Automatic Music Genre Classification using Ensemble Learning

by

Orgest Xhelili

Bachelor Thesis in Computer Science

Prof. Dr. Herbert Jaeger
Bachelor Thesis Supervisor

Date of Submission: May 9, 2018

Jacobs University — Focus Area Mobility

With my signature, I certify that this thesis has been written by me using only the indicates resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Signature Place, Date

Abstract

Organizing music libraries and databases requires automatic classification of music. Music genres are essential descriptors created by humans to label pieces of music. This paper presents a novel approach of ensemble learning for solving the problem of genre classification. Ensemble learning is a machine learning paradigm where multiple models are trained to solve the same problem and the final result is a combination of the individual results by each model. Our system consists of three different classifiers: echo state networks, support vector machines and logistic regression.

Echo state networks (ESNs) provide a supervised learning principle for analyzing and training recurrent neural networks (RNNs). They present a more practical approach to the usage of RNNs because of their computational efficiency and simple implementation. Support vector machines (SVMs) are supervised learning models which can be used for regression analysis or classification problems.
Logistic regression is a regression model which performs linear classification by estimating probabilities of each of the outcomes.

Contents

1 Introduction

Music genres are labels created by humans to classify pieces of music giving rise to organized music libraries. However, the distinction between genres remains not well defined, as does their definition. Because of the vast amount of musical pieces, automatic genre classification is crucial for organizing huge music databases allowing users to find the music they want to. Automatic genre classification is part of automatic music information retrieval where much research has been ongoing in the 21st century. A summary of state of the art automatic music genre classification may be found on this survey [1].

Echo state networks provide an architecture and supervised learning principle for recurrent neural networks. They present a practical approach to training RNNs because of their simple implementation and computational efficiency. RNNs represent a large and varied class of computational models that are designed by more or less detailed analogy with biological brain modules. In an RNN numerous abstract neurons (also called units or processing elements) are interconnected by likewise abstracted synaptic connections (or links), which enable activations to propagate through the network [2]. ESNs differ from classic RNNs because of the usage of a fixed random reservoir and the training of only output neurons weights [3]. Because there are no cyclic dependencies between the trained readout connections, training an ESN becomes a simple linear regression task [4].

Support vector machines are a set of supervised learning methods used for classification, regression and outliers detection. Given training data labeled in two classes, the SVM algorithm computes an optimal hyperplane to categorize new data. Thus a SVM behaves as a non-probabilistic binary linear classifier. However, they can be extended to support multi class classification of non-linearly separable data in a probabilistic setting. To support non-linear classification, SVMs use the so called kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. To support multi class classification, we use the one-vs.-one strategy, reducing a K class classification problem to $K(K-1)/2$ binary classifiers. Probabilities can be estimated for binary classification using Platt Scaling [5]. For multi class classification the probability scores can be estimated by using a pairwise coupling strategy [6]. It should be noted that the estimation of probabilities is rather an expensive procedure. SVMs are widely used nowadays for different machine learning problems because of their effectiveness, versatility and efficiency.

Logistic regression sometimes called the logistic model or logit model, analyzes the relationship between multiple independent variables and a categorical dependent variable by estimating probabilities of occurrence of an event [7]. In its classical form, logistic regression is used as a binary classifier. In this case the dependent variable is dichotomous. When the dependent variable is not dichotomous, a multinomial logistic regression can be applied.

The report is organized as follows. The second section states the motivation of this research and describes in more detail the problem of the automatic music genre classification problem. The third section gives a more in-depth analysis of ESNs, SVMs and logistic regression. The fourth section documents the ensemble learning strategy used to solve the problem of automatic music genre classification. This section also gives a summary of the state of the art and the dataset used. The fifth section documents the evaluation criteria and the results of our experiment. The last section discusses the results and potential focus areas for future research in the topic.

2 Statement and Motivation of Research

2.1 Music Genre Classification

The basis of automatic music genre classification is the representation of musical pieces and the extraction of feature vectors from the agreed representation. A symbolic representation of musical pieces is rarely available so one has to deal with audio samples. This representation presents a good strategy as it places music genre classification problem in the sphere of speech recognition where more research has been done. However, using the exact waveform of a musical piece for automatic classification is not feasible, because of the low level information contained in the audio samples. Therefore the first step in automatic genre classification is the extraction of useful features from the audio representation of musical pieces. Much work on extraction of features from music has been devoted to timbral texture features, rhythmic content features and pitch content features [8].

