# Active Learning for User-Tailored Refined Music Mood Detection.

**Álvaro Sarasúa Berodia**

MASTER THESIS – UPF 2011
Master in Sound and Music Computing

Master Thesis Supervisors:
Perfecto Herrera and Cyril Laurier

Department of Information and Communication Technologies
Universitat Pompeu Fabra, Barcelona

# Active Learning for User-Tailored Refined Music Mood Detection.

Master Thesis, Master in Sound and Music Computing

Álvaro Sarasúa Berodia.

alvarosarasua@gmail.com

Department of Information and Communication Technologies.

Music Technology Group.

Universitat Pompeu Fabra, Barcelona.

# Abstract

Mood detection is an increasingly area of interest in Music Information Retrieval (MIR). This thesis, which is built on top of the work by Cyril Laurier and Perfecto Herrera in the Music Technology Group (MTG), deals with two identified problems which are interrelated: first, the need for expanding current mood tags (*happy, sad, aggressive* and *relaxed*) to more specific and complex emotions; second, the need for customization that these complex emotions require.

For the first problem, four new mood classifiers are implemented: *mysterious, humorous, sentimental* and *triumphant*. New song collections are created and user-validated in order to achieve this task.

For the second issue (customization), the use of active learning techniques is explored. Active learning is based on the idea that a system can converge to its best performance more quickly by being able to smartly select those songs with which it is trained. A state-of-the-art review on active learning techniques is presented and uncertainty-based techniques are tested over the new and already existing song collections. ANalysis Of VAriance (ANOVA) showed that there is no significant improvement between active learning and standard full-set batch training. Active learning, though, would require the processing of fewer instances to achieve equivalent performance.

As an extra experiment, a new *happy* collection with classical music is created. Different cases are studied and compared: a system trained with the old collection (that does not contain classical music) tested over the new one and vice versa, a system trained joining both collections and tested by 10-fold cross validation, etc. The results suggest that systems trained with certain genres do not generalize well to others.

# Acknowledgments

First of all, I want to thank my tutors Perfecto Herrera and Cyril Laurier who gave me more support and time than I could imagine. This work would not have been done without their help. I want to mention as well Nicolas Wack and Hendrik Purwins, who were not my supervisors but who were always there when I asked for help. Second, I thank Xavier Serra for trusting me and allowing me to become a part of the Music Technology Group. Also, I want to thank everyone in this Group, teachers and colleagues; it has been an incredible experience to learn so many things during these two years. All of you made me think that moving to Barcelona during this time has been one of my best decisions ever.

Then, I want to thank those people in Barcelona who became more than colleagues. You became my family during this time. I hope we can all meet again sometime. I am sure it will happen. Although it is always hard to mention someone, I need to specially thank Andrés Bucci. I see you as my (older) brother.

Finally but most important, I thank my family, not just for their support, but mainly for always giving the feeling that all my successes make them truly happy. That is priceless.

# Contents

# 1. Introduction

*"Though defined in many different ways, basically, music could be defined as the art of organizing sounds with the goal of conveying a certain emotion in the listener"*

Enric Herrera

Music is one of the most popular arts. Technology has developed changing the way in which people enjoy music. Apart from buying albums, attending concerts or listening to the radio, the number of people who enjoys music on the Internet is constantly increasing. In this context, users usually look for single songs and try to find new artists which they may like. Recommender Systems (RS) try to make this easier by automatically detecting songs or artists that are likely to fit the user's interests.

In this context, mood detection has become a topic of increasing interest and for example the Music Information Retrieval Evaluation eXchange (MIREX) has a specific context on Audio Music Mood Classification. Systems developed in the MTG got very good results. MTG technology Essentia includes mood labels among the high level descriptors that can be extracted from audio.

However, there are things which are of course still to be improved. Two problems are addressed in this work: first, the emotional complexity of emotions in music cannot be explained with 4 or 5 labels; second (but related), when dealing with non-basic emotions the agreement among users is reduced and thus there is a necessity of training the systems differently for each of them. The problem for this is that getting labels from the user may be expensive and the systems should ideally get a good performance by being trained with few instances.

## 1.1. Goals

The goals of this work are the following:

1- Add 4 new labels to the current ones (*happy*, *sad, aggressive*, *and relaxed*). The four selected labels are:

   a. Mysterious

   b. Humorous

   c. Sentimental

   d. Triumphant

   This goal implies another one which is creating song collections for each of the new labels and validating them with users.

2- Reduce the size of the training sets for the old and new categories by exploring active learning techniques. Active learning is based on the idea that the systems can increase their accuracy if they are able to *actively* select samples which may improve the most their performance.

3- As an extra goal, we study how the old *happy* model extends to a new *happy* classical music collection which is created.

## 1.2. Structure of the thesis

The current document is organized as follows:

Chapter 2 reviews the state of the art of the fields that relate to this work. First, it deals with the relation of music and emotion. Then, there is a summary on the techniques that are used in the Music Information Retrieval (MIR) field in terms of feature extraction and classification. Finally, there is an explanation on what active learning consists on, a summary of the different approaches that appear in literature and some examples of its use in MIR.

Chapter 3 explains the methodology used in this thesis. That implies explaining how the new labels were selected, how the song collections were created and evaluated and how active learning was implemented and tested. The results of all these tasks are presented in Chapter 4.

Chapter 5 shows the results for the additional goal of exploring how old models extend to classical music.

Chapter 6 summarizes the conclusions that can be extracted from the results and proposes extensions to the work that is presented.

Finally, there is an Appendix about some problems that were found when using GAIA for active learning during this work.

# 2. State Of The Art

In the following chapter we make a literature review on the topics that relate to this Master Thesis. Our work will be built on top of Cyril Laurier's [1] which already contains a comprehensive state of the art review on Mood Detection. Because of this, the present Chapter tries to give an overview on the topics that directly relate to our work; further details can be found in the mentioned reference.

First, we deal with the relationship between music and emotion and the different approaches for representing it. Second, we study the techniques that are used for content-based (i.e., from the audio) mood detection in music and justify the need for customization in order to detect non-basic emotions. Finally, we explain the idea behind active learning and the way it can help us make our system customizable.

## 2.1. Music and emotion

The relation of music and emotion has been broadly studied from many different perspectives and fields such as psychology, sociology, neuroscience, musicology, philosophy or machine learning. Probably, the complete

understanding of music emotions phenomenon needs this multidisciplinary approach to be completely explained as it involves cultural, social, psychological, cognitive and musical aspects.

Indeed, we can assert that emotions play an important role in music. Every individual involved in music has somehow an emotion-related motivation most of the times: composers expressing their feelings, performers inducing them or listeners *feeling* them. The music itself has some characteristics on its structure (tonal, rhythmic…) that have some resemblances to other mechanisms of emotion expression [2]. However, even though it seems quite obvious for everyone that emotions and music are closely related, it is not that obvious at all why and how does that happen.

### 2.1.1. Emotion and mood

Most of the time in this work we talk about "mood detection" instead of "emotion detection". It seems that emotion and mood could be used indifferently, but actually they are not exactly the same and there are some nuances that should be clarified. With this purpose, we will now define both.

Emotion itself is not easy to define, as stated by Fehr and Russell: "Everyone knows what an emotion is, until asked to give a definition" [3]. It seems interesting to study the etymology of the word in order to better understand its meaning. In this case, emotion comes from Latin *emovere*, from e- 'out' + movere 'move', so it would be something like the "external impulse that induces to motion / action" [4]. Indeed, some authors say that the importance of emotion is related to survival (e.g., fear as necessary to escape from dangerous situations) [5]. According to Damasio, emotions are a series of body state changes that are connected to mental images that activate a given brain subsystem. In

this sense, emotions would involve physiological reactions but they would also be object-oriented, and they will induce an attitude towards this object [6].

Moods could be considered as lasting emotional states. They are not object-oriented and can take into account many general feelings. According to this idea, it seems better to talk about moods in our work as we are dealing with what music transmits to the listener during time. We are not studying "musical events", but whole excerpts of music that we assume to convey or contain certain moods. Again, the difference between emotions and moods is very subtle, but for example we would not consider surprise as a mood, while we could consider it as an emotion [2].

### 2.1.2. How does music convey emotion?

Understanding how music conveys emotion is a complex problem which is still studied with different approaches. Kivy [7] proposes two different hypotheses. First, there may be a "hearing resemblance between the music and the natural expression of the emotion". For example, both angry voice and music share high loudness and spectral dissonance. The second hypothesis states that the emotions appear because of "accumulated connotations a certain musical phenomena acquire in a culture". According to this hypothesis, emotions conveyed by music would depend on the culture of the listener.

Balkwill and Forde Thomson [8] used a cross-cultural approach in which Western listeners were asked to rate the degree of basic emotions (joy, sadness, anger and peace) in Hindustani music, as well as some psychophysical variables (tempo, rhythmic complexity, melodic complexity and pitch range). According to their results, judgments of emotions were correct in the sense that they corresponded to the "rasas" (moods) that the songs were supposed to contain. Moreover, the emotion ratings were highly related to the psychophysical

variables, so they concluded that there were some universal aspects in music that relate directly to basic emotions, not being necessarily culture-dependent.

Grewe et. al [9] demonstrated that emotions can also depend on individuals' musical experience. For example, musicians that have studied a piece for its performance are more likely to rate an emotion higher, being this an auto-reinforcement by training. We know, however, that listening too many times to a song can cause the contrary effect, i.e., a person can feel bored or less sensitive to a certain piece after listening to it a lot of times.

Relatively recent studies in neuroscience, that take advantage of new brain imaging techniques, are also giving some interesting hints about the way we perceive emotion in music. Koelsch et. al [10] used fMRI to see which parts of the brain were activated for different tasks related to music and emotion. Blood and Zatorre [11] showed that music that conveyed "shivers-down-the-spine" or "chills"-like emotions activate regions that are also activated for other euphoria-inducing stimuli such as food, sex and drugs, and that are involved in reward and motivation.

According to Huron [12], expectation plays an important role on emotions evoked by music. He identifies different functional systems related to expectation that cause emotional reactions on the subject. He relates musical *tools* (e.g. syncopation, cadence, meter, climax) to these systems giving clues about how the combination of musical characteristics lead to several complex emotional responses.

Finally, it is important to distinguish between induced and perceived emotions [13]. The former would be the "emotion in music", i.e., the intended emotion. The latter would be the "emotion from music", i.e., the one that is felt while listening to a musical piece. For example, if someone is angry, people might perceive this anger but would feel scared or defensive. Both aspects are related, but it is

important that people usually agree more on the perceived emotion than in the induced emotion. The perceived emotion is the one we will usually refer to.

Summarizing, it seems clear that the relationship of music and emotion exists and it is definitely important in the way people enjoy music. However, emotions that appear in music seem to be a sum of different aspects that have to do with not just the music itself, but also with cultural and musical experience aspects of the listeners.

### 2.1.3. Emotional representations

In our work we want our system to be sensitive to emotions or, to say in some other way, to be able to automatically detect emotions from audio. It seems clear that, in this context, it becomes necessary to build a model to represent these emotions in a clear and meaningful way. In the literature we find two main paradigms of emotion representation (which are not music specific, but have been refined for music): the *categorical* and *dimensional* representations.

*Categorical representation*

In the categorical representation approach, emotions are divided into categories where each emotion is defined with one or several adjectives. One of the most important works in this direction is the one by Hevner [14], where she proposes the adjective circle we can observe in Fig. 1. In this representation, 67 words are arranged into 8 clusters. In each of the clusters, the adjectives are closely related. With this philosophy, one can work at the cluster level, being the categories actually 8 instead of 67. These categories have been used and modified during time, especially for western classical music. The problem of the categorization is that these word lists cannot completely describe the complexity of the possible emotions that can appear in music. On the other hand, it helps to

achieve agreement between people and simplifies the problem for automatic mood detection systems [2].

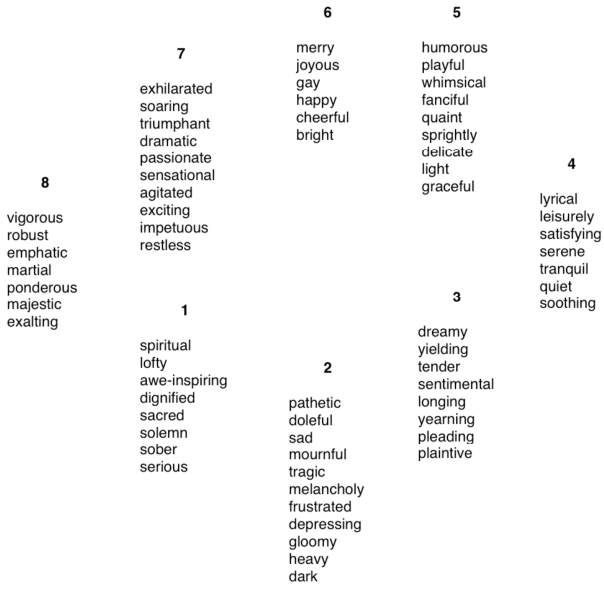



Fig. 1. Adjectives and clusters, adapted from [14]

*Dimensional representation*

In this other approach, emotions are organized along axes. The most important and used model is the "circumplex model of affect" by Russell [15], that consists in a two-dimensional space where the axes correspond to arousal (activity, excitation of the emotion) and valence (whether if the emotion is positive or negative). Fig. 2 shows a list of adjectives represented in their correspondent points in the space.

Some other models have been developed but they somehow relate to the same concept. The main advantage of using this approach is that almost any emotion can be represented, although the use of these dimensions represents a limitation itself. The main disadvantage is that two emotions with very different semantic meaning (or different psychological and cognitive mechanisms involved) can appear close to each other in the representation. See, for example, that "Alarmed"

and "Angry" appear quite close in Fig. 2, when actually they refer to quite different (though maybe related) concepts.

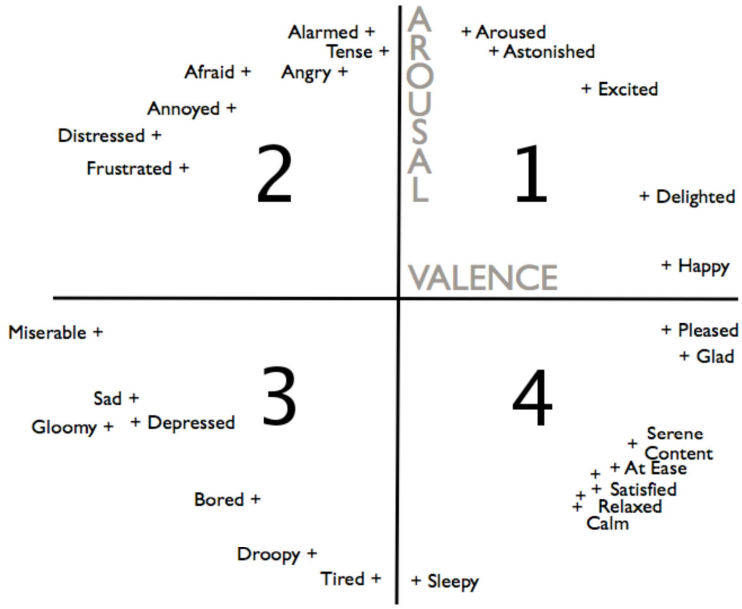



Fig. 2. "Circumplex model of affect", adapted from [15]

### 2.1.4. Musical features and emotion

Many studies deal with the relationship between musical features and the emotions they convey/express. A quite comprehensive and rigorous study on this way is the one made by Juslin and Laukka in [13]. In Table 1 we can observe the results they obtained for the relationship between many musical features and some basic emotions.

Many of the features we observe in Table 1 can be extracted nowadays with existing technology. Of course, the methods for extracting them are still to improve, and most of them perform better for Western music, as that is the framework they have been developed for. For example, tempo can be quite well estimated by onset and beat detection, performing even better for songs with prominent percussion (as, for example, in techno music). A study on the performance of different tempo estimation methods can be found in [16]. Some

other features such as key and mode can be quite reliably obtained in Western music by using methods as Gómez's [17]. In Table 1, features that can be obtained from polyphonic signals are marked with one asterisk (*). Those which can be obtained from monophonic signals are marked with two asterisks (**). In both cases, we are talking about state-of-the-art techniques.

## 2.2. Automatic emotion and mood detection in music

When we talk about automatically classifying music by emotions or moods, people are somehow skeptic, as it seems that emotions are completely subjective (i.e., one song will contain or convey different emotions to each person). Indeed, emotion is something that has to do with human's perception and does not seem as objective as, for example, genres.

However, many studies demonstrated that musical emotions are not too subjective; within a given culture, these emotional responses to music are quite consistent among different listeners [13][18]. Actually, some studies suggest that, at least for basic emotions, this consistency is kept even among cultures [8]. Given this fact, we can think about creating systems that detect those emotions automatically from the music having satisfying results for most users.

