages 577–584. Morgan Kaufmann, 2001.

[56] A. Wang. The shazam music recognition service. *Communications of the ACM*, 49(8):44–48, 2006.

[57] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[58] W. You and R. Dannenberg. Polyphonic music note onset detection using semi-supervised learning. *Proc. ISMIR, Vienna, Austria*, 2007.

[59] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan. Semi-supervised adapted hmms for unusual event detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 611 – 618 vol. 1, june 2005.

[60] Z. Zhang and B. Schuller. Semi-supervised learning helps in sound event classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 333–336, march 2012.

[61] X. Zhu, Z. Ghahramani, J. Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

[62] X. Zhu and A. Goldberg. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2009.