2.2 Audio Features

- Timbral texture features are used to differentiate sounds with the same pitch and loudness from each other. The use of timbral texture features originates from speech recognition [9]. A detailed list of features used to characterize timbre may be found in [10]. These features are usually referred to as being low-level as they represent sound on samples of milliseconds. We summarize here the main low-level features used in genre classification problems as given in [1]:
    - temporal features: features computed from the audio signal frame (zero-crossing rate and linear prediction coefficients)
    - energy features: features that describe the energy content of the signal
    - spectral shape features: features that describe the shape of the power spectrum of a signal frame: centroid, spread, skewness, kurtosis, slope, roll-off frequency, Mel-frequency cepstral coefficients (MFCC)
- Rhythmic content features are the most widely used mid-level features in audio-based music classification [11]. These features are used to characterize the temporal regularity of a musical piece. A review of automatic rhythm description systems may be found in [12]. Tzanetakis and Cook proposed the calculation of rhythmic content features for genre classification by extracting periodic changes from the beat histogram [13]. The beat histogram models the distributions of the regularities exhibited in the envelop signal, where rhythmic features can be obtained such as magnitudes and locations of dominant peaks and BPM(beat-per-minute) [11].
- Pitch content features are another set of important mid-level music features. Pitch is determined by what the ear judges to be the most fundamental frequency of the sound [14]. However, the perception of pitch is completely subjective which makes pitch not equal to the fundamental frequency. Tzanetakis and Cook proposed the calculation of pitch content features by extracting periodic changes from the pitch histogram [13]. The pitch histogram models the distribution of candidate pitches extracted by all frames [11].

Analysis and Texture Window

Most of the low-level features are computed at regular time intervals, over short windows of length 10-100ms. These segments, called analysis windows, have to be small enough so that the frequency characteristics of the magnitude spectrum are relatively stable [13]. However, in order to capture the long term nature of sound ”texture”, means and variances of the extracted features can be computed over a number of analysis windows. This larger window is referred to as a texture window [13]. In our system, an analysis window of 100ms and a texture window of 2s is used.

2.3 Extracted Features

This paper uses a variety of low-level features to extract from the audio representation of musical pieces. As documented in two surveys for automatic music classification [1, 11], the low-level features (especially MFCCs) give the best results for the problem of genre classification. We provide below a description of each of the features used in our experiment.

- MFCC - This paper uses Mel-frequency cepstral coefficients (MFCCs) as the most important feature to extract from the audio representation of musical pieces. MFCCs are a feature set popular in audio processing. MFCCs fall in the category of timbral texture features that describe the shape of the power spectrum of a signal frame. MFCCs features are based on the short time Fourier transform (STFT). This feature set is obtained as follows: We first frame the audio signal in multiple windows. For each frame the logarithm of the amplitude spectrum based on STFT is computed. Usually the frequencies are divided in 13 bins using Mel-frequency scaling giving rise to 13 coefficients. However, the usage of 20 coefficients has shown to give better results [15]. Therefore we are using 20 MFCCs in this experiment. In the end we apply Discrete Cosine Transform to obtain the feature vectors.

- Short-time Energy - The short-time energy is the signal energy over a certain audio frame. It is computed as:

$$E = \sum_{m=1}^{N} x^2(m) \tag{1}$$

  where E represents the energy of the signal $x(m)$ [16].

- Short-time Energy Entropy - This feature can be interpreted as a measure of abrupt changes in the signal energy. It is computed as:

$$H = -\sum_{t=1}^{N} E_n[t] \cdot log_2(E_n[t]) \tag{2}$$

  where $E_n[t]$ is the normalized energy over a sub-frame t.

- Zero Crossing Rate - This feature describes the rate at which the signal changes its sign. It is computed as:

$$zcr = \frac{1}{N} \sum_{t=1}^{N} sign(x_t x_{t-1}) \tag{3}$$

  where x is a signal of length N ans $sign$ is an indicator function.

- Spectral Centroid - The spectral centroid is defined as the center of gravity of the magnitude spectrum of STFT [13]. It is computed as:

$$C_t = \frac{\sum_{n=1}^{N} M_t[n] \cdot n}{\sum_{n=1}^{N} M_t[n]} \tag{4}$$

  where $M_t[n]$ is the magnitude of STFT at frame t and frequency bin n.