Most current methods are based on feature extraction and statistical analysis of these. There are, however, some differences between these current approaches. Many consider a categorical representation with few basic categories [19-22], while some others as Wieczorkowska [23] use multiple non-mutually exclusive labels (adjectives). This latter approach is more difficult to validate due to the use of many categories.

| Musical Features | Happiness (1) | Sadness (3) | Anger (2) | Fear (2) | Tenderness (4) |
|---|---|---|---|---|---|
| Tempo* | Fast, small variability | Slow | Fast, small variability | Fast, large variability | Slow |
| Mode* | Major | Minor | Minor | Minor | Major |
| Harmony* | simple and consonant | dissonant | atonality, dissonant | dissonant | consonant |
| Loudness* | medium-high, small variability | low, moderate variability | high, small variability | low, large level variability, rapid changes | medium-low, small variability |
| Pitch** | high, much variability, wide range, ascending | low, narrow range, descending | high, small variability, ascending | high, ascending, wide range, large contrasts | low, fairly narrow range |
| Intonation** | rising | flat, falling | accent on tonally unstable notes | - | - |
| Singer's formant** | raised | lowered | raised | - | lowered |
| Intervals** | perfect 4th and 5th | small (minor 2nd) | major 7th and augmented 4th | - | - |
| Articulation** | staccato, large variability | legato, small variability | staccato, moderate variability | staccato,large variability | legato, small variability |
| Rhythm* | smooth and fluent | ritardando | complex, sudden changes, accelerando | jerky | - |
| Timbre* | bright | dull | sharp | soft | soft |
| Tone attacks** | fast | slow | fast | soft | slow |
| Timing variability* | small | large (rubato) | small | very large | moderate |
| Vibrato** | medium-fast rate, medium extent | slow, small extent | medium-fast rate, large extent | fast rate, small extent | medium fast, small extent |
| Contrast between long and short notes** | sharp | soft | sharp | - | soft |
| Micro-structure* | regularities | irregularities | irregularities | irregularities | regularities |
| Others | | pauses | spectral noise | pauses | accents on tonally stable notes |

Table 1. Frequent musical features and their relation with basic emotion categories (from [13], as cited in [2]). One asterisk (*) means that the feature can be extracted from polyphonic audio; two asterisks (**) mean that it can be extracted from monophonic audio with state-of-the-art techniques. In parenthesis is the quadrant number in Russell's dimensional space (Fig. 2).

Some authors such as Li and Ogihara [24] use a similar method (feature extraction and Support Vector Machine (SVM) classifier training) to the one we explain in detail by Laurier et al. [20], but they use a high number of categories (13) that causes the classifier not to give convincing results (0.32 average

precision). Apparently, these results are due to the small dataset labeled by just one person and the large number of categories itself. However, Mandel's work [25], using active learning in order to reduce the needed size for the dataset, could be used to increase the number of categories without dramatically decreasing accuracy. This is a key idea in our work, and we will come back to it in Section 0.

We now explain the feature extraction and statistical analysis stages, with a special focus on the method developed in the MTG-UPF [20] (see Fig. 3), as it is the base system that we try to improve. It is based in a supervised learning approach. More details are given in Sections 2.2.1 and 0.

An important thing to mention is that, in this approach, we want to independently decide about each of the labels. Accordingly, a binary decision is taken for every label. This means that every song will be classified as "happy" or "not happy", and at the same time as "sad" or "not sad" and so on for every label. This way, the output that the classifier will give for each song will be a degree of weight for each mood (e.g., 0.3 happy, 0.6 sad, 0.2 aggressive…).

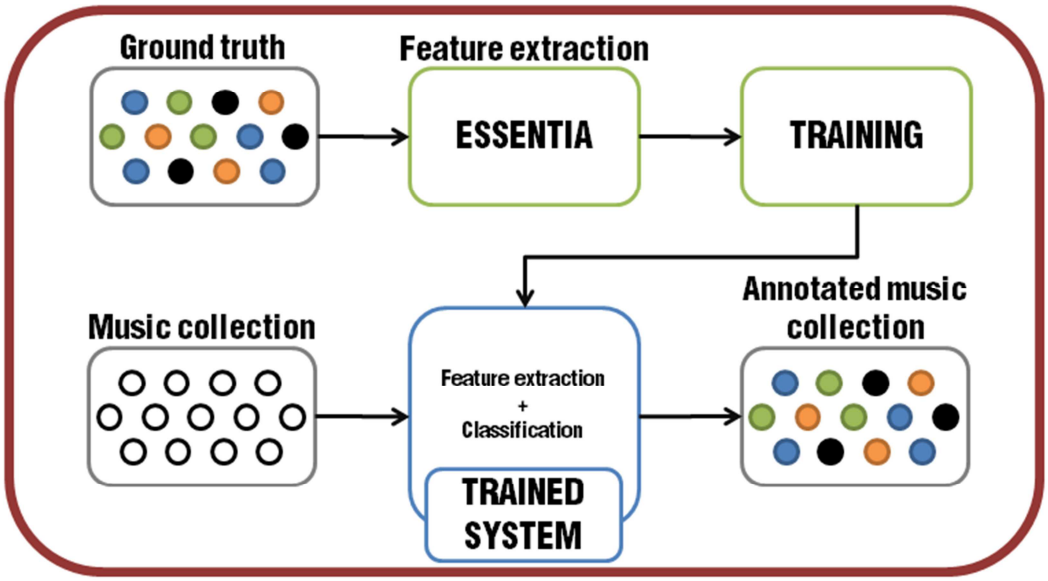



Fig. 3. Schema of the supervised learning approach (adapted from [20]). The classifier is trained with a set of labeled examples. Once it is trained, it can classify any new song.

### 2.2.1. Feature extraction

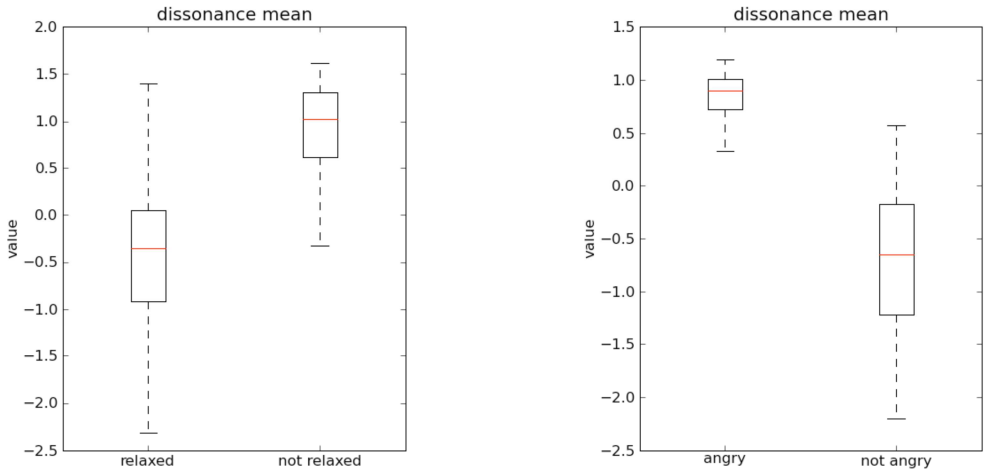
As mentioned in previous sections, we try to classify music by mood just using audio content information. In order to do so, we are interested in getting as many mood-related audio features (see Table 1) from the signal. It is possible now to extract a big number of features with existing state-of-the-art software, such as Essentia[1].

These features are able to capture characteristics of the signal that directly relate to the musical content. Table 2 contains a list of some of the musical features that we extract, classified according to the musical characteristics they relate to (timbre, tonal or rhythm). Many of these features are good for mood classification, according to how much they change among different moods. For a detailed description of the most important features and their relation to moods, see [20]. Fig. 4 shows how "dissonance mean" feature changes for "relaxed" and "not relaxed" and "angry" and "not angry" categories in this work by Laurier et al.

| | |
|---|---|
| Timbre | Bark bands, MFCCs, pitch salience, HFC, loudness, spectral: flatness, flux, rolloff, complexity, centroid, kurtosis, skewness, crest, decrease, spread |
| Tonal | dissonance, chords change rate, mode, key strength, tuning diatonic strength, tristimulus |
| Rhythm | bpm, bpm confidence, zero-crossing rate, silence rate, onset rate, danceability |

Table 2. Overview of extracted audio features classified by musical characteristic they relate to.

[1] http://www.mtg.upf.edu/technologies/essentia




(a) (b)

Fig. 4. Box-and-whisker plots of standardized dissonance mean for the (a)"relaxed" and "not relaxed" and (b)"angry" and "not angry" categories [20].

### 2.2.2. Classification (statistical analysis)

The very basic idea behind this stage is to let the system to learn which values of the extracted features define every group. More concretely, the classifier is trained with a dataset of labeled examples for it to learn a statistical mapping that models the problem. For example, it could automatically learn that "angry" songs are more likely to have a high dissonance than "relaxed" songs.

Many methods have been developed to achieve this task (SVMs, decision trees, k-Nearest Neighbors (k-NN), Logistic Regression, Gaussian Mixture Models (GMMs)…). Moreover, these methods are already implemented in existing software like WEKA[2]. MTG-UPF technology Gaia can be also used for statistical treatment of data, and it is especially developed to work with features extracted by Essentia. One important step in this task is the *dimensionality* reduction, which we do using Principal Component Analysis (PCA). SVMs have shown good

[2] http://www.cs.waikato.ac.nz/ml/WEKA/

performance both in Mood Detection [26] and Active Learning [27] tasks. We now explain both of these techniques (PCA and SVM) with more detail.

#### 2.2.2.1. **Principal Component Analysis (PCA)**

Principal Component Analysis (PCA) is one of the most used dimensionality reduction techniques in machine learning. Its goal is to reduce the dimensionality of data with minimal loss of information. The main idea is that, starting from a dataset of possibly correlated numerical variables $\{x_1, x_2, \dots, x_p\}$ it can be described with a smaller set of variables (called Principal Components) which are linear combinations of the original ones.

The first Principal Component accounts for as much of the variability in the data as possible and each succeeding Component accounts for as much as the remaining variability as possible. Here, the *covariance method* for PCA [28] is explained.

In order to understand this method, some statistical definitions must be provided.

The **variance** ($s^2$) of a single variable $x$ of size $n$ is obtained by:

$$s^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Being $x_i$ each of the elements in $x$ and $\bar{x}$ the **mean**.

The **standard deviation** ($s$) is just the square root of the **variance**, and it is the average distance from the mean of the dataset to a point.

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Both standard deviation and variance operate in one dimension. When dealing with multidimensional problems **covariance** is calculated, as it measures how much the dimensions vary with respect to each other. It tells how two variables vary together and it is calculated as:

$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n}$$

Note that the covariance of a variable *and itself* is the same as the variance. For a set of variables (features) $\{x_1, x_2, \dots, x_n\}$, the **covariance matrix** can be built. It is a matrix in which each element is the covariance between each pair of variables. It is symmetric and it has the variance of each variable in the main diagonal.

$$cov(X) = \begin{pmatrix} cov(x_1,x_1) & cov(x_1,x_2) & \cdots & cov(x_1,x_n) \\ cov(x_2,x_1) & cov(x_2,x_2) & \dots & cov(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(x_n,x_1) & cov(x_n,x_2) & \cdots & cov(x_n,x_n) \end{pmatrix}$$

Finally, we define the **eigenvectors** and **eigenvalues** of a square matrix, which are also used in this method. Eigenvectors are those which, after being multiplied by the matrix remain proportional to the original vector or become zero. The eigenvalue is the factor by which the eigenvector is changed when multiplied by the matrix.

More concretely, if $A$ is a square matrix, a non-zero vector $C$ is called an **eigenvector** of $A$ if and only if there exists a number $\lambda$ such that $AC = \lambda C$. $\lambda$ is an **eigenvalue** of $A$.

An important property of eigenvectors for us is that they are all orthogonal, which means that the data can be expressed in terms of those eigenvectors instead of the usual axes.

The already mentioned *covariance method* follows these steps:

1- Data organization

Data vectors can be placed in columns with dimension $J$ features x $I$ objects, forming a matrix $X$.

2- Get the mean vector and 'center' the data

First, the mean is subtracted from each column. This is equivalent to getting the distance from the mean (i.e. the variance). Let this new matrix be **B**.

3- Get the covariance matrix $C$

Following the explained formula, $C$ is obtained as

$$C = \left(\frac{1}{I} - 1\right) B'B$$

4- Calculate eigenvector and eigenvalues of the Covariance Matrix

The eigenvectors of the covariance matrix define those directions which characterize the data.

5- Choose components and form a feature vector

The eigenvector with the highest eigenvalue is the *principal component* of the data set, meaning that it represents the most significant relationship between the data dimensions.

Once the eigenvectors are found, they are ordered from highest to lowest eigenvalue. Here comes the dimensionality reduction. If all the eigenvectors are used there is no such, but discarding those with the lowest eigenvalue should cause a small information loss.

With the selected eigenvectors a *feature vector* $W$ containing them is created.

6- Project data in the new basis

Finally, the new transformed matrix $S$ is obtained from the original $X$ as

$$S = WX$$

The more elements are discarded forming $W$, the bigger the dimensionality reduction will be, but the more information will be lost.

#### 2.2.2.2. **Support Vector Machines (SVM)**

Support Vector Machines [29] is a common classification algorithm that has been widely used in MIR research. The idea behind SVMs is to try to find the hyper-plane separating two classes of data that will generalize best to future data. To explain it more concretely, we will define the problem:

- We have $l$ training data $x_1, \dots, x_l$ which are vectors in a feature space $X \epsilon R^N$
- We have their labels $y_1, \dots, y_l$ where $y_i \epsilon \{-1,1\}$ (e.g., happy - not happy…)

Assuming that the samples are linearly separable, the *best* hyper-plane would be the one that separates them in such a way that the distance from it to the closest points is maximum. This optimal hyper-plane could be characterized by a weight vector $w$ and a bias, such that:

$$w^T x + b = 0$$

and the decision function for classifying some unlabeled point $x$ is

$$f(x) = \text{sign}(w^T x + b)$$

being

$$w = \sum_{i=1}^{N} a_i y_i x_i$$

$N$ is the support vector number, $x_i$ is the support vector, $y_i$ is the class to which $x_i$ belongs and $b$ is set by the Karush Kuhn Tucker conditions [30]. $a_i$ maximizes the function defined by

$$L_D(a) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i,j} a_i a_j y_i y_j x_i x_j$$

subject to

$$\sum_{i=1}^{N} a_i y_i = 0 \quad \forall i, a_i \geq 0$$

The Support Vector are the subset of $a_i$s which are non-zero. However, most of the times the data we work with cannot be linearly separated [27]. The solution in this case is to map the input space into a higher dimensional one in which a new optimal hyper-plane can be calculated.

Given that the only calculations that we need to perform are dot products, we do not need to know the function $\Phi(\mathrm{x})$ that defines the mapping from our space $R^N$ to the goal space $Q$. What we actually use are the so-called 'kernel functions' $(K)$, which take the vectors in input space to compute the dot product in the feature space $Q$:

$$K(x_i, x_j) = \Phi(\mathrm{x_i}) * \Phi(\mathrm{x_j})$$

Frequently used kernel functions are shown in Table 3.

| **Kernel function** | $\boldsymbol{K(x,y)} =$ |
|---|---|
| Linear | $x \cdot y$ |
| Polynomial | $(\gamma(x \cdot y) + coef0)^d, \gamma > 0$ |
| Radial Basis Function (RBF) | $\exp(-\gamma\|x - y\|^2), \gamma > 0$ |
| Sigmoid | $\tanh(\gamma(x \cdot y) + coef0)$ |

Table 3. Common SVM Kernels

## 2.3. Active learning

In this Section, we explain what is the idea behind active learning (first introduced by Lewis et al. [31]), we review some literature and we discuss how it suits our problem. The main source from which we got the information is the

'Active Learning Literature Survey' by Settles [32], where a comprehensive state-of-the-art active learning review updated in January 2010 can be found.

### 2.3.1. What is active learning?

In a very general way, the main idea behind active learning is that the accuracy of a machine learning algorithm can be improved with fewer labeled examples if it is allowed to choose the data from which it learns. In order to do so, the system may pose *queries* (unlabeled data instances) to be labeled by an *oracle* (in our case, a human annotator). It is useful for problems where unlabeled data is easy to obtain, while labels are not.

Fig. 5 illustrates the pool-based active learning process (explained later in this Section). The system is first trained with a small set of labeled samples $\mathcal{L}$. Then, it selects some instances from an unlabeled pool $\mathcal{U}$ to be annotated by the *oracle*. Once they are annotated, the model is updated with this information and ready to start the cycle again.

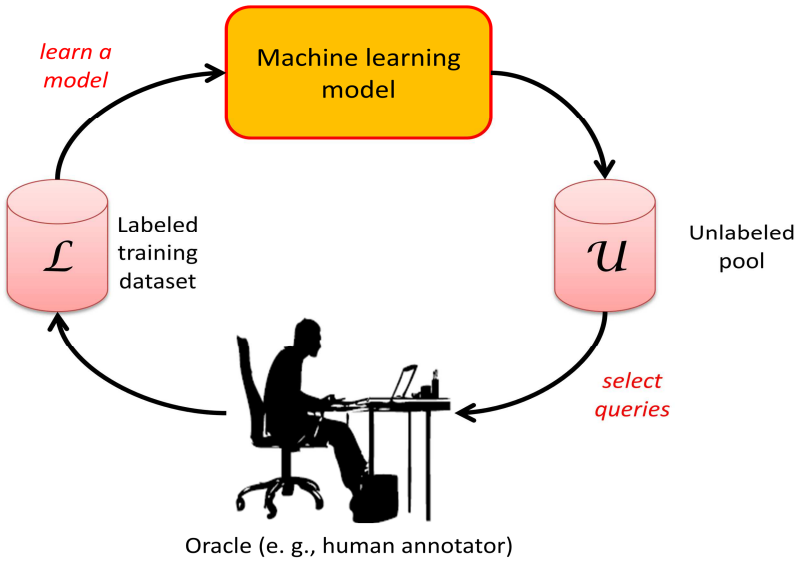



Fig. 5. Pool-based active learning process (adapted from [32])

According to [33], there are three factor that should always be considered when facing an active learning problem. These three factor refer to the features that the chosen instance to be labeled should have:

- **Represented in the training set.** If a given point is already represented by the existing training set, asking the oracle to label it is likely to give redundant information. Active learning methods should favor items which are not represented in the training set.
- **Representative of the test set.** The selected item should be representative of the test set in such a way that they give information about the other unlabeled items.
- **Results.** The selected point should, after being labeled, improve the accuracy of the predictions that the system does. In this sense, there are many different strategies which are explained later in 2.3.3.