- Spectral Spread - The spectral spread is defined as the variance of the magnitude spectrum of STFT. It is computed as:

$$S_t = \sqrt{\sum_{n=1}^{N} \frac{(C_t - n)^2 \cdot M_t[n]}{M_t[n]}} \tag{5}$$

  where C_t and $M_t[n]$ are defined as in equation 4.

- Spectral Entropy - This feature is similar to Energy Entropy. In contrast the entropy is calculated over sub-frames of the spectrum, not the signal itself.

- Spectral Flux - The spectral flux is defined as the squared difference between the normalized magnitudes of successive spectral distributions [13]. It is computed as:

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2 \tag{6}$$

  where $N_t[n]$ denotes the normalized magnitude of the Fourier transform at frame t and frequency bin n.

- Spectral Rolloff - The spectral rolloff is defined as the frequency R_t below which 85% of the magnitude distribution is concentrated [13]. It is computed as:

$$\sum_{n=1}^{R_t} M_t[n] = 0.85 \cdot \sum_{n=1}^{N} M_t[n] \tag{7}$$

  where $M_t[n]$ is defined as in equation 4.

- Delta-Cepstral Coefficients - These features serve as the first order derivative of MFCCs and are appended to MFCCs as documented below.
  Given MFCCs $C[n]$ over an analysis window, we calculate the delta features

$$D[n] = C[n] - C[n-1] \tag{8}$$

  where n is the index of the analysis frame. It has been shown that the usage of delta-cepstral coefficients produces better results in the field of speech recognition [17].

The extraction of features for the experiment was done using pyAudioAnalysis [18].

2.4 Research Objectives

I state that this guided research does not aim to achieve state of the art solution to the problem of music genre classification. The main objective of the research is to apply a novel ensemble learning approach to this problem, by combining different classifiers. In the end, we provide a comparison of the performances of the different classifiers.

3 Ensemble of Classifiers

3.1 Echo State Networks

The following section documents a summary of ESNs as introduced in [3, 19, 20].
Echo state networks provide an architecture and supervised learning principle for recurrent neural networks. The main idea is to drive a random, large, fixed recurrent neural network with the input signal, thereby inducing in each neuron within this reservoir network a nonlinear response signal, and combine a desired output signal by a trainable linear combination of all of these response signals [3].

We consider discrete-time neural networks with K input units, N internal network units and L output units. Activations of input units at time step n are K-dimensional vectors $\mathbf{u}(\mathbf{n}) = (u_1(n)...u_K(n))^T$, of internal units N-dimensional vectors $\mathbf{x}(\mathbf{n}) = (x_1(n)...x_N(n))^T$, and of output units L-dimensional vectors $\mathbf{y}(\mathbf{n}) = (y_1(n)...y_L(n))^T$. Real-valued connection weights are collected in a $N \times K$ weight matrix $\mathbf{W^{in}}$ for the input weights, in a $N \times N$ matrix $\mathbf{W}$ for the internal connections, and in a $L \times N$ matrix $\mathbf{W^{out}}$ for the connections to the output units. The basic architecture of the network is shown in Figure 1.

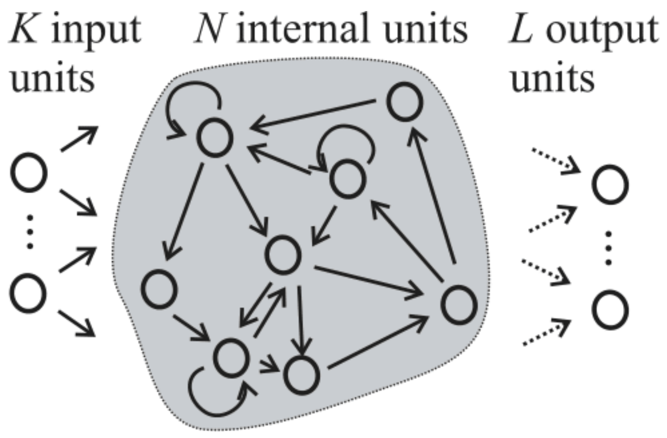


Figure 1: Architecture of a network with K input units, N internal units and L output units (graphics taken from [20]).