### 2.3.2. Scenarios

Depending on how the system asks queries, there are three main settings:

#### 2.3.2.1. **Membership Query Synthesis**

In this approach [34], the learner synthesizes queries, meaning that it requests labels for any unlabeled instance in the input space, usually being instances that the learner generates. We will not go into details, as this approach is not suitable for problems where the oracle is a human annotator (as in our case).

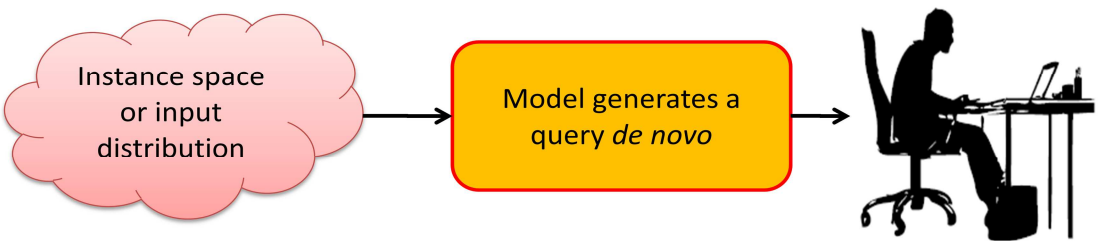



Fig. 6. Membership query synthesis (adapted from [32])

#### 2.3.2.2. **Stream-based selective sampling**

In *selective sampling* [35], the system samples an unlabeled instance from the actual distribution and then decides to query or not. This means that the sample selection is "blind", we just take samples from the unlabeled dataset and decide for each one if it is worth querying.

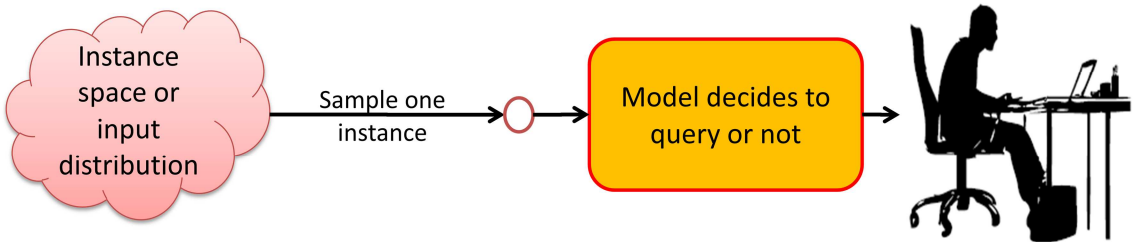



Fig. 7. Stream-based selective sampling (adapted from [32])

#### 2.3.2.3. **Pool-based sampling**

This last approach, which was already illustrated in Fig. 5, is the one that best suits our problem. The assumption in *pool-based sampling* [31] is that there is a small set of labeled data $\mathcal{L}$ and a large pool of unlabeled data $\mathcal{U}$ available. Usually, the system makes some kind of informativeness measure to rank all the instances in $\mathcal{U}$. The main difference with *selective sampling* is that, instead of scanning sequentially through the data, the learner evaluates the entire collection of unlabeled samples.

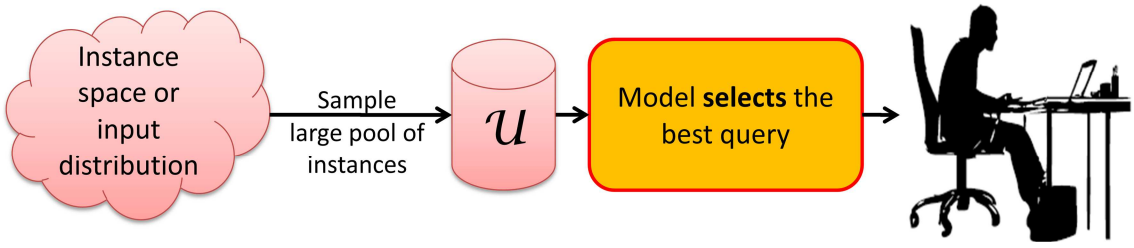



Fig. 8. Pool-based sampling (adapted from [32])

### 2.3.3. Query strategy frameworks

Independently of which sampling setting is used, it is necessary to perform an evaluation of the informativeness of unlabeled instances. This is not a trivial problem, and different strategies have been defined in the literature to address it. Following the same notation as in [32], we will refer to the most informative instance according to a certain algorithm A as $x_A^*$.

#### 2.3.3.1. Uncertainty Sampling

This is the most straightforward and commonly used method [31]. It is quite suitable for probabilistic learning models. If, for example, the classification is binary (as in our case, as we perform this kind of classification for each mood label), the instance that would be queried would be the one which probability of being positive is closest to 0.5 (total uncertainty).

In a more general way (with more class labels), we would consider the most informative sample (or the most uncertain)

$$x_{LC}^* = \text{argmax}_x \, 1 - P_\theta(\hat{y}|x),$$

where $\hat{y}$ is the class label with the highest posterior probability under the model $\theta$.

There are some refinements on the method with the same idea that are not really relevant to mention in this work. It is relevant though to mention that some works use uncertainty sampling with non-probabilistic classifiers. For example, some relevant works in this direction are Lewis' [31] (using decision trees classifiers), Fujii's [36] (with nearest neighbors classifiers, letting each neighbor to "vote" on the class label of $x$) or Tong's [37] (using SVMs for text classification). Mandel [25] also uses SVM active learning, and his work is relevant for us as well because he uses it for music retrieval.

Chen [38,39] used relevance feedback for SVM classification. In relevance feedback, it is the user who selects the relevant samples according to subjective judgment. Although it is not exactly the same as our case, some concepts are still applicable and interesting for us.

#### 2.3.3.2. **Query-By-Committee**

The Query-By-Committee (QBC) algorithm [40] is based on a different idea. The Committee $C$ is a "collection" of models $C = \{\theta^{(1)}, \dots, \theta^{(C)}\}$ which are all trained with the same current labeled set $\mathcal{L}$, but that represent different hypotheses.

Each of these members of the committee (models) "votes" on the labeling for query candidates. The most informative sample will be the one with the highest disagreement between models. The idea is illustrated in Fig. 9. The three blue lines represent three models that suit the classification of the data according to the labeled dataset (represented by the green triangles and red circles, which represent two different classes). The squares represent unlabeled samples from which the learner has to decide which is the most informative. According to the QBC algorithm, the yellow square would be taken as the most informative sample, it is the one with the smallest agreement between different models (for this illustrative example, the other two unlabeled samples, black squares, would have total agreement). Different definitions on how to measure this disagreement can be found in [32], pg. 17.

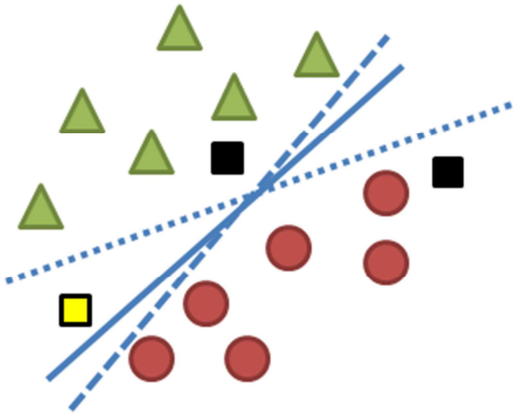


Fig. 9. Illustration of the QBC algorithm. Blue lines represent different models that suit the classification according to the labeled set of two classes (green triangles and red circles). Squares are unlabeled samples, from which the yellow one would be taken as the most informative as it is the one with the smallest agreement between models.

#### 2.3.3.3. **Expected Model Change**

The idea in this case, which can be understood from the method's name, is that the sample which will be considered as the most informative is the one that would cause the greatest change in the model in case its label was known.

A method for it was proposed in [41], using the "expected gradient length" (EGL) for discriminative probabilistic model classes. The gradient represents the "magnitude of change" of the model, and it is directly related to how much the value of the parameters is affected.

Intuitively, this framework detects the samples that are likely to have the highest influence in the model, without taking care of the resulting label after the query. Apparently, it has shown quite good results in empirical studies, but it requires expensive computation. Moreover, an unusual large value of some feature in the unlabeled sample can make the system over-estimate it.

#### 2.3.3.4. **Expected Error Reduction**

In this case, the goal is not to measure how much the model will change, but how much the generalization error is likely to be reduced. The method is based on taking the current model $\theta$ and dataset of labeled samples $\mathcal{L}$, adding a sample from the unlabeled dataset $\mathcal{U}$ and calculating the expected error (or risk) for each possible query label. It is easy to see that the computational cost of this method is huge, and it increases a lot as the size of $\mathcal{U}$ increases.

The method was proposed by Roy and McCallum in [42], and some interesting improvements have been done by authors like Zhu [43], who combined the framework with a semi-supervised learning approach.

#### 2.3.3.5. **Variance Reduction**

*Variance Reduction* is an alternative to *Error Reduction* that uses the results from [44], which state that the expected future error (that the previous method tries to minimize) can be decomposed in three terms: *noise* (the variance of the true label given a instance), *bias* (error due to the model class itself) and *variance*. This variance can be computed more easily than the *expected error*, and minimizing it guarantees minimizing the error as well, because the other two terms do not depend on the learner. For further details on different implementations of *variance reduction*, see [32], pgs. 21-15.

#### 2.3.3.6. **Density-Weighted Methods**

So far we have seen methods that try, using different concepts, to get the most uncertain labels in order to use them for queries, assuming that they will be the most informative (in the sense that they will be the ones that will make the system learn the most about the problem). However, it is not always the case that the most uncertain instance is the most informative. Fig. 10 illustrates the

case where this could happen. We have a binary linear classifier in which the green triangle and the red circle represent labeled data. Squares represent unlabeled data. Using uncertainty sampling (i.e. the methods we have seen so far), the red square would be taken for querying, as it falls exactly in the decision boundary. However, we could prefer to choose the yellow one as it is the most informative of those which *represent the actual underlying structure*.

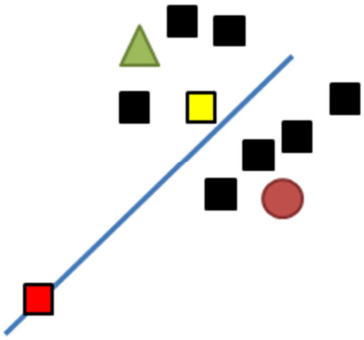


Fig. 10. Illustration of the problem that can appear using uncertainty sampling. There is a binary linear classifier. Green triangle and red circle represent labeled instances. Squares represent non-labeled instances. Uncertainty sampling would take the red square for query as it falls on the decision boundary, but choosing the yellow one is likely to give more information as it represents better the underlying structure.

Different approaches use this concept. For example, Fujii et al. [36] achieve this goal by selecting queries that are least similar to the labeled dataset $\mathcal{L}$, but most similar to the rest of unlabeled instances in $\mathcal{U}$. Mandel et al. [25], whose is explained with more detail in Section 2.3.4, use *angle diversity* (proposed in [45]), which balances the closeness to the decision boundary with the coverage of the feature space. Wang et al. [27] propose selecting the samples not just according to their closeness to the decision boundaries, but also according to how "scattered" they are in the space. This relates not just to the concept discussed here, but also with choosing non-redundant samples. Anyway, Wang's approach is also further explained in Section 2.3.4.

### 2.3.4. Active Learning and Music Information Retrieval

In this Section, we have seen some basic concepts about *active learning* as a tool for selecting informative unlabeled samples to be queried. Of course, this method has been used for different disciplines, as classification and, more generally, information retrieval appears in many fields. However, we will now mention some works where it has been used for Music Information Retrieval (MIR), so that we can understand better how it suits our problem.

Mandel et al. [25] demonstrated that active learning techniques can be used for music retrieval with quite good results. Concretely, they classified songs according to their *style* and *mood*, being able to perform the same accuracy with half as many samples as without using active learning to intelligently choose the training examples.

This suggests that active learning can be useful for us taking into account that:

- The more labels we add, the higher is the necessity to make them user-dependent, as the general agreement among users for mood labels seems to be high just for basic emotions [24].
- When dealing with user-tailored systems, we should try to avoid asking the user to label too many examples. Active learning could help us to get the most valuable information from the user, training the system with fewer examples.

In the already mentioned work by Mandel et al. [25], SVM active learning is used to classify music according to *mood* and *style*. One of the concepts that is interesting for our work is that they use active learning with the idea of "learning a classifier for each query […] customized to each user". Considering the time the time that it takes to the user to label a new song that is presented to him by the

system, the goal of active learning is to minimize as much as possible the number of songs the system needs to perform well. According to their results, using active learning allows the system to get the same accuracy as without it with half as many labeled samples (or, with the same number of labeled samples, increasing the accuracy in an average 10%). The features they work with are based on Gaussian Mixture Models (GMMs) of Mel-Frecuency Cepstrum Coefficients (MFCCs). Maybe the low precision they achieve in mood classification (below 0.5 mean precision in top 20 results) is due to the use of just this feature. As we have seen in Table 1 (page 23), the relation between musical features and emotions is quite complex and requires considering many musical aspects.

The problems they observe have to do with the strategies for querying. If a small set of examples is used at the beginning, the system has trouble identifying which are the most discriminative to label. What they suggest to solve this problem is to make a "special" first training round with a larger set of labeled samples. This would allow the system to detect better which unlabeled samples are the most informative in order to fully take advantage of active learning techniques.

Wang et al. [27] propose a strategy for multi-samples selection for SVM active learning. Their assumption is that, in music retrieval systems, it is necessary to present multiple samples (i.e. to make multiple queries) at each iteration. This is because the user could very likely lose patience after some time if just one sample is presented to him to label at each iteration. Their strategy is based on selecting samples that are not just close to the SVM decision boundary, but also as scattered as possible in the feature space, thus reducing redundancy. With this purpose, they define a "distance diversity" measure, defined as the minimum distance between a sample set $S$ (which is the sample of sets to be queried). The

higher this distance is, the more scattered they are in the feature space (i.e. the lower is the redundancy among the samples).

They also include a correction to avoid taking outliers (which has to do with the concept of *Density-Weigthed Methods* previously explained in this section). Basically, what they do is to include a term that weights the density of the region where a sample is, by computing the distance to its 10 closest samples. In their work, they do different experiments in order to evaluate the performance of their method and there are some conclusions which may be relevant to mention:

- The initial set construction strategy and the different kernel functions have an important influence on the performance. For their experiments, the combination of RBF kernel function and K-means selection of initial set is the one that performs the best.
- There are two parameters that have to be carefully tuned: the size of the initial set (that they set to 10) and the number of samples selected at each round for labeling. For this second one there is a trade-off, because smaller sizes require more rounds of feedback to achieve some precision level.
- Finally, they compare the precision of their sample selection method with simple selection (just distance to boundary), *angle diversity* [45] and random selection. The method they propose outperforms these three selection strategies.

## 2.4. Conclusions

During the present chapter we have seen that mood detection is a problem that involves musicology, Music Information Retrieval, machine learning, etc. We have seen that mood detection systems are evolving and getting good results for basic moods, but the complexity of emotions in music needs more accurate explanation. In this context, the less basic the emotions are the less agreement is

expected to be found among users, something that may require customized systems in which the classification is done according to the user. As getting labels from a specific user to train the system is likely to be expensive, being able to reduce the number of songs which are required for good classification may help achieving this purpose.

Given that our work is built on top of Cyril Laurier's [1], we focus on extending the current system to new mood labels following the methods seen in section 2.2. Moreover, we explore the usefulness of active learning techniques for these new labels and for the already existing ones in order to reduce the size of the training sets without reducing accuracy. For this purpose, we take the approach presented by Wang *et al* (explained in this same chapter in section 2.3.4) as a starting point.

# 3. Methodology

This chapter explains the methodology that has been used in order to achieve the goals of the present work. First, we explain how the new mood labels have been selected. Afterwards, we discuss the criteria that we have applied to create the initial datasets and the process to filter them by listeners' validations. Then, we study how we evaluate the validity of these datasets and the performance of different classification algorithms over them. Finally, we see the process for implementing and testing the active learning techniques, which implies defining some experiments and criteria for improving the algorithm depending on the results of these experiments.

## 3.1. Label selection

The mood labels that we finally have chosen to implement are the following:

- **Mysterious**
- **Humorous**
- **Triumphant**
- **Sentimental**

They have been selected in order to fulfill two main criteria:

- They have a certain **social relevance**. Taking into account that the context in which our work would be useful has to do with Recommender Systems (RS), we tried to use mood labels which are being used by users in different contexts: social networks, music compilations, online playlists… In Table 4 we can see a summary of the usage of these labels (and synonyms of them retrieved in Word Reference[3]) by users of last.fm. Fig. 11 shows the number of songs that are labeled with some tags. All those that appear in blue are tags extracted from the 'Most used tags' chart in lastfm[4]. As we can see, the selected tags (together with their synonyms) are not as relevant as the most used ones, which are related to most popular genres (rock, pop, electronic…). However, they are as relevant as some other tags that still appear among the list of most used tags (cool, celtic, latin, anime…).

| | Users who used the tag | Songs tagged with the label |
|---|---|---|
| Mysterious | 24546 | 88461 |
| Humorous | 46664 | 165835 |
| Triumphant | 10077 | 55479 |
| Sentimental | 71638 | 402890 |

Table 4. Usage of the selected mood labels in last.fm. The data correspond to the date of publication of this document.

- Their capacity to **improve the emotional representation of music** given by current mood detection systems. By this, we mean that the labels to be added have to add new information, so they should not have a very close semantic relationship with the current *happy, sad, aggressive* and *relaxed* mood tags.

[3] www.wordreference.com

[4] http://www.last.fm/charts/toptags

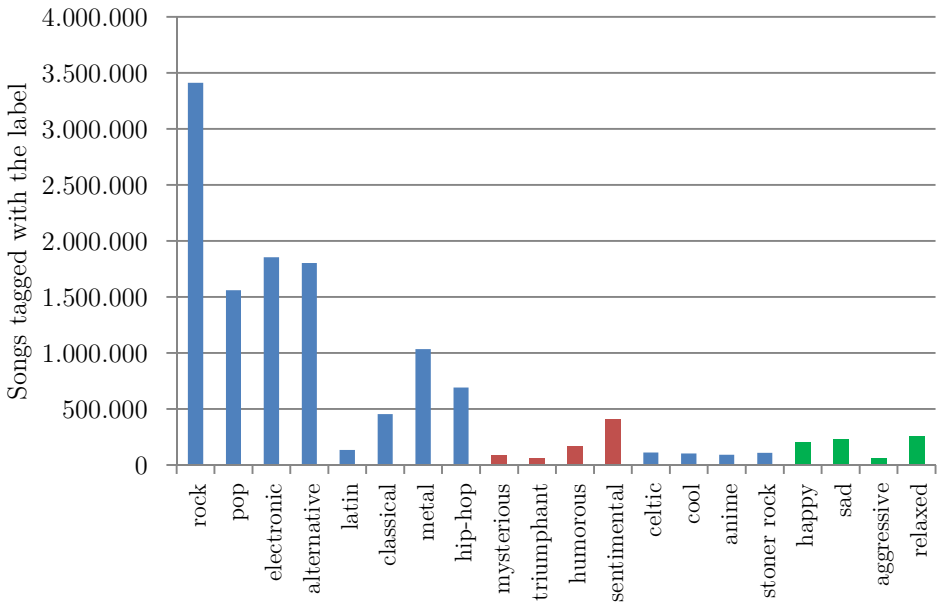



Fig. 11. Comparison of different tags in lastfm according to the number of songs which are labeled with them. The data correspond to the date of publication of this document. In red, new moods. In green, already implemented moods.

## 3.2. Creating the audio datasets

As explained during the whole document, in our problem we perform a binary classification for each of the mood labels, this is, given a certain song, the system has to decide for a given category whether if the song belongs or not to it.

Training a system which is able to perform such a classification requires then, not just collecting songs that belong to that category, but also to the complementary one (i.e. songs which do not have that particular mood). Also, it is important to have a dataset in which both "category" (e.g. *mysterious*) and its complementary (e.g. *not mysterious*) datasets are representative enough of the mood information, which means collecting songs from different genres, timbres, etc. This is crucial in order to avoid the classification being biased by musical characteristics that are not related to mood. The main features of the created datasets are:

- 30 seconds audio MP3 files (<= 128kbps to avoid wrong feature extraction).
- All the songs have no lyrics in order to avoid listeners' validation being biased by emotions expressed in them.
- High coverage of genres and artists (not more than 1 song per artist/composer).
- 150 to 200 files for each category and its complementary.

Songs have been chosen using tags in lastfm[5]. In lastfm tags are created by users, who simply add whatever they think that could describe a certain song. We looked for instrumental songs (or songs with long instrumental excerpts) that were tagged with each of the moods (or synonyms of them). Then, the 30 seconds excerpts were chosen avoiding parts of songs in which there are sudden changes on many features, as that can turn into wrong feature extraction. For example, we would not take a music excerpt in which there is a steady note during 28 seconds and, suddenly, a *fortissimo* note plays. This kind of excerpts, of course, could be interesting in some other approaches dealing with expectations or surprise in music.

In order to create a valid ground truth to carry out our final experiments, we performed a listeners' validation to avoid the datasets being biased by wrong online labels or even personal criteria of the authors (which also was used on the initial datasets creation).

## 3.3. Listeners validation

In these listeners' validation experiments, 6 listeners were presented with playlists containing the 150 to 200 30 seconds excerpts of the songs which previously had been selected as candidates for each category. From these

[5] www.last.fm

playlists, they had to remove those that they did not consider to belong to the corresponding category. In order to check the correct feedback from the users, some patently wrong examples (in principle not likely to belong to the target category) were also included in the playlists. Luckily, all the users correctly removed these spurious examples from the playlists, so we can consider that they performed the task by correctly understanding it and paying attention to the music (i.e. their judgments are high quality information).

We decided to keep those songs which were accepted by at least **60% of the users**, discarding the rest for posterior experiments. Some other works as [2], for example, kept those songs which were not discarded by at least one user, as the selected candidates had already been labeled on the internet. Our approach somehow gives more importance to the *wisdom of the few* (as opposed to the *wisdom of the crowd*) [26], considering that there can be many wrong labels on social networks.

## 3.4. Classification experiments

This step has two main goals: checking the validity of the ground truth and exploring the accuracy of different classifiers (and their parameters) in order to work with them afterwards.

Although the focus of this work is not on the feature extraction part, we must keep in mind that everything that comes next depends on the robustness of the features extracted from the audio. We use the feature extractor of Essentia (an MTG technology) to get around 900 low-level, rhythmic and tonal features related to musical characteristics which, at the same time, are likely to be related to moods as explained in 2.1.4.

The first experiments are done using WEKA[6]. This software allows us to perform several experiments with our datasets, previously generated using the Essentia extractor (to obtain descriptors) and Gaia (to merge those descriptors into a single dataset file). The process that we follow in WEKA to achieve the mentioned goals is the following:

1. Visual inspection of features to remove outliers (i.e. songs which values do not make sense or are completely separated from the rest).
2. Attribute selection. This is a supervised process in which the most correlated features to a category are selected. We use the following parameters:
    a. Evaluator: cfsSubsetEval. It evaluates the features considering individual predictability and global redundancy.
    b. Search: BestFirst. Searches the best features following a descending order.
3. Principal Component Analysis (PCA): it performs a linear and weighted combination of features to reduce the dimensionality of data, being each combination a "component". After this step we get around ***dataset_size* / 20** linearly merged components. We do so to avoid over-fitting the dataset. As stated in [33], this rule of thumb allows us to make our model more general. The parameters we use in WEKA are:
    a. Maximum attributes: -1. This means there is no limit in the number of features taken for creating each component.
    b. Normalize: True. Normalizes input data.
4. Normalization of all the descriptors, which is necessary to avoid having descriptors with bigger weight in the classification.

[6] http://www.cs.waikato.ac.nz/ml/WEKA/

5. Compensation of sizes for both categories. We want our classes (category, not-category) to have the same amount of instances to guarantee equal correct classification probability.

One this dataset filtering is done, we perform several classifications in order to study which classifiers give the best performance. We perform 10 runs of 10-fold cross validation experiments, taking different seeds at each run to avoid the results being biased by the dataset division made during cross-fold validation. It is important to use different classifiers in order to ensure that the results do not depend on one specific classification technique. More information about them can be found in [46] or [47] (with specific information for WEKA). The classifiers we use are the following, keeping the default parameters when the contrary is not specified:

- Zero Rule Classifier (**0R**): A classifier that decides according to the majority class. It is equivalent to a random guess, so actually it is used as a baseline that should be outperformed by any other classifier.
- One Rule Classifier (**1R**): Classifier that takes a single feature, which is the one that has the minimum prediction error. It is the one that discriminates the most between classes. It is interesting to compare it with more complex methods as it gives an idea about how worth it is complicating the problem (e.g. if a single feature is able to get a similar performance to the rest of methods, maybe it has no sense using them).
- Naïve Bayes (**Bayes**): Classifier based on Bayes' theorem. It is based on the assumption that there must be a given feature in a class completely unrelated to the presence of any other feature.
- K-Nearest neighbors (**IBk**): A classification algorithm which is based just on the proximity (vicinity) of points belonging to the same class. We arbitrarily set **5** as the number of nearest neighbors.

- Random Forest (**Forest**): Based on decision trees; groups of features are selected randomly at each node.
- Support Vector Machines: explained in detail in 2.2.2. We use two different kernel functions:
  - PolyKernel (**SVM-p**): Polynomial Function. Parameter E = 1
  - RBFKernel (**SVM-r**): Radial Basis Function. Parameter gamma = 0.125
- Linear Logistic Model (**SL**): Classifier that fits the features into a sigmoid curve or logistic function calculating the probability of a class to be predicted.

These classifiers are evaluated over every dataset before and after listeners' validation (to check if the validation improves the performance).

## 3.5. Active learning implementation and testing

During our work, we will study how two different Support Vector Machine active learning strategies can help us training the system with fewer instances compared to the case in which the samples are selected randomly. We perform the experiments with an algorithm which takes into account not only the proximity to the boundary, but also tries to maximize the distance among samples over the feature space, weighting as well the density around them to avoid taking outliers. This is the multi-sample selection strategy presented by [27], and previously explained in the present work in Section 2.3.4.

The datasets are built from the song collections built in this project but also from the collections by Cyril Laurier for the *happy, sad, aggressive* and *relaxed* moods. This is done with the intention of discarding problems that may appear in our own datasets, but also trying to generalize the behavior of the studied active learning techniques.

With this purpose, we take advantage of the Python bindings of Gaia[7] (written for easier database manipulation and experimentation), which allow us to "play" with our datasets and perform classifications. Taking into account that the scenario in which we would like our systems to work is Recommender Systems, we follow some steps in our experiments to simulate the interaction with a user. In this way, this is the experiment (represented also in Fig. 12) that we run for different active learning strategies (used as well in [27]):

1. Splitting the database randomly into a test and a training set, both with the same size.
2. Select a random sample from the training set as the seed (the song for which the user is looking for songs with the same mood).
3. If we are in the first round, select ITS-1 (Initial Train Size) samples plus the seed as the initial set for feedback. We choose them randomly. Otherwise, select EPI (Elements to add Per Iteration) samples according to the **sample selection strategy** (see details in 3.5.3).
4. According to the ground truth, automatically label the selected samples (simulating user relevance feedback). Add the labels to the labeled dataset and remove them from the training set.
5. Retrain the SVM model with the available dataset. Get precision and recall (see 3.5.1) over the test dataset.
6. Repeat 3-5 seven times to simulate user feedback.
7. Repeat 1-6 100 times to average and avoid being biased by the selected seed and initial random-selected training set.

[7] http://mtg.upf.edu/technologies/essentia

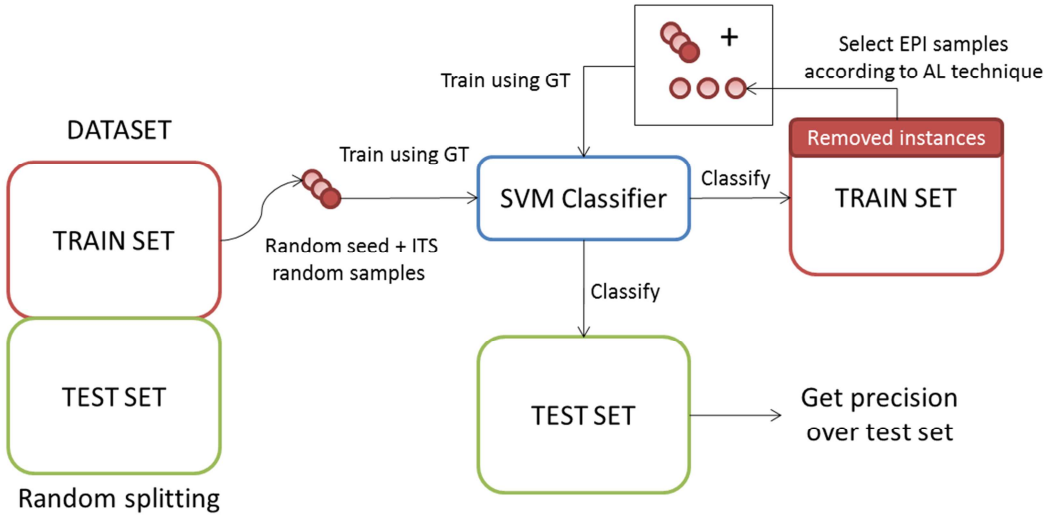



Fig. 12. Active learning experiments scenario. ITS: Initial Training Size; GT: Ground Truth; EPI: Elements Per Iteration; AL: Active Learning

### 3.5.1. Recorded results

In Step 5 of the presented algorithm we talk about "getting precision and recall". In Results, plots with both values are presented. *Precision* refers to the fraction of retrieved elements that are relevant. *Recall* is the fraction of relevant instances that are retrieved. In the context of our problem, we consider *relevant* those points that belong to the category for which we are looking for similar songs. More concretely, we get the values for precision and recall doing:

$$Precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Being *true positives* songs that were correctly considered to belong to the category, *false positives* songs that were wrongly considered to belong to the category and *false negatives* songs that were wrongly considered not to belong to the category.

Also, when we study the influence of each parameter on the overall results we talk about *F-measure*, which is a measure that combines precision and recall. The formula we use to get the *F-measure* is:

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

### 3.5.2. Dataset management

Before the algorithm is executed, we need of course to create points representing the audio files and perform some pre-processing in order to avoid having problems such as having non-valid features (constant values, variable length) or outliers.

At the beginning, we have a different file[8] for every song containing all the values that the Essentia extractor gets. This includes high-level, low-level, rhythm and *sfx* descriptors. From this point, we perform the following steps before running the algorithm:

- Merge all the files for a specific genre into a single database file. All the points are taken and a single file containing the values for all of them. More specifically, we use gaiafusion, a tool which creates a `.db` file which can later be used in Gaia, where any kind of operations can be done over the database (querying, modifying, filtering points, removing outliers, classifying…). This step already removes those descriptors that:
    - Have variable length (e.g. chords progression descriptor might have different length for different files).
    - Are constant (these are meaningless for classification).
    - Contain `NaN` or `Inf` values.

[8] Using Essentia, there is a `.sig` file for each song containing the values for the features.

- Feature selection. For our experiments, we discard high-level descriptors, so we remove them from the database. They are discarded because there is no expected relationship between them and the moods (e.g. female/male singer, instrumental/not instrumental…) [1]
- Normalize descriptors and reduce dimensionality using Principal Component Analysis (PCA). The number of components is different depending on the size of the dataset ($\approx$ dataset_size/20). By this we get two interesting advantages:
    - Avoid over-fitting the dataset using a *fair* number of components.
    - Reducing the number of dimensions results on a much faster performance of the algorithm.
- Compute the density around each point. This is one of the parameters that the active learning strategy uses to measure the *informativeness* of each point. It is explained with detail in 3.5.3 and, differently from the other two (distance to boundary and distance diversity) is computed just once at the beginning, having the same value during the entire execution.

### 3.5.3. Active learning strategies

In order to implement the distance to the decision boundary, the scikits-learn[9] library was used. Please see "Appendix: GAIA and active learning" for more details about this.

In this experiment explained at the beginning of this Section, step (3) mentions "select samples according to the sample selection strategy". This sample selection strategy is actually the *active learning*. At that stage, we select unlabeled samples to be labeled using the given ground truth (in a real scenario,

[9] http://scikit-learn.sourceforge.net/

to be labeled by the user). For our experiments we use two different active learning strategies:

- **Wang's multi-sample selection strategy** [27]. In this approach, multiple samples are selected in a way that they are not just close to the boundary (most uncertain/informative samples), but also representative of the underlying structure and not redundant among them. To fulfill these three criteria, three values are calculated on every iteration for each point:
    - the **distance to the decision boundary**,
    - the **distance diversity** and
    - the **density** around the sample.

  The first one is given by the own SVM classifier, which calculates the decision boundary ant tells the distance of each point to it. The diversity is calculated every time a new unlabeled sample **x** is selected as a candidate to be added to the current sample set **S**. It is defined as the minimum distance between samples in the current selected sample set **S** (the higher the diversity, the more scattered the set of samples are in space) and is calculated as

$$\mathrm{Diver}\,(S + x) = \arg \min_{x_i, x_j \in \{S+x\}} D(x_i, x_j)$$

  Where $D(x_i, x_j)$ is the distance between points $x_i$ and $x_j$

$$D(x_i, x_j) = \sqrt{\left(\Phi(x_i) - \Phi(x_j)\right)^2} = \sqrt{\Phi(x_i)^2 + \Phi(x_j)^2 - 2\Phi(x_i) * \Phi(x_j)}$$

  Which in terms of the kernel function is

$$D(x_i, x_j) = \sqrt{K(x_i, x_i) + K(x_j, x_j) - 2K(x_i, x_j)}$$

  The density around the sample, which is included to avoid choosing outliers as candidates, selects samples from the denser regions. An average distance $T(x)$ from a particular sample $x$ to its 10 closest neighbors is computed off-line as

$$T(x) = \frac{D(x_{j1}, x) + \cdots D(x_{j10}, x)}{10}, x_{j1} \neq \cdots x_{j10}$$

Once these values are obtained, the selected point is the one that minimizes (**distance_to_boundary − diversity + density**) and the process is repeated as many times as samples are to be added at the current iteration.