In this project we are using the version of ESNs with leaky integration as described in [21]. Using this particular type of the network the activation of internal units follows the equation:

$$\mathbf{x}(n+1) = (1-\alpha)\mathbf{x}(n) + \alpha\mathbf{f}(\mathbf{W}\mathbf{x}(n) + \mathbf{W^{in}}\mathbf{u}(n+1) + \mathbf{b_N}) \tag{9}$$

where $\mathbf{b_N}$ is a fixed random N dimensional bias vector, α is the leaking rate of the network and $\mathbf{f} = (f_1...f_N)$ are the internal unit's output functions. In our case we will be using $\mathbf{tanh}$ as this function. The output is updated according to the equation:

$$\mathbf{y}(n+1) = g(\mathbf{W^{out}}\mathbf{x}(n+1)) \tag{10}$$

where $\mathbf{g}$ is an output activation function. In our case we will be using the identity function so g will be ignored.

Training ESNs

In the stage of training, the network is driven by a given teacher input signal $\mathbf{u}(1), ..., \mathbf{u}(n_{train})$ which yields a sequence $\mathbf{x}(1), ..., \mathbf{x}(n_{train})$ of network states. The obtained network states are collected into a matrix $\mathbf{X}$ of size $N \times (n_{train} - n_0)$, where n_{train} is the training length and n_0 is the washout time. Along the teacher input signal, a teacher output signal is provided. The desired outputs $\mathbf{y}(n)$ are sorted row-wise into a teacher output collection matrix $\mathbf{Y_t}$ of size $(n_{train} - n_0) \times L$ [3]. Then the output weights $\mathbf{W^{out}}$ are trained using linear regression following the equation:

$$\mathbf{W^{out}} = \mathbf{Y_t X^T}(\mathbf{XX^T} + \gamma^2 I)^{-1} \tag{11}$$

where γ^2 is the regularization coefficient and I is the identity matrix. After training the output weight by linear regression the network is ready for testing.

Achieving Echo States

Under certain conditions, the activations state $\mathbf{x}(n)$ of a recurrent neural network are a function of the input history presented to the network [19]. More precisely there exists an echo function $\mathbf{E}$, such that the activation state of the network is calculated as:

$$\mathbf{x}(n) = \mathbf{E}(..., \mathbf{u}(n-1), \mathbf{u}(n)) \tag{12}$$

We give here a proposition from [19] which gives sufficient conditions for the existence and the non-existence of echo states.

Proposition 1 *Assume a sigmoid network with unit output functions $f_i = tanh$. Let the weight matrix $\mathbf{W}$ have its largest singular value $\sigma_{max} < 1$. Then the network has echo states for all admissible inputs. Let the weight matrix have a spectral radius $|\lambda_{max}| > 1$, where the spectral radius denotes the eigenvalue of the weight matrix with the largest absolute value. Then this network has no echo states if $\mathbf{u}(n) = 0$ is an admissible input sequence.*

Note that scaling the weight matrix $\mathbf{W}$ with a scalar α scales the spectral radius and the singular values accordingly. In practice it is enough to achieve echo states if we have a spectral radius marginally smaller than 1. This would be achieved by scaling the matrix $\mathbf{W}$ with $\beta/|\lambda_{max}|$ where β is marginally smaller than 1.

3.2 Support Vector Machines

Support Vector Machines (SVMs) are a popular machine learning method for classification, regression and other learning tasks. The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. The current standard version was initially proposed by Cortes and Vapnik in 1993 and published in 1995 [22]. The following sections explain the SVM algorithm as presented in [23, 24, 25].

Motivation behind SVMs

The basic idea of the SVM algorithm is to find an optimal hyperplane for linearly separable data points. Given linearly separable training data points $\mathbf{x}_i \in R^n, i = 1, ..., l$, where each $\mathbf{x}_i$ belongs in one of the two classes $\mathbf{y}_i = -1$ or $+1$, the SVM algorithm tries to find an optimal hyperplane separating the two classes [23]. This hyperplane can be defined formally as:

$$\mathbf{w} \cdot \mathbf{x} + \mathbf{b} = 0 \tag{13}$$

where $\mathbf{w}$ is a weight vector normal to the hyperplane and $\mathbf{b}$ is a bias vector. Support vectors are the data points closest to this separating hyperplane and the aim of the algorithm is to maximize the distance of the hyperplane to the support vectors [24]. This distance is called the margin of the separating hyperplane.

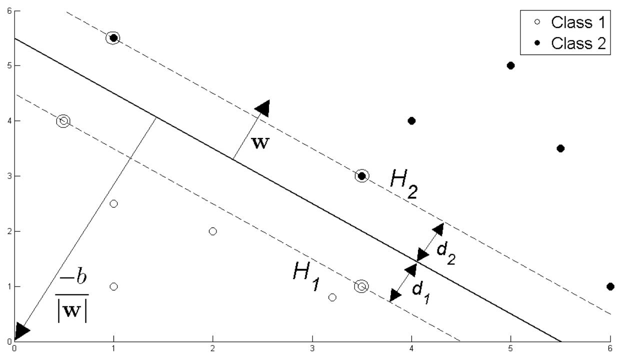


Figure 2: Hyperplane through two linearly separable classes (graphics taken from [23]).

The figure above shows the optimal hyperplane through two linearly separable classes. H_1 and H_2 denote the planes on which the support vectors lie on. The margin of the hyperplane is $m = |d_1| + |d_2|$. Referring to this figure, implementing the SVM boils down to selecting variables $\mathbf{w}$ and $\mathbf{b}$ such that:

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} \geq +1 \quad \forall \mathbf{y}_i = +1 \tag{14}$$

$$\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b} \leq -1 \quad \forall \mathbf{y}_i = -1 \tag{15}$$

The equations can be combined into:

$$\mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b}) - 1 \geq 0 \quad \forall i \tag{16}$$

As we said earlier the SVM algorithm aims to maximize the margin m. Simple vector geometry shows that $m = \frac{2}{||\mathbf{w}||}$ [23]. Maximizing m is equivalent to minimizing $\frac{1}{2}||\mathbf{w}||^2$, which leads to solving the following constrained optimization problem:

$$min\ \frac{1}{2}||\mathbf{w}||^2 \quad s.t. \quad \mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b}) - 1 \geq 0 \quad \forall i \tag{17}$$

The solution to this problem and the mathematical formalism behind it are documented in [24]. We ignore the step by step solution and present just the result. The original problem 17, known as the primal problem, is equivalent to the following dual problem [24]:

$$max\ W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{l} \alpha_i \alpha_j \mathbf{y}_i \mathbf{y}_j \mathbf{x}_i \mathbf{x}_j \quad s.t \quad \alpha_i \geq 0, \sum_{i=1}^{l} \alpha_i \mathbf{y}_i = 0 \tag{18}$$

This is a quadratic programming problem and a solution $\alpha = (\alpha_i)$ can be found using different approaches. For SVM, sequential minimal optimization seems to be the most popular [24]. A nice characteristic of the solution is that many of the α_i are 0 and the $\mathbf{x}_i$ with non-zero α_i are the support vectors. Therefore we can construct $\mathbf{w}$ as a linear combination of a small number of data points [24]:

$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} \mathbf{y}_{t_j} \mathbf{x}_{t_j} \tag{19}$$

where $t_j (j = 1, ..., s)$ are the indices of the s support vectors. After the optimal hyperplane is computed, the SVM classifier can categorize new data $\mathbf{z}$ by looking at the sign of the expression:

$$f = \mathbf{w} \cdot \mathbf{z} + \mathbf{b} \tag{20}$$

and categorize it as class 1 if the f is positive and class -1 otherwise.

Extension to non-linearly Separable Data

In order to extend the SVM methodology to handle data that is not fully linearly separable, we relax the constraints 14 and 15 to allow for misclassified points [23]. This is done by introducing positive slack variables ξ_i, $i = 1, ..., l$ and the new equations can be combined into:

$$\mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b}) - 1 + \xi_i \geq 0 \quad \text{where} \quad \xi_i \geq 0 \; \forall i \tag{21}$$

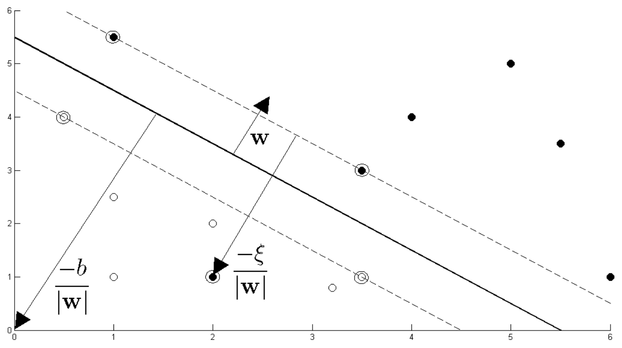


Figure 3: Hyperplane through two non-linearly separable classes (graphics taken from [23]).