- **Modified Wang's multi-sample selection strategy.** This strategy is proposed as an option to solve the problems of uncertainty-based active learning strategies (mentioned in [33]), which are even clearer when small training sets are used. What is done is to perform exactly the same strategy just explained for half of the samples to be added, while the other half is actually selected from those furthest from the boundary. This can seem contradictory with the explained idea behind uncertainty-based active learning, in which most uncertain samples are chosen, but actually this method just tries to correct when necessary the initial assumptions of our classifier, in such a way that we do not get "more certain about the wrong thing".

### 3.5.4. Experiments and analysis

Many parameters (initial training size, elements to add per iteration…) are involved in the scenario explained in this section. From previous works ([25], [27]) we know that they all have an influence on the results. Trying to study this influence of the mentioned parameters, we performed several experiments for each of the datasets including:

- 3 values for initial training dataset size: 2, 5 and 10.
- 3 values for the number of elements to be added on each iteration: 2, 6 and 8.

- 4 different combinations of weights for the values that are calculated for the active learning strategy (explained in 3.5.3): [1,1,1], [4,1,1], [1,4,1] and [1,1,4] (each number is the weight for *distance to boundary, diversity* and *density* values respectively).
- 2 active learning approaches (explained in 3.5.3): Wang's and modified Wang's.

The different combinations of these values lead to $3 \cdot 3 \cdot 4 \cdot 2 = \mathbf{72}$ experiments with different results for each of our 8 datasets (**576** experiments in total which are performed 100 times each to get an average behavior). This is a quite big amount of data from which conclusions are not immediate to extract.

### 3.5.5. ANalysis Of VAriance (ANOVA)

As we see, the experiments result on a quite big amount of data. 100 runs are done for every single configuration and we are interested on the mean behavior in each of them. However, in order to extract conclusions we should be able to quantify the influence that each parameter has on the results. With this purpose we use ANalysis Of VAriance (ANOVA). A good introduction to ANOVA can be found in [48] but also in websites such as StatSoft's[10].

The goal of ANOVA is to test for significant differences between means by comparing variances. In our case, we study the influence of the different values of all the parameters and the interactions among them on the value of *F-measure* for a given iteration. What we do for that is to perform the test comparing the results for different conditions (e.g. different values of a certain parameter). If the change in these conditions leads to significant differences in the mean, we state that this change is due to that particular change. If the difference in the means is

[10] http://www.statsoft.com/textbook/

not significant, then the differences in particular experiments are considered to be due to chance.

The null hypothesis in the test is that the means of assumed normally distributed populations, all having the same standard deviation, are equal. This means that if we reject the null hypothesis we say that actually the means are different. In our specific case, each distribution corresponds to a certain configuration of values of one (several) parameter(s). If the null hypothesis is rejected, they value(s) of that (those) parameter(s) is considered to influence the results of F-measure.

Table 5 shows an example in which the influence of a variable called WEIGHTS (which can take four values) has on the F-measure in the last iteration. The p-value is the one we are most interested in. In this work we take a significance value of 0.05% (one of the most commonly used), so those factors that have $p<0.05$ are considered to be significant. This value actually tells that, according to the analyzed data, differences on the mean will be due to the studied factor p% of the time, while the rest will be due to chance. Here we would conclude that WEIGHTS has an influence on the F-measure in the last iteration, and we would express it showing the F-ratio as a function of the degrees of freedom $F\ (3, 114196) = 50.715$, $p = 0.000$[11].

[11] This is a standard way of reporting ANOVA results in APA style. http://vault.hanover.edu/~altermattw/methods/stats/

| Source | Type III SS | df | Mean Squares | F-ratio | p-value |
|---|---|---|---|---|---|
| WEIGHTS | 1.594 | 3 | 0.531 | 50.715 | 0.000 |
| Error | 1206.677 | 115196 | 0.010 | | |

Table 5. Results of ANOVA test on the influence of WEIGHTS on the F-measure in the last iteration (ac6).

# 4. Results

This chapter focuses on presenting the results of the different steps of this project. Accordingly, it is divided in three parts corresponding to the listeners' validation, the experiments in WEKA and the exploration of Support Vector Machine active learning techniques. This last part implies studying the effect of all the involved parameters: active learning strategy, initial training size, elements to add per iteration and weights given to the three parameters considered for active learning. Moreover, statistical analysis (ANOVA) is done in order to quantify the significance of the influence of possible combinations of parameters.

## 4.1. Listeners' validation

As explained in the corresponding part of Methodology (Listeners validation 3.3), this validation is done in order to remove songs which are wrongly labeled. We decided to keep those songs which were accepted by at least **60% of the users**, discarding the rest for posterior experiments. Fig. 13 shows the outcome of the listeners' validation. Note that this figure shows how much agreement there is among users. We consider it is interesting to show the results this way because, as we can see, there are songs which are discarded by all users, which may mean

that the selected candidates were actually wrongly chosen, though they represent less than 5% in all categories, being *humorous* and *sentimental* the most rejected ones. Also, showing the results in such a way, we see how much we would decrease the size of the dataset depending on the criterion we establish.

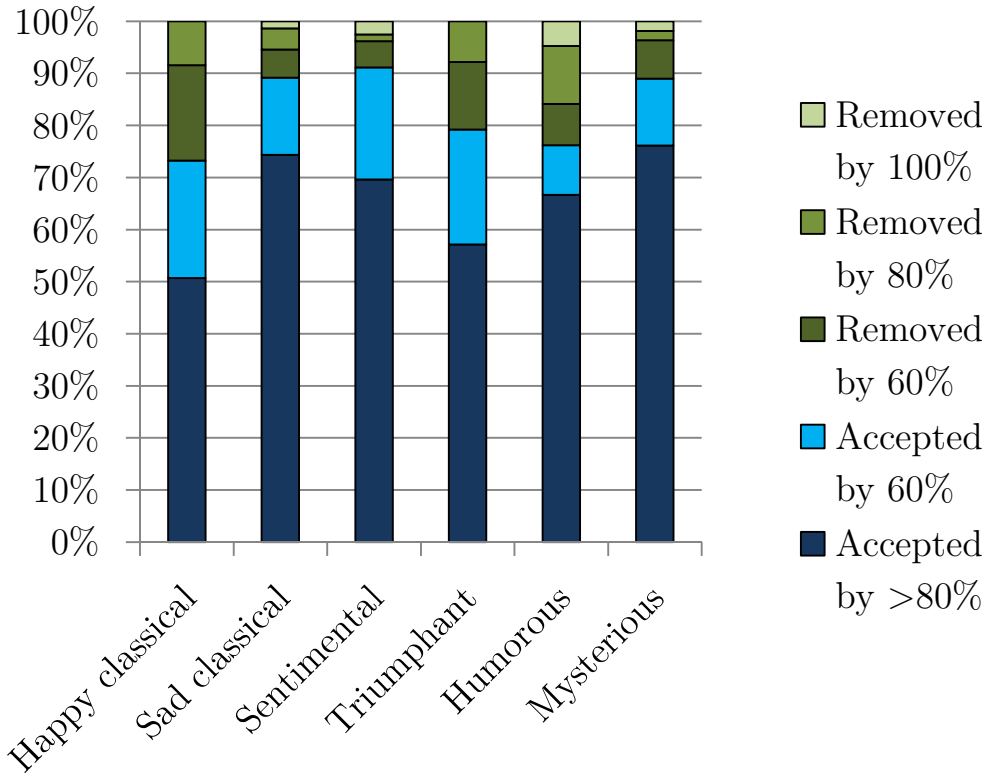



Fig. 13. Results of listeners validation. In blue (light+dark), songs that were kept after validation; in green, songs that were discarded.

## 4.2. Classification experiments

The F-measure of different classifiers is evaluated over the datasets of the mood labels to be added before and after the listeners' validation. Table 6 shows the F-measure of the evaluated classifiers.

| | 0R | 1R | Ibk | Bayes | Forest | SVM-p | SVM-r | SL |
|---|---|---|---|---|---|---|---|---|
| Sentimental | 0.50 | 0.62 | 0.80 | **0.88** | 0.86 | 0.86 | 0.84 | 0.83 |
| *Sentimental* | 0.46 | 0.69 | 0.83 | **0.89** | 0.87 | 0.83 | 0.85 | 0.84 |
| Humorous | 0.49 | 0.79 | 0.87 | 0.89 | **0.96** | 0.90 | 0.90 | 0.85 |
| *Humorous* | 0.47 | 0.85 | 0.89 | 0.89 | **0.93** | 0.90 | 0.90 | 0.87 |
| Mysterious | 0.48 | 0.81 | 0.88 | 0.90 | 0.89 | 0.91 | **0.92** | 0.88 |
| *Mysterious* | 0.49 | 0.75 | 0.79 | 0.89 | 0.89 | **0.94** | 0.93 | 0.89 |
| Triumphant | 0.47 | 0.68 | 0.80 | 0.87 | 0.88 | 0.87 | **0.92** | 0.87 |
| *Triumphant* | 0.47 | 0.64 | 0.83 | 0.91 | 0.91 | **0.93** | 0.92 | 0.87 |

Table 6. F-measure results after 10 runs of 10-fold cross validation for different classifiers over all datasets. Datasets in *italic* are post-listeners' validation. Best result for each dataset appears in **bold**.

The average behavior is again compared in Fig. 14. We observe that there is not a visible difference among methods apart from Nearest Neighbors (Ibk), which is outperformed by the rest of classifiers. It is also interesting to see that 1-Rule (which works with only 1 descriptor or, in our case, with a single PCA component) performs quite well for some datasets. However, it is clearly outperformed by most of more complex classifiers in most of the cases.

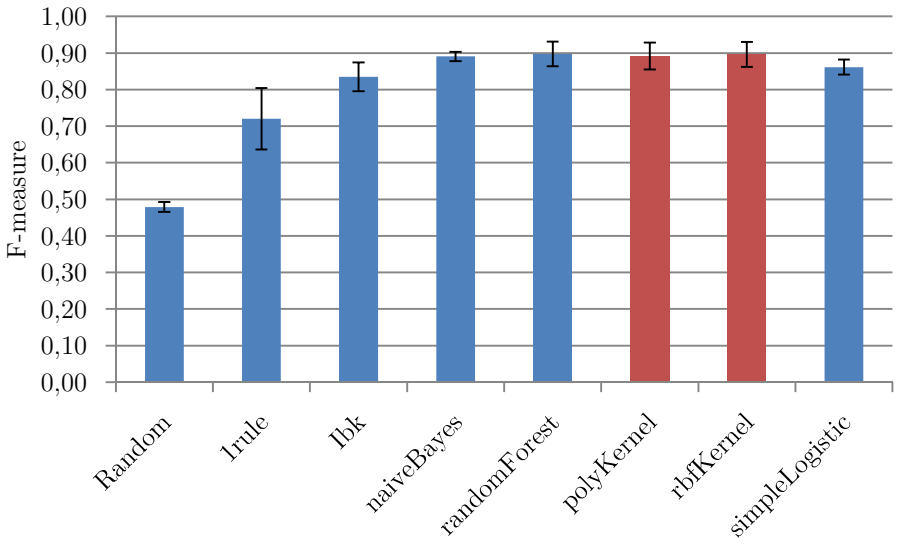



Fig. 14 Comparison of average F-measures over all datasets for different classifiers. SVM techniques are highlighted in red.

These results reinforce the idea that SVM classifiers show good performance for our problem. Accordingly, we consider it is worth exploring SVM active learning techniques to reduce the size of training datasets.

As it can be observed in Fig. 15, the results of these classifications also show that, in average, the F-measure of classifiers is not clearly higher for the datasets after listeners' validation. However, the differences seem to be small in any case, so anyway we keep working with the datasets that result from the listeners' validation, as they are considered to have a more valuable ground truth, agreed by several listeners.

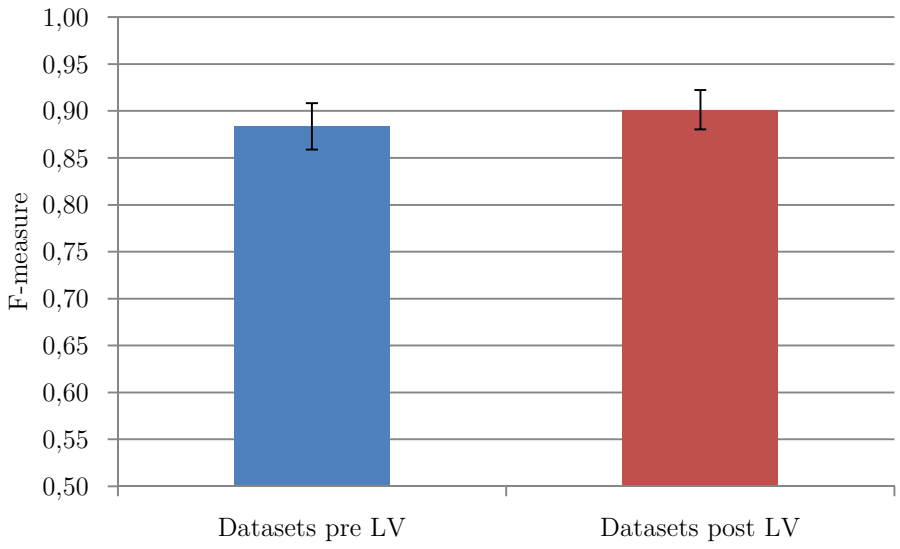



Fig. 15. Average F-measure for all the classifiers over all the datasets (pre and post listeners' validation)

## 4.3. Active learning experiments

In the following Section, the influence of the parameters that play a role in active learning is studied. First, we study how the two different proposed active learning strategies (**METHOD**, Wang's or *modified* Wang's) work in average over all the datasets. Then, we see how precision and recall are affected by the weights of the three considered parameters (distance to boundary (**W1**), distance

diversity (**W2**) and density around the sample (**W3**), with possible combinations [1,1,1], [4,1,1], [1,4,1] and [1,1,4]). At the same time, the way in which precision and recall are affected by the initial training sizes (**ITS**, with values 2, 5 and 10) and elements to add per iteration (**EPI**, with values 2, 6 and 8) is studied.

Apart from the graphs with the direct results of the experiments, statistical analysis (ANOVA) of the influence and interactions of these parameters is presented to discuss if the observed differences between mean behaviors are actually significant.

### 4.3.1. Influence of METHOD

The following results are the average performance over all datasets.

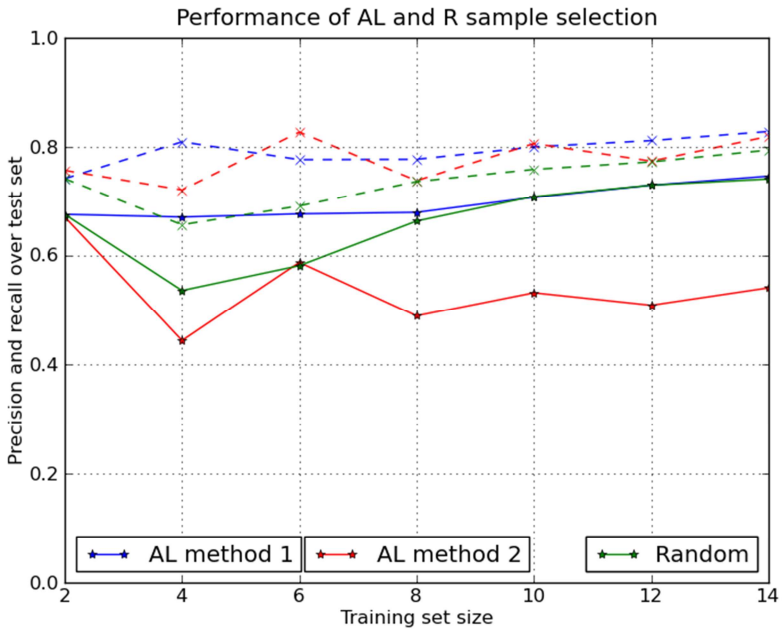



Fig. 16. Precision (-) and recall (--) during 7 rounds with ITS=2, EPI=2, W1=W1=W3=1 for both active learning methods and random selection.

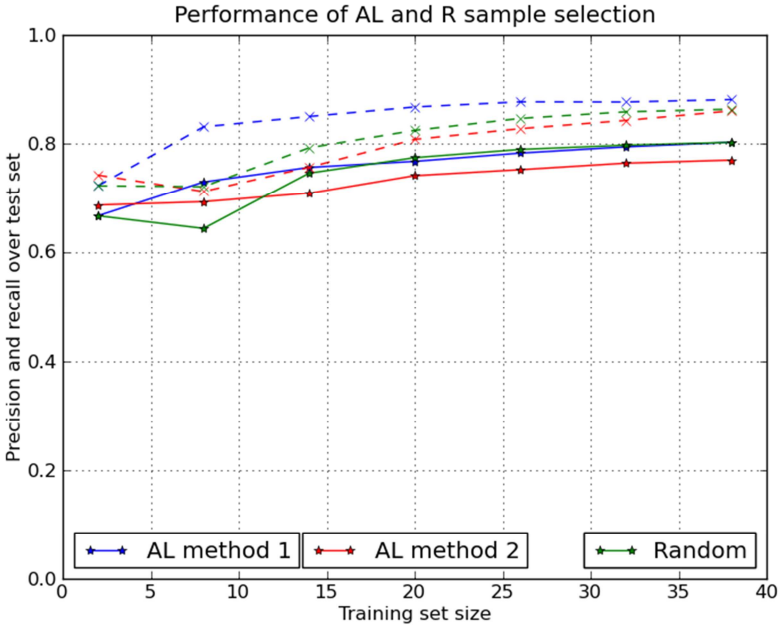



Fig. 17. Precision (-) and recall (--) during 7 rounds with ITS=2, EPI=6, W1=W1=W3=1 for both active learning methods and random selection.

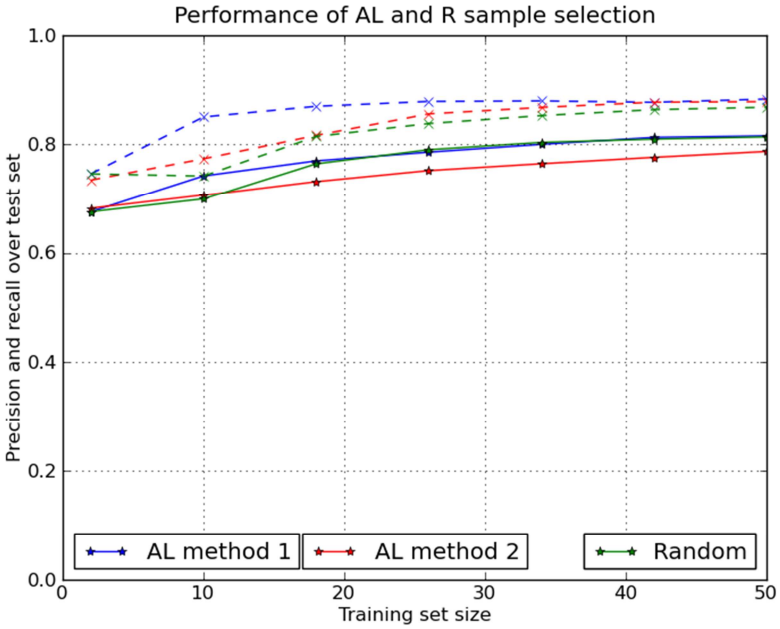



Fig. 18. Precision (-) and recall (--) during 7 rounds with ITS=2, EPI=8, W1=W1=W3=1 for both active learning methods and random selection.

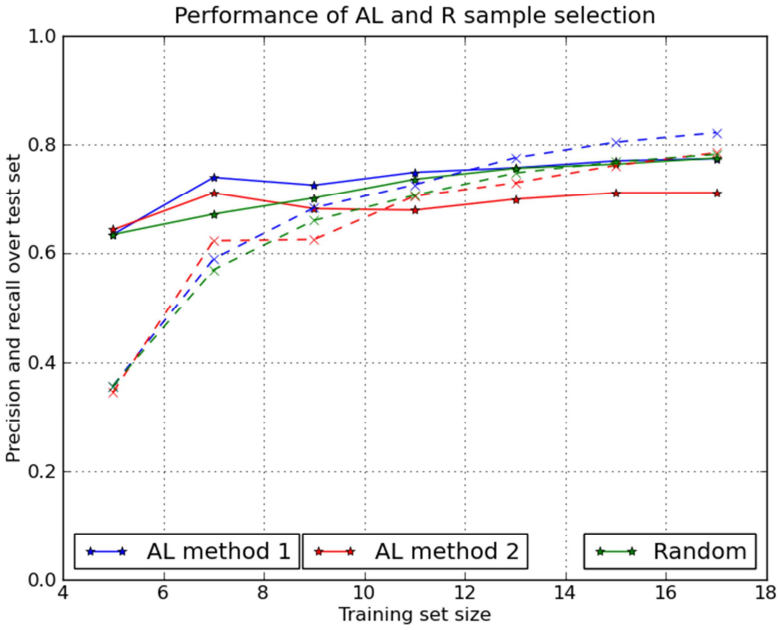



Fig. 19. Precision (-) and recall (--) during 7 rounds with ITS=5, EPI=2, W1=W1=W3=1 for both active learning methods and random selection.

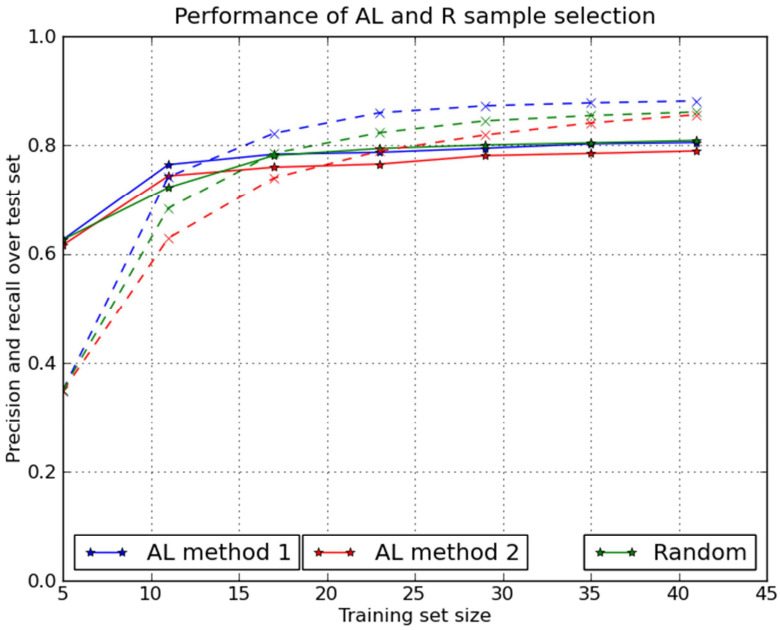



Fig. 20. Precision (-) and recall (--) during 7 rounds with ITS=5, EPI=6, W1=W1=W3=1 for both active learning methods and random selection.

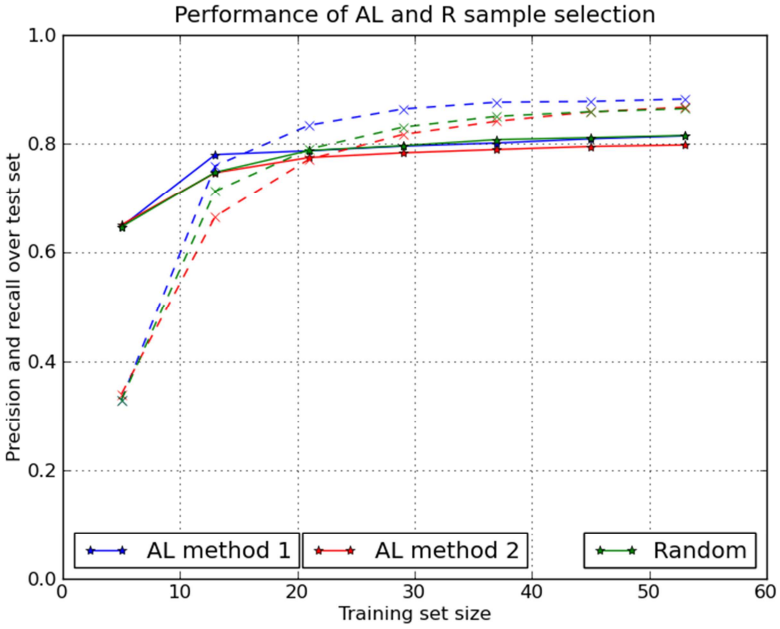



Fig. 21. Precision (-) and recall (--) during 7 rounds with ITS=5, EPI=8, W1=W1=W3=1 for both active learning methods and random selection.

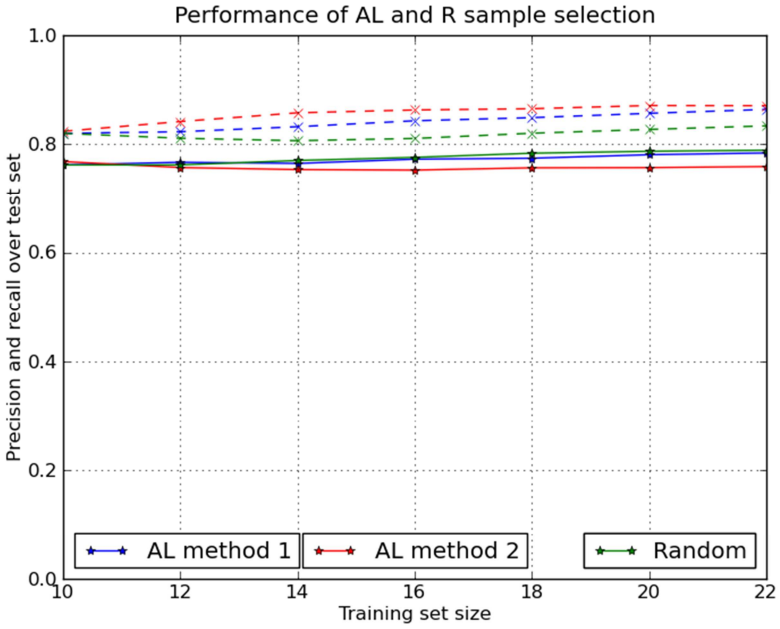



Fig. 22. Precision (-) and recall (--) during 7 rounds with ITS=10, EPI=2, W1=W1=W3=1 for both active learning methods and random selection.

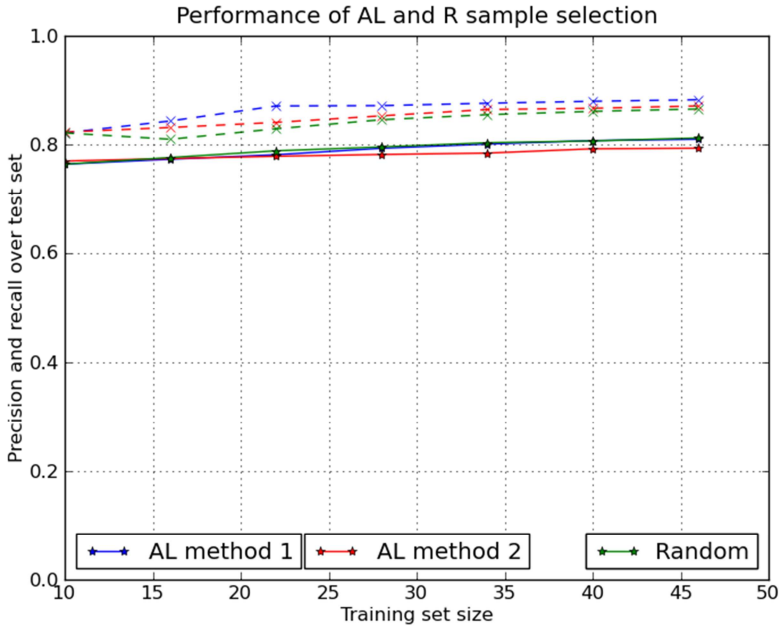



Fig. 23. Precision (-) and recall (--) during 7 rounds with ITS=10, EPI=6, W1=W1=W3=1 for both active learning methods and random selection.

According to the results, the second active learning method (consisting on taking half of the elements following the same algorithm as in first method and the other half those furthest from the boundary) has even worse performance than random sample selection. A reason for this may be that the half of the elements that are chosen for being furthest from the boundary could actually be outliers. Reducing the number of elements that are taken according to this criterion could help.

For the first method, we do not see that in general it is better than random selection. There are, however, some scenarios in which it is for certain iterations. From Fig. 16 to Fig. 18 (with ITS = 2), it is shown that active learning can improve the performance of the classifier in the first rounds for very small initial training sets.

Observe also that, while in Fig. 16 both method 1 and random selection have exactly the same performance at $5^{th}$ iteration (training size = 10), in Fig. 18 there

is a difference of around 5% for the same training size (2nd iteration). This suggests that the number of elements which are added per iteration plays a role on the evolution of the system.

ANOVA tests have been done to study the influence of ITS, EPI and METHOD for the **F-measure** (correctly classified instances) in first and last iterations.

*First iteration*

For the first iteration, the results (presented in Table 7) are quite obvious. The difference in the F-measure differed significantly for the 3 values of ITS (2, 5, 10), $F(2, 115363) = 23831.965$, $p = 0.000$. However, EPI ($F(2, 115363) = 1.522$, $p = 0.218$) and METHOD ($F(1, 115363) = 0.034$, $p = 0.967$) do not have any effect on the F-measure for the first iteration at $p < 0.05$. Also, the interactions among them are not significant either (observe rows 5 to 8 in Table 7).

Although these results are the ones we could expect, it is worth checking that the results of the ANOVA tests under a controlled situation make sense: the only factor that influences the results on the first iteration is the size of the initial training set.

| Source | Type III SS | df | Mean Squares | F-ratio | p-value |
|---|---|---|---|---|---|
| ITS | 2705.505 | 2 | 1352.753 | 23831.965 | 0.000 |
| EPI | 0.173 | 2 | 0.086 | 1.522 | 0.218 |
| METHOD | 0.004 | 2 | 0.002 | 0.034 | 0.967 |
| ITS*EPI | 0.306 | 4 | 0.077 | 1.349 | 0.249 |
| ITS*METHOD | 0.019 | 4 | 0.005 | 0.086 | 0.987 |
| EPI*METHOD | 0.078 | 4 | 0.020 | 0.344 | 0.848 |
| ITS*EPI*METHOD | 0.326 | 8 | 0.041 | 0.718 | 0.676 |
| Error | 6548.247 | 115363 | 0.057 | | |

Table 7. Results of ANOVA test on the influence and interactions among ITS, EPI and METHOD on the F-measure in the 1st iteration.

*Last iteration*

The same test is done studying the F-measure in last iteration this time. The results of the test are completely different and shows that all the parameters (ITS, EPI and METHOD) have a significant influence on the F-measure in the last iteration.

Observe, for example, that while in the previous case there was no influence on any case of interaction among the parameters, now there is even influence on the interaction of all of them, $F(8, 115363) = 55.251$, $p = 0.000$. All the results are shown in Table 8.

| Source | Type III SS | df | Mean Squares | F-ratio | p-value |
|---|---|---|---|---|---|
| ITS | 38.307 | 2 | 19.154 | 1261.109 | 0.000 |
| EPI | 147.580 | 2 | 73.790 | 4858.475 | 0.000 |
| METHOD | 33.143 | 2 | 16.572 | 1091.116 | 0.000 |
| ITS*EPI | 35.687 | 4 | 8.922 | 587.433 | 0.000 |
| ITS*METHOD | 11.219 | 4 | 2.805 | 184.672 | 0.000 |
| EPI*METHOD | 7.145 | 4 | 1.786 | 117.606 | 0.000 |
| ITS*EPI*METHOD | 6.713 | 8 | 0.839 | 55.251 | 0.000 |
| Error | 1752.117 | 115363 | 0.015 | | |

Table 8. Results of ANOVA test on the influence and interactions among ITS, EPI and METHOD on the F-measure in the last iteration.

About METHOD, which is the parameter we are studying now, the results of the ANOVA test confirm what was concluded from the graphics: there is an influence of METHOD on the results, F (2, 115363 = 55.251), p = 000.

### 4.3.2. Influence of WEIGHTS

*Note: results for the 1st iteration in the following graphs are not exactly the same because the experiments were done separately. However, they are averaged after 100 runs.*

As here we try to see what is the influence that the four proposed combination of weights ([1,1,1], [4,1,1], [1,4,1] and [1,1,4]) the results for METHOD 1 are presented. This is because in METHOD 2 the WEIGHTS are just playing a role for half of the elements that are selected.

#### 4.3.2.1. **Influence of WEIGHTS for different ITS**

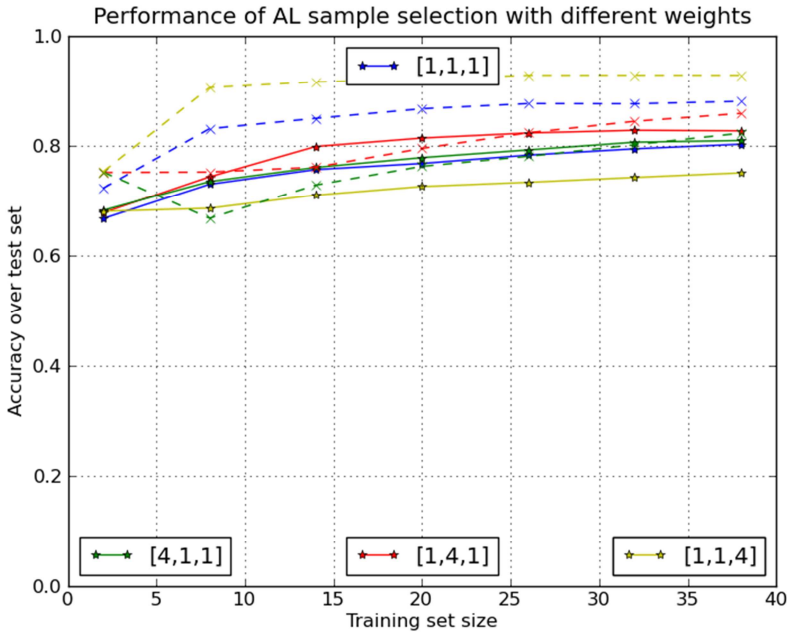



Fig. 24. Precision (-) and recall (--) during 7 rounds with ITS=2, EPI=6, METHOD=1 for the four combinations of WEIGHTS.