In this SVM version, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it. The correctly classified data points have $\xi_i = 0$. As we are trying to reduce the number of misclassifications, we adapt the previous objective function 17 in the following way:

$$min \; \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i=1}^{l} \xi_i \quad s.t. \quad \mathbf{y}_i(\mathbf{x}_i \cdot \mathbf{w} + \mathbf{b}) - 1 + \xi_i \geq 0 \; \forall i \tag{22}$$

where the parameter C controls the trade-off between the slack variable penalty and the size of the margin. The new constrained optimization problem can be solved in the same way as in the linear separable case, except there is an upper bound C on α_i now [24]. The dual problem in this case is:

$$max\ W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{l} \alpha_i \alpha_j \mathbf{y}_i \mathbf{y}_j \mathbf{x}_i \mathbf{x}_j \quad s.t \quad C \geq \alpha_i \geq 0,\ \sum_{i=1}^{l} \alpha_i \mathbf{y}_i = 0 \tag{23}$$

The weight vector $\mathbf{w}$ can be constructed in the same way as in the linearly separable case.

Extension to non-linear Decision Boundary

Many datasets cannot be separated by a linear hyperplane. The idea is to transform these data points into a higher dimensional feature space which is linearly separable. After the transformation, the SVM algorithm can be applied to the transformed data points. Computation in this new space can be quite costly but this is not a problem for the SVM, since we only calculate the inner product of the data points. As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly [24]. This is the so called kernel trick. We define the kernel function K:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) \tag{24}$$

where $\phi(\mathbf{x})$ is a map which transforms the data points into the linearly separable feature space. For the linear decision boundary case, ϕ is the identity function. The dual problem in the case of non-linearly decision boundary transforms to:

$$max\ W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{l} \alpha_i \alpha_j \mathbf{y}_i \mathbf{y}_j K(\mathbf{x}_i, \mathbf{x}_j) \quad s.t \quad C \geq \alpha_i \geq 0,\ \sum_{i=1}^{l} \alpha_i \mathbf{y}_i = 0 \tag{25}$$

The new data $\mathbf{z}$ can be categorized by looking at the sign of the expression:

$$f = \mathbf{w}\phi(\mathbf{z}) + \mathbf{b} = \sum_{j=1}^{s} \alpha_{t_j} \mathbf{y}_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + \mathbf{b} \tag{26}$$

where t_j is defined as in 19. The kernel function, being an inner product, can be viewed as measure of similarity between two data points. We present here 3 popular examples of kernel functions:

- Polynomial kernel function with degree d: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}\mathbf{y} + 1)^d$
- Radial basis function with width ψ: $K(\mathbf{x}, \mathbf{y}) = e^{\frac{-||\mathbf{x}-\mathbf{y}||^2}{2\psi^2}}$
- Sigmoid function with parameters κ and θ: $K(\mathbf{x}, \mathbf{y}) = tanh(\kappa\mathbf{x}\mathbf{y} + \theta)$

In conclusion, training a SVM includes selecting a kernel function, choosing the kernel parameters and the value of C and solving a quadratic programming problem.

Multi-class Classification

By their design SVMs are binary classifiers. To support multi-class classification, we use a so called one-vs.-one strategy. If k is the number of classes, $\frac{k(k-1)}{2}$ SVMs are constructed as described above and each one of them trains data from two classes solving the quadratic programming problem 25. For classification a voting strategy is used. Each binary classifier casts a vote as described in 26 and in the end the class with the most votes is chosen as the predicted class. In case that two or more classes have identical votes, a random choice is made between these classes.

Estimating Probabilities

The SVM algorithm predicts only class labels without probability information. Because we aim to apply an ensemble learning approach, we need to extend the SVM algorithm to give probability estimates. We are using LIBSVM [26] for constructing the SVM classifier. Based on their implementation the probability weights for multi-class classification are estimated as follows:

Given k classes of data, for any $\mathbf{x}$, pairwise class probabilities are estimated first based on [27]:

$$r_{ij} = P(\mathbf{y} = i|\mathbf{y} = i \text{ } or \text{ } j, \mathbf{x})$$

If u is the decision value at $\mathbf{x}$, we estimate r_{ij} by the equation:

$$r_{ij} = \frac{1}{1 + e^{Au+B}} \tag{27}$$

where A and B are estimated by minimizing the negative log likelihood of training data. It has been observed that decision values from training might overfit the model 27, so a five-fold cross-validation scheme is conducted to obtain the decision values before estimating A and B. After collecting all r_{ij} values, the following optimization problem is solved as proposed by [6]:

$$\underset{\mathbf{P}}{min} \ \frac{1}{2}\sum_{i=1}^{k}\sum_{j:j\neq i}(r_{ji}p_i - r_{ij}p_j)^2 \quad subject \ to \quad \forall i \ p_i \geq 0, \quad \sum_{i=1}^{k} p_i = 1 \tag{28}$$

After solving this optimization problem, the SVM can give probability estimates p_i for each class. A summary of the algorithm to solve the problem 28 is documented on [26] while more details are presented on [6].