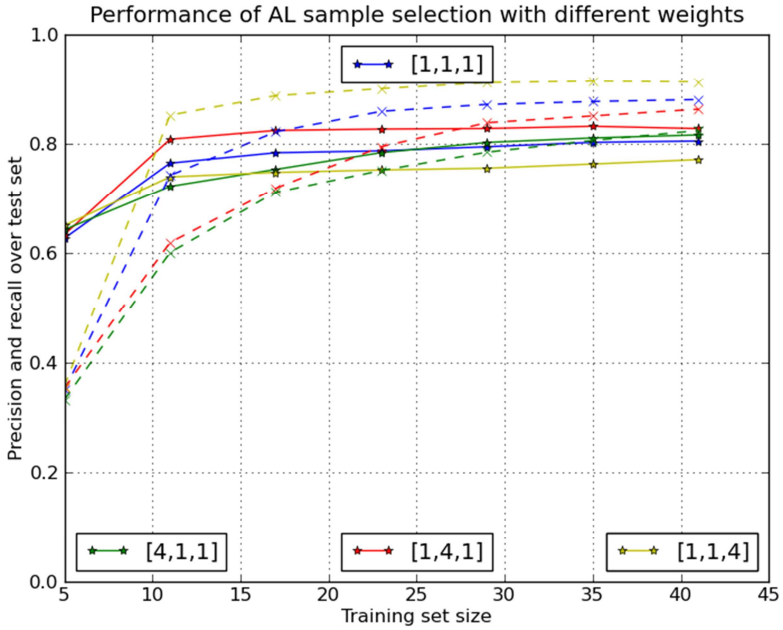



Fig. 25. Precision (-) and recall (--) during 7 rounds with ITS=5, EPI=6, METHOD=1 for the four combinations of WEIGHTS.

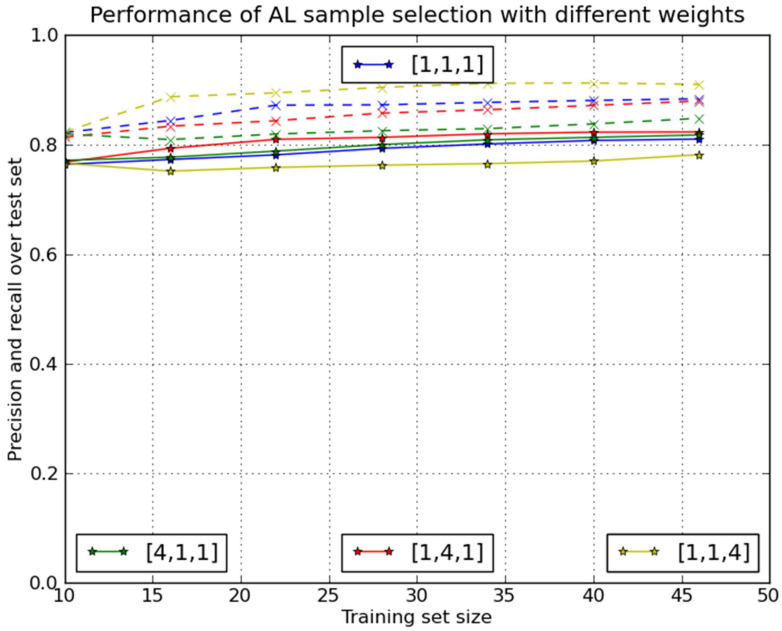



Fig. 26. Precision (-) and recall (--) during 7 rounds with ITS=10, EPI=6, METHOD=1 for the four combinations of WEIGHTS.

These results lead to some conclusions within this scenario:

- Combination of weights [1,1,1] (three parameters with same weight) is not clearly outperformed by any other combination. This is in agreement with the results in [25,27] and the idea that it is actually the fact of taking into account all the factors what makes the algorithm more accurate.
- Combinations of weights [4,1,1] (more weight to distance to the boundary) and [1,1,4] (more weight to density around the sample) are outperformed by the two other combinations. In the first case, it can be due to the problem that we tried to solve with METHOD 2 about uncertainty-based methods. This is, those samples closest to the boundary can actually be neutral and can lead to "more certainty about the wrong thing" [33].In the second case, in which samples from denser regions are taken, it can happen that we keep learning about the same region all the time, thus not being able to make the system understand the structure in the rest of the space.

- Combination of weights [1,4,1] (more weight to distance diversity) works as well as [1,1,1]. This may indicate that taking samples that are as non-redundant as possible is a good strategy for our framework.

#### 4.3.2.2. **Influence of WEIGHTS for different EPI**

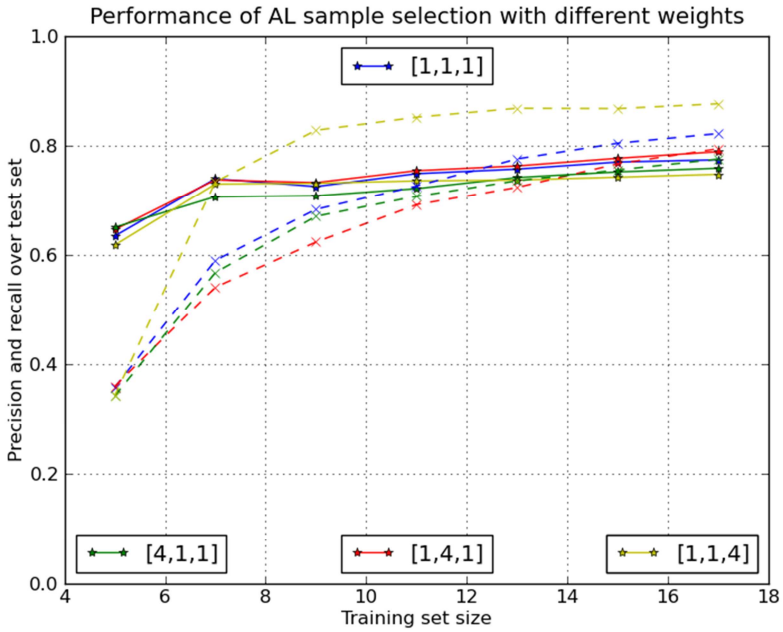



Fig. 27. Precision (-) and recall (--) during 7 rounds with ITS=5, EPI=2, METHOD=1 for the four combinations of WEIGHTS.

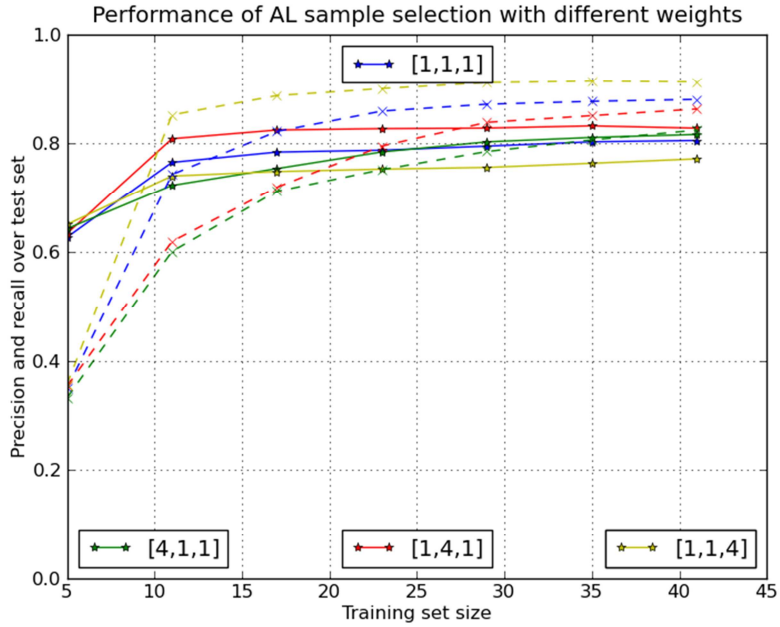



Fig. 28. Precision (-) and recall (--) during 7 rounds with ITS=5, EPI=6, METHOD=1 for the four combinations of WEIGHTS.

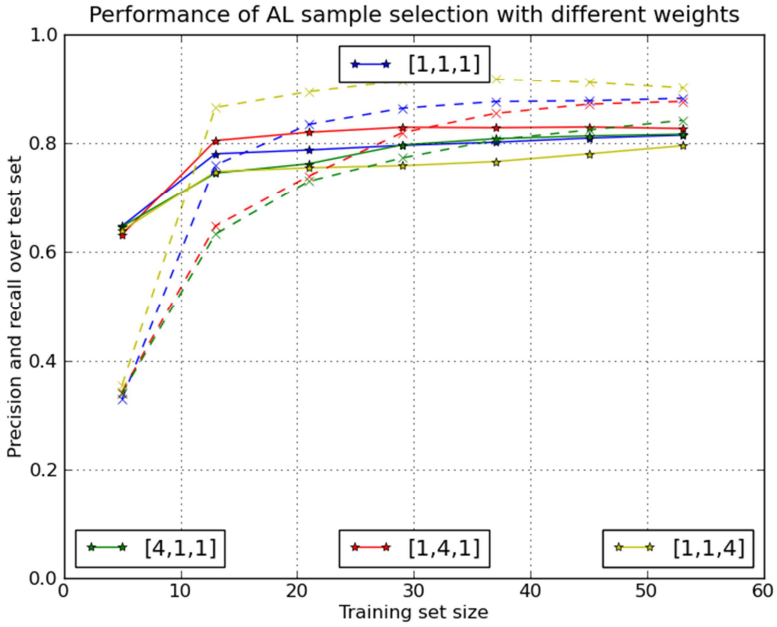



Fig. 29. Precision (-) and recall (--) during 7 rounds with ITS=5, EPI=8, METHOD=1 for the four combinations of WEIGHTS.

The conclusions here are not much different from the ones commented in the case of different ITS except for one thing: Fig. 27 (with EPI=2) shows a good

performance of combination [1,1,4], which is generally outperformed by the rest of combinations for other cases. It is not easy to draw a conclusion from this fact, but it seems that giving importance to density may be good in cases in which the multi-sample selection strategy is looking for very few samples.

### 4.3.3. ANOVA

In the case of this statistical analysis, we use a variable WEIGHTS which can take 4 values that code the 4 possible combinations of weights. The results (Table 9) show that the combination of weights has a significant influence on the F-measure for the last iteration, $F(3, 115354) = 53.056$, $p = 0.000$.

| Source | Type III SS | df | Mean Squares | F-ratio | p-value |
|---|---|---|---|---|---|
| WEIGHTS | 2.484 | 3 | 0.828 | 53.056 | 0.000 |
| ITS | 33.247 | 2 | 16.624 | 1065.060 | 0.000 |
| EPI | 151.742 | 2 | 75.871 | 4860.999 | 0.000 |
| WEIGHTS * ITS | 2.052 | 6 | 0.342 | 21.913 | 0.000 |
| WEIGHTS * EPI | 3.281 | 6 | 0.547 | 35.033 | 0.000 |
| ITS * EPI | 32.534 | 4 | 8.134 | 521.110 | 0.000 |
| WEIGHTS * ITS * EPI | 1.913 | 12 | 0.159 | 10.215 | 0.000 |
| Error | 1800.461 | 115354 | 0.016 | | |

Table 9. Results of ANOVA test on the influence and interactions among ITS, EPI and WEIGHTS on the F-measure in the last iteration.

The same analysis is done for the results in the 3rd iteration, as a way to check if there is an influence already at that stage. The results (Table 10) of the

analysis show that the combination of weights also has significant influence on the F-measure at the 3rd iteration, F (2, 115354) = 203.867.

| Source | Type III SS | df | Mean Squares | F-ratio | p-value |
|---|---|---|---|---|---|
| WEIGHTS | 23.388 | 3 | 7.796 | 203.867 | 0.000 |
| ITS | 207.052 | 2 | 103.526 | 2707.192 | 0.000 |
| EPI | 210.897 | 2 | 105.449 | 2757.472 | 0.000 |
| WEIGHTS * ITS | 8.497 | 6 | 1.416 | 37.031 | 0.000 |
| WEIGHTS * EPI | 0.380 | 6 | 0.063 | 1.657 | 0.127 |
| ITS * EPI | 58.173 | 4 | 14.543 | 380.306 | 0.000 |
| WEIGHTS * ITS * EPI | 0.952 | 12 | 0.079 | 2.074 | 0.015 |
| Error | 4411.256 | 115354 | 0.038 | | |

Table 10. Results of ANOVA test on the influence and interactions among ITS, EPI and WEIGHTS on the F-measure in the 3rd iteration.

## 4.4. Summary

The main results that have been presented in this Chapter are the following:

- The collections after listeners' validation do not decrease performance and thus they can be used for posterior analysis given that they are considered to be a more valid groun truth.
- SVM classifiers are not outperformed by any other of the evaluated classifiers. In this sense, posterior active learning experiments are done using SVM techniques.

- The first proposed active learning technique outperforms random selection in certain cases. The second one is outperformed by the two others.
- With the first proposed active learning technique, top results are achieved without using the whole dataset (see Figs. 20 to 23).
- The influence of all the parameters is statistically checked via ANOVA. The results show that the METHOD and the WEIGHTS (for the first method) have an influence in the results. Also, the ITS and EPI parameters influence the performance.

# 5. Expanding old models to classical music

One of the problems that has been detected in previous works is that models that are trained with non-classical music have problems to extend to this genre [2]. Some of the features that are extracted from the audio, especially those related to tempo and rhythm can give meaningless results when applied to music with smooth and subtle rhythms.

In this context, as an extra task during our work, we performed some experiments in which we try to see this effect in different ways. With that purpose, we focused on the *happy* category as a case of study. It was selected because it is a specific case in which it had been shown that generalization to classical music did not get good results [2]. A new song collection of happy classical music has been created and validated (following the same methodology explained for the rest of collections) and we also used the one created by Cyril Laurier for his works [2,20,26,49,50].

The experiments include 10 runs of 10-fold cross validation of both datasets separately and two experiments having one as the training set and the other as test set.

## 5.1. Results and discussion

Table 11 and Table 12 show the results of 10 runs of 10-fold cross validation over the two datasets separately. As we observe in Table 13 and Table 14, the accuracy is dramatically decreased when trying to extend one of the two models to the other collections. Both cases (classifying the classical music collection with the model trained with the old one and vice versa) show very low accuracies.

In Table 15, both collections are joined forming a single one. The results are quite similar to those of Table 11 (where just the old collection is used). This may mean that, when mixing both collections, genre-dependent features lose importance. This fact reinforces the idea mentioned in 3.2 (Creating the audio datasets): it is very important to create collections in which many genres are represented in order to make the model just depend on related-to-mood features.

However, we also try in this work to deal with reducing the size of training sets. In this sense, it seems there is a limit on reducing this size if all genres must be represented for the model to be accurate. Nevertheless, we observe in Table 12 that the mood classification within classical music has a good performance.

Summarizing, the results here may suggest that reducing the size of training sets is a task that may require having genre-dependent mood classifiers in certain contexts. Genre classification is a topic of big interest which results keep improving and which are particularly good in general for the specific case of classical music [25,27,51][51][49][49]. Nevertheless, it has been also shown than mixing both categories does not result in worse performance for this particular case, so just being able to build collections that really represent the complete variety of songs that can have a particular mood seems to be a good approach.

| | 0-R | 1-R | Bayes | Ibk | Forest | SVM-p | SVM-r | SL |
|---|---|---|---|---|---|---|---|---|
| Precision | 0.25 | 0.631 | 0.678 | 0.732 | 0.759 | 0.741 | 0.76 | 0.778 |
| Recall | 0.5 | 0.646 | 0.815 | 0.701 | 0.755 | 0.741 | 0.75 | 0.777 |
| F-measure | 0.333 | 0.638 | 0.740 | 0.716 | 0.757 | 0.741 | 0.755 | 0.777 |

Table 11. Results of 10 runs of 10-fold cross validation over Cyril Laurier's *happy* songs collection.

| | 0-R | 1-R | Bayes | Ibk | Forest | SVM-p | SVM-r | SL |
|---|---|---|---|---|---|---|---|---|
| Precision | 0.25 | 0.824 | 0.863 | 0.843 | 0.854 | 0.824 | 0.844 | 0.883 |
| Recall | 0.5 | 0.824 | 0.863 | 0.843 | 0.853 | 0.824 | 0.843 | 0.882 |
| F-measure | 0.333 | 0.824 | 0.863 | 0.843 | 0.853 | 0.824 | 0.843 | 0.882 |

Table 12. Results of 10 runs of 10-fold cross validation over classical music *happy* songs collection.

| | 0-R | 1-R | Bayes | Ibk | Forest | SVM-p | SVM-r | SL |
|---|---|---|---|---|---|---|---|---|
| Precision | 0.25 | 0.786 | 0.758 | 0.747 | 0.725 | 0.725 | 0.754 | 0.611 |
| Recall | 0.5 | 0.625 | 0.696 | 0.735 | 0.647 | 0.725 | 0.627 | 0.559 |
| F-measure | 0.333 | 0.696 | 0.726 | 0.741 | 0.684 | 0.725 | 0.685 | 0.584 |

Table 13. Results of classifier trained with Cyril Laurier's song collection and evaluated over classical music *happy* songs collection.

| | 0-R | 1-R | Bayes | Ibk | Forest | SVM-p | SVM-r | SL |
|---|---|---|---|---|---|---|---|---|
| Precision | 0.25 | 0.786 | 0.72 | 0.768 | 0.725 | 0.773 | 0.769 | 0.764 |
| Recall | 0.5 | 0.625 | 0.545 | 0.567 | 0.647 | 0.585 | 0.571 | 0.554 |
| F-measure | 0.333 | 0.696 | 0.620 | 0.652 | 0.684 | 0.666 | 0.655 | 0.642 |

Table 14. Results of classifier trained with classical music *happy* songs collection and evaluated over Cyril Laurier's song collection.

| | 0-R | 1-R | Bayes | Ibk | Forest | SVM-p | SVM-r | SL |
|---|---|---|---|---|---|---|---|---|
| Precision | 0.25 | 0.695 | 0.752 | 0.75 | 0.759 | 0.78 | 0.758 | 0.783 |
| Recall | 0.5 | 0.725 | 0.715 | 0.73 | 0.755 | 0.755 | 0.755 | 0.782 |
| F-measure | 0.333 | 0.710 | 0.733 | 0.740 | 0.757 | 0.767 | 0.756 | 0.782 |

Table 15. Results of 10 runs of 10-fold cross validation over Cyril Laurier's and classical music *happy* song collections joined together.