3.3 Logistic Regression

We present in this section a summary of the logistic regression model as described in [28]. Regression methods are a set of statistical processes for estimating the relationship between a response variable and one or more explanatory variables. Logistic regression has become the standard model of analysis in the situation when the outcome of the response variable is discrete. What distinguishes logistic regression from linear regression is that the outcome variable in logistic regression is dichotomous.

Motivation behind Logistic Regression

The logistic regression model is used to estimate the probability of a binary response based on ore more predictors. We give first a definition of the logistic function. The logistic function σ is a sigmoid function which takes a real input and outputs a value between zero and one:

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t} \tag{29}$$

Assume that we have l training data points $(\mathbf{x}_i, \mathbf{y}_i)$ where $\mathbf{x}_i \in R^n$ and $\mathbf{y}_i = 1$ or 0. The value of $\mathbf{y}$ 1 means that a data point belongs to one class and the value 0 denotes the membership to the other class. We can treat the membership to a certain class as a dichotomous variable and use the logistic regression model to estimate the probability of the membership of a data point $\mathbf{x}$ in the 'first' class:

$$p(\mathbf{x}) = P(y = 1|\mathbf{x}) = \sigma(g(\mathbf{x})) \tag{30}$$

where g is the inverse of the logistic function also called the logit function:

$$g(\mathbf{x}) = ln(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}) = \mathbf{w}_0 + \mathbf{w}_1\mathbf{x}_1 + ... + \mathbf{w}_n\mathbf{x}_n \tag{31}$$

The assumption that $g(\mathbf{x})$ is a linear combination of coordinates of $\mathbf{x}$ comes from linear regression. It remains for the logistic regression model to choose an optimal weight vector $\mathbf{w} \in R^{n+1}$.

The method of estimation used in logistic regression is maximum likelihood. This method yields values for the unknown parameters which maximize the probability of obtaining the observed data. In order to apply this method we construct first the likelihood function as follows:

$$l(\mathbf{w}) = \prod_{i=1}^{l} p(\mathbf{y}_i|\mathbf{x}_i) = \prod_{i=1}^{l} p(\mathbf{x}_i)^{\mathbf{y}_i}(1-p(\mathbf{x}_i))^{1-\mathbf{y}_i} \quad . \tag{32}$$

The principle of maximum likelihood is to select the estimate of $\mathbf{w}$ which maximizes the likelihood function. However, it is easier mathematically to maximize the log of the likelihood function. This expression, known as log likelihood, is defined as:

$$L(\mathbf{w}) = \sum_{i=1}^{l} \mathbf{y}_i ln[p(\mathbf{x}_i)] + (1-\mathbf{y}_i) ln[1-p(\mathbf{x}_i)] \quad . \tag{33}$$

Maximizing the log likelihood require special mathematical software which is present in most statistical packages. We do not present here an algorithm for maximizing 33 but such a solution is known to be found.

Classification

Having found $\mathbf{w}$ that maximizes 33, we can proceed to classifying testing data. For a new data point $\mathbf{z}$, we calculate $p(\mathbf{z}) = \frac{1}{1+e^{-\mathbf{wz}}}$. If $p(\mathbf{z}) \geq 0.5$, we classify $\mathbf{z}$ in the first class; otherwise $\mathbf{z}$ belongs to the other class. Here we use the convention of letting $\mathbf{z}_0 = 1$, so that $\mathbf{wz} = \mathbf{w}_0 + \sum_{j=1}^{n} \mathbf{w}_j\mathbf{z}_j$.

L_2 Regularization

To prevent overfitting, we use a L_2-regularized version of logistic regression. We are using LIBLINEAR [29] for constructing the logistic regression model. Based on their implementation, the model solves the following optimization problem:

$$\underset{\mathbf{w}}{min} \ \frac{1}{2}\mathbf{w}^2 + C\sum_{i=1}^{l} \xi(\mathbf{w};\mathbf{x}_i;\mathbf{y}_i) \tag{34}$$

where $\xi(\mathbf{w};\mathbf{x}_i;\mathbf{y}_i)$ is a loss function and $C \geq 0$ is the trade-off parameter as in the SVM model. We should note here that the data is labeled differently with $\mathbf{y}_i \in \{-1, 1\}$. In this case, it is clear that we can assign $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{yx})$. As we said before the aim of the model is to maximize the log likelihood. As a loss function, this corresponds to minimizing the negative of the log likelihood. We calculate in this way the loss function to be minimized:

$$J(\mathbf{w}) = -ln(\prod_{i=1}^{l} p(\mathbf{y}_i|\mathbf{x}_i)) = -\sum_{i=1}^{l} ln(p(\mathbf{y}_i|\mathbf{x}_i)) = \sum_{i=1}^{l} ln(\sigma(\mathbf{y}_i\mathbf{w}\mathbf{x}_i)^{-1}) = \sum_{i=1}^{l} ln(1+e^{-\mathbf{y}_i\mathbf{w}\mathbf{x}_i}) \tag{35}$$

which shows that $\xi(\mathbf{w};\mathbf{x}_i;\mathbf{y}_i) = ln(1 + e^{-\mathbf{y}_i\mathbf{w}\mathbf{x}_i})$. Problem 34 is solved by LIBLINEAR by implementing a trust region Newton method as described in [30]. After finding the optimal solution $\mathbf{w}$ we classify new data $\mathbf{z}$ by looking at the sign of the expression $\mathbf{wz}$.

Multi-class Classification

Logistic regression is a probabilistic binary classifier. To support multi-class classification, we use a so called one-vs.-rest strategy. If k is the number of classes, k binary models are fit (one for each class). Each binary model solves the optimization problem 34 by treating samples of each class as the positive outcome and all the other samples as the negative outcome. For a testing data point $\mathbf{z}$, we calculate the probability of the membership to each class. In the end the k probability scores are normalized and the class with the highest probability is chosen as the predicted one.

3.4 Ensemble Learning

Ensemble learning algorithms take a different approach from the classical machine learning algorithms. Rather than constructing one best hypothesis to solve a certain problem, ensemble systems construct a set of hypotheses and combine these hypotheses to solve the original problem [31]. Experimental evidence has shown that ensemble methods are often more accurate than any single hypothesis. Learning algorithms that output only a single hypothesis might suffer from three problems that can be partly overcome by ensemble methods [31]:

- the statistical problem - This problem arises when the learning algorithm is searching a space of hypotheses that is too large for the amount of available training data. In such cases, there may be several good hypotheses to be chosen from the hypothesis space. A simple vote of equally-good classifiers can improve the performance of the model. A learning algorithm that suffers from this problem is said to have high ”variance”.
- the computational problem - The computational problem arises when the learning algorithm cannot guarantee to find the best hypothesis within the hypothesis space. For example, the procedure of training neural networks with gradient descent can

get stuck in local minima. A weighted combination of several different local minima can reduce the risk of choosing the wrong local minima to output.

- the representation problem - This problem arises when the hypothesis space does not contain any hypotheses that are good approximations of the true function f. By taking a weighted vote of hypotheses, the learning algorithm may be able to form a more accurate approximation of f. A learning algorithm that suffers from this problem is said to have high ”bias”.

Experimental measures have shown that ensemble methods can reduce both the variance and the bias of learning algorithms [31].

In the context of our problem, the ensemble system consists of 3 different classifiers (ESNs, SVMs, LR) which are trained individually to solve the problem of automatic music genre classification. Each of these classifiers predicts a 10 dimensional probability vector for each song in the validation sets. We combine these probability scores by a soft voting strategy. Assume that for a testing data point $\mathbf{z}$, the three classifiers output respectively the three probability vectors $p_{\mathbf{z}_1},\ p_{\mathbf{z}_2},\ p_{\mathbf{z}_3}$. We calculate a weighted sum of these probabilities:

$$r_{\mathbf{z}} = \mathbf{w}_1 p_{\mathbf{z}_1} + \mathbf{w}_2 p_{\mathbf{z}_2} + \mathbf{w}_3 p_{\mathbf{z}_3} \tag{36}$$

where $r_{\mathbf{z}}$ denotes the final hypothesis for a data point $\mathbf{z}$. The predicted genre by the ensemble system for $\mathbf{z}$ is the index of the highest value in $r_{\mathbf{z}}$. The weights $\mathbf{w}_1,\ \mathbf{w}_2,\ \mathbf{w}_3$ are also learned by a cross validation scheme.

4 Documentation of Methods