Fig. 30 shows the results for F-measure on the previous configurations for the concrete case of SVM RBF classifier. Observe how joining both collections results on a very similar performance to the one achieved using only the old collection

and how training the system with one collection and evaluating it over the other one results on worse performance.

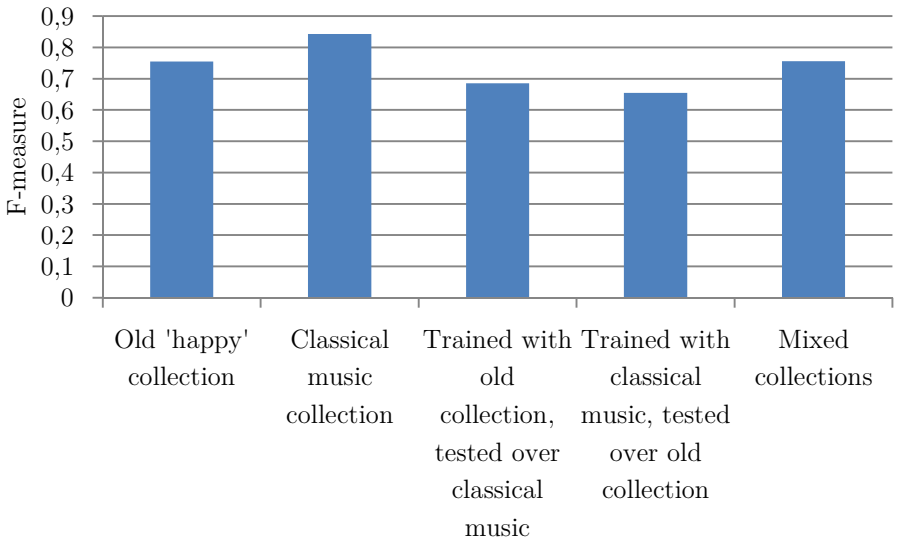



Fig. 30. Comparison of F-measure results for different settings for SVM RBF classifier.

# 6. Conclusions

This work had two main goals: expanding the current mood tags and exploring the use of active learning techniques as a tool for allowing the automatic mood detection systems being trained with fewer songs.

New songs collections have been created and are now available for future works in the Music Technology Group for 4 new mood categories: *triumphant, sentimental, mysterious* and *humorous*. These datasets have been created using online annotations of songs and validated with single users. The results of classification experiments with these new datasets (around 0.9 average F-measure value in 10-fold cross validation with Support Vector Machine classifiers) suggest they could be used to create new models to be added to Essentia. In addition, new datasets for *happy* and *sad* classical music have also been created to study how the current models can expand to that kind of music.

A comprehensive state-of-the-art review of active learning has been done in order to identify the techniques that could better suit our problem. In that sense, the approach by Wang *et al.* was considered as a good starting point as it had shown good results in genre classification problems. The results of the experiments in this work do not show a significant improvement, but some

analysis and conclusions could be extracted. By studying the dependency of the results on the initial training sizes and elements to add per iteration, the rest of the parameters can be fine-tuned depending on the scenario in which the problem appears. Depending on the cost of creating the first training set and getting labels from the user, the approach should be selected consequently. For example, it has been shown that for cases in which the initial training set is very small it is worth using active learning to get a faster improvement of the performance adding a small number of songs at each iteration.

Experiments shown in Chapter 5 suggest that is necessary to have as many genres as possible included in the song collection with which the model is created in order to make the model general. In this work we also try to be able to minimize the songs that are needed to achieve results with a certain error given that getting labels from users may be a difficult task. A solution for certain contexts where the model is not required to be completely general may be creating models which are genre-dependent, something that seems feasible according to how genre classification techniques evolve.

As an unexpected outcome of this work, a problem in Gaia was identified and explained: the probability estimates done by SVM classifiers (that use svmlib) are done through a cross-validation method which makes the results non-deterministic. When deterministic results are needed, probability estimates should be disabled.

## 6.1. Future work

Apart from the work that is presented in this thesis, some tasks are still to be done. Some of the points that could be addressed as extensions to the work presented here are:

- **Add the new mood labels to existing systems** (e.g. Essentia). With the created mood datasets and after checking with experiments that they may be useful for training classifiers, they could be used to create new mood categories in software such as Essentia.
- **Keep improving the mood detection systems adding more labels**. As stated during this work, the complexity of emotions in music is difficult to address with few tags, so the more there are, the more complete that representation is. This problem, however, need to be addressed trying to choose those labels in such a way that they actually improve the emotional explanation.
- **Perform active learning experiments with users**. The experiments in this project emulate the scenario in which a user is asked to label songs during some iterations. In order to fully understand how the system would work, these experiments should be repeated having actual users who give feedback.
- **Explore more active learning techniques**. The strategy presented here, which mixes uncertainty-based and density-weighted methods is not the only possibility. In the state-of-the-art chapter of this thesis, many active learning strategies are presented and it may be worth trying to extend them to our problem. For example, some of the techniques that have been explained in this work such as "Expected Error Reduction" or "Expected Variance Reduction" guarantee a certain improvement over random selection (as they calculate the improvement that all the samples cause *a priori*). The problem of these techniques is that they have a much higher computational cost, but it would be reasonable to implement such techniques in problems in which the size of the datasets is not extremely large.

# Bibliography

[1] Laurier C. Automatic Classification of Musical Mood by Content Based Analysis. Dept. of Technology, Universitat Pompeu Fabra, Barcelona 2011.

[2] Laurier C, Herrera P. Automatic detection of emotion in music: Interaction with emotionally sensitive machines. Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence 2009: pages 9–32.

[3] Fehr B, Russell JA. Concept of emotion viewed from a prototype perspective. J Exp Psychol: Gen 1984; 113: pages 464-486.

[4] Dictionary OE. Oxford: Oxford University Press. Retrieved May 2004; 24: 2004.

[5] Rolls ET. Emotion explained: Oxford University Press, USA; 2005.

[6] Damasio AR. Descartes' error: Emotion, reason, and the human brain: Quill New York: 2000.

[7] Kivy P. Sound sentiment: an essay on the musical emotions, including the complete text of The Corded shell: Temple Univ Pr; 1989.

[8] Balkwill LL, Thompson WF. A cross-cultural investigation of the perception of emotion in music: Psychophysical and cultural cues. Music Perception 1999: 43-64.

[9] Grewe O, Nagel F, Kopiez R, Altenmüller E. Emotions over time: Synchronicity and development of subjective, physiological, and facial affective reactions to music. Emotion 2007; 7: pages 774-788.

[10] Koelsch S, Fritz T. Investigating emotion with music: an fMRI study. Hum Brain Mapp 2006; 27: pages239-250.

[11] Blood AJ, Zatorre RJ. Intensely pleasurable responses to music correlate with activity in brain regions implicated in reward and emotion. Proc Natl Acad Sci U S A 2001; 98: 11818.

[12] Huron DB. Sweet anticipation: Music and the psychology of expectation: The MIT Press; 2006.

[13] Juslin PN, Laukka P. Expression, perception, and induction of musical emotions: A review and a questionnaire study of everyday listening. Journal of New Music Research 2004; 33: pages 217-238.

[14] Hevner K. Experimental studies of the elements of expression in music. Am J Psychol 1936; 48: pages 246-268.

[15] Russell JA. A circumplex model of affect. J Pers Soc Psychol 1980; 39: pages 1161-1178.

[16] Gouyon F, Klapuri A, Dixon S, Alonso M, Tzanetakis G, Uhle C, et al. An experimental comparison of audio tempo induction algorithms. Audio, Speech, and Language Processing, IEEE Transactions on 2006; 14: pages 1832-1844.

[17] Gómez E. Tonal description of music audio signals. Doctoral dissertation, Dept. of Technology, Universitat Pompeu Fabra, Barcelona 2006.

[18] Bigand E, Vieillard S, Madurell F, Marozeau J, Dacquet A. . Cognition & Emotion 2005; 19: pages 1113-1139.

[19] Lu L, Liu D, Zhang HJ. Automatic mood detection and tracking of music audio signals. Audio, Speech, and Language Processing, IEEE Transactions on 2005; 14: pages 5-18.

[20] Laurier C, Meyers O, Serrà J, Blech M, Herrera P, Serra X. Indexing music by mood: Design and integration of an automatic content-based annotator. Multimedia Tools Appl 2010; 48: pages 161-184.

[21] Y.-Y. Shi, X. Zhu, H.-G. Kim, and K.-W. Eom. A tempo feature via modulation spectrum analysis and its application to music emotion classification. In Proceedings of the IEEE International Conference on Multimedia and Expo, pages 1085–1088, Toronto, Canada, 2006.

[22] Sordo M, Laurier C, Celma O. Annotating music collections: How content-based similarity helps to propagate labels. In ISMIR, 2007.

[23] Wieczorkowska A, Synak P, Lewis R, W. Raś Z. Extracting emotions from music data. Foundations of Intelligent Systems 2005: pages 456-465.

[24] Li T, Ogihara M. Detecting emotion in music. Proceedings of ISMIR, Baltimore, MD, USA, pages 239–240, 2003.

[25] Mandel MI, Poliner GE, Ellis DPW. Support vector machine active learning for music retrieval. Multimedia systems 2006; 12: pages 3-13.

[26] C. Laurier, O. Meyers, J. Serrà, M. Blech, P. Herrera: Music Mood Annotator Design and Integration, 7th International Workshop on Content-Based Multimedia Indexing, Chania, Crete, 2009.

[27] Wang TJ, Chen G, Herrera P. Music retrieval based on a multi-samples selection strategy for support vector machine active learning. In: SAC'09: Proceedings of the 2009 ACM symposium on Applied Computing, ACM, 2009, pages 1750-1751.

[28] Smith LI. A tutorial on principal components analysis. Cornell University, USA 2002

[29] Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. In Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, 1992. ACM.

[30] Burges CJC. A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery 1998; 2: pages 121-167.

[31] Lewis DD, Gale WA. A sequential algorithm for training text classifiers: corrigendum and additional data. SIGIR Forum 29, 2, pages 13–19.

[32] Settles B. Active Learning Literature Survey. Mach Learning 2010; 15: pages 201-221.

[33] Rubens N, Kaplan D, Sugiyama M. Active learning in recommender systems. Recommender Systems Handbook 2011: pages 735-767.

[34] Angluin D. Queries and concept learning. Mach Learning 1988; 2: pages 319-342.

[35] Cohn D, Atlas L, Ladner R. Training connectionist networks with queries and selective sampling: Dep. of Computer Science and Engineering, Univ. of Washington; 1990.

[36] Fujii A, Tokunaga T, Inui K, Tanaka H. Selective sampling for example-based word sense disambiguation. Computational Linguistics 1998; 24: pages 573-597.

[37] Tong S, Koller D. Support vector machine active learning with applications to text classification. The Journal of Machine Learning Research 2002; 2: pages 45-66.

[38] Chen G, Wang T, Herrera P. Relevance Feedback in an Adaptive Space with One-Class SVM for Content-Based Music Retrieval. Proceedings of the International Conference on Audio, Language and Image Processing, 2008. ICALIP.

[39] Chen G, Wang TJ, Herrera P. A Novel Music Retrieval System with Relevance Feedback. 2008 3rd International Conference on Innovative Computing Information and Control, 2008

[40] Seung H, Opper M, Sompolinsky H. Query by committee. Proceedings of the Fifth Annual Workshop on Computational Learning Theory. Pages 287-294.

[41] Settles B, Craven M, Ray S. Multiple-instance active learning. In Advances in Neural Information Processing Systems (NIPS), volume 20, pages 1289–1296. MIT Press, 2008.

[42] Roy N, McCallum A. Toward optimal active learning through sampling estimation of error reduction. In Proceedings of the International Conference on Machine Learning (ICML), pages 441–448. Morgan Kaufmann, 2001.

[43] Zhu X. Semi-supervised learning literature survey. Computer Science, University of Wisconsin-Madison 2006.

[44] Geman S, Bienenstock E, Doursat R. Neural networks and the bias/variance dilemma. Neural Comput 1992; 4: 1-58.

[45] Chang EY, Tong S, Goh K, Chang C. Support vector machine concept-dependent active learning for image retrieval. IEEE Transactions on Multimedia 2005; 2.

[46] Duda RO, Hart PE, Stork DG. Pattern classification: wiley New York; 2001.

[47] Witten IH, Frank E. Practical machine learning tools and techniques. Morgan Kauffman 2005.

[48] Stephens LJ. Advanced statistics demystified: McGraw-Hill Professional; 2004.

[49] Laurier C, Herrera P. Mood Cloud: A Real-Time Music Mood Visualization Tool. Proceedings of the 2008 Computers in Music Modeling and Retrieval Conference, Copenhagen, Danemark, 2008, pages 163–167, 2008.

[50] Laurier C, Lartillot O, Eerola T, Toiviainen P. Exploring relationships between audio features and emotion in music. In 7th Triennial Conference of European Society for theCognitive Sciences of Music (ESCOM 2009).

[51] Guaus i Termens E. Audio content processing for automatic music genre classification: descriptors, databases, and classifiers. Dept. of Technology, Universitat Pompeu Fabra, Barcelona 2009.

# 7. Appendix: GAIA and active learning

This appendix is done in order to clarify some problems that appeared during the development of this work and which are considered to be interesting for future users of GAIA that are interested on using the bindings with libsvm, specifically for probability estimates. An alternative library (in this case, scikits-learn) can be used as a tool for considering distances to the boundary in Support Vector Machine problems keeping the results deterministic.

## 7.1. Identified problem

As seen and explained in the rest of this document, it is important to be able to compare different techniques (active learning, random selection) for multi-sample selection. In order to be able to compare them, the classifiers should, at each step, be completely deterministic. This means that given a training set with which the classifier is trained; the accuracy over a given test set should always be exactly the same.

However, in the first experiments this was not accomplished. As shown in Fig. 31, the results for the active learning method and the random sample selection method are radically different, when actually they should be exactly the same,

given that the train set is the same for both methods in the first iteration and the test set is shared by both in all iterations.

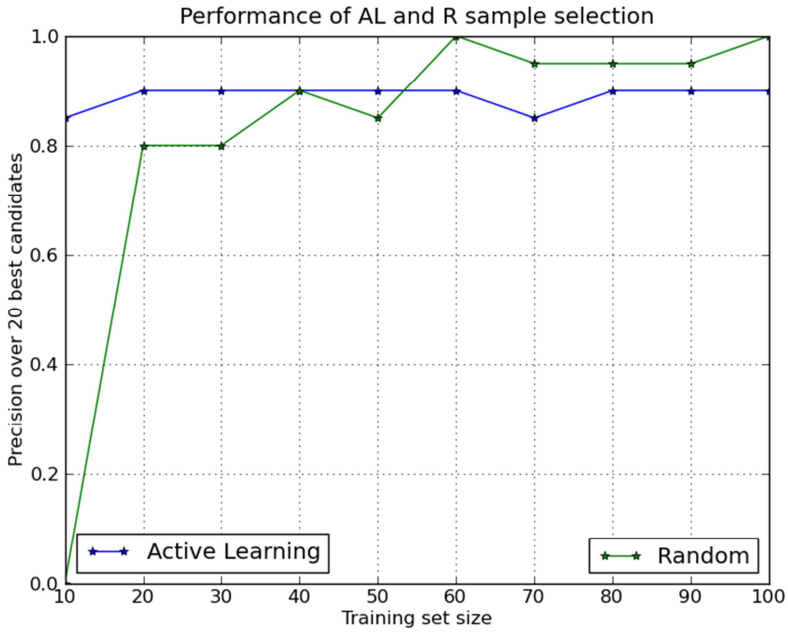



Fig. 31. Accuracy over 20 best candidates for active learning and random sample selection methods in a single run. The train set is the same for both in the first iteration and the result should accordingly be exactly the same.

## 7.2. Cause of the problem

One of the factors that is used to select points in active learning is the distance to the boundary. In a similar way, these first experiments were done using the probability of belonging to a class as that measure of certainty (furthest points having probability = 0 or 1 and closest points having probability = 0.5).

In GAIA, the function `SVMtrain` has a parameter called probability which tells "wheter to train the model for probability estimates (default: `False`)". This parameter was set to `True`.

However, it is actually activating probability estimates what makes the classifier non-deterministic. This problem appears as well in some other libraries

that use libsvm. For example, scikits-learn[12] has similar bindings to those that GAIA has and probability estimates can be activated as well. However, in its online help we can see the following:

*The probability model is created using **cross validation**, so the results can be slightly different than those obtained by predict. Also, it will give meaningless results on very small datasets.*

Indeed, as this cross validation is done taking random seeds for each fold, the results are unpredictable and are just estimation.

## 7.3. Conclusion and solutions

The clearest and most immediate conclusion of this problem is that, when using GAIA, probability estimates **must be disabled** (`probability = False`) if a deterministic behavior is expected.

If `probability` is set to `True`, the corresponding results are estimations after cross validation and must be understood as such. Moreover, they should not be used if the dataset is very small, as cross validation is meaningless in that case.

In this project, the solution was done using scikits-learn. This library has a function (which does not need enabling probability estimates) called `decision_function (x)` which tells directly the distance of point `x` to the boundary.

Alternatively, as just stated, probability estimates could be used as estimations, but the classification should be done with them disabled if results must be deterministic. Again, this would still be problematic if, as in our

[12] http://scikit-learn.sourceforge.net/

problem, we must tell the distance to the boundary when the datasets are very small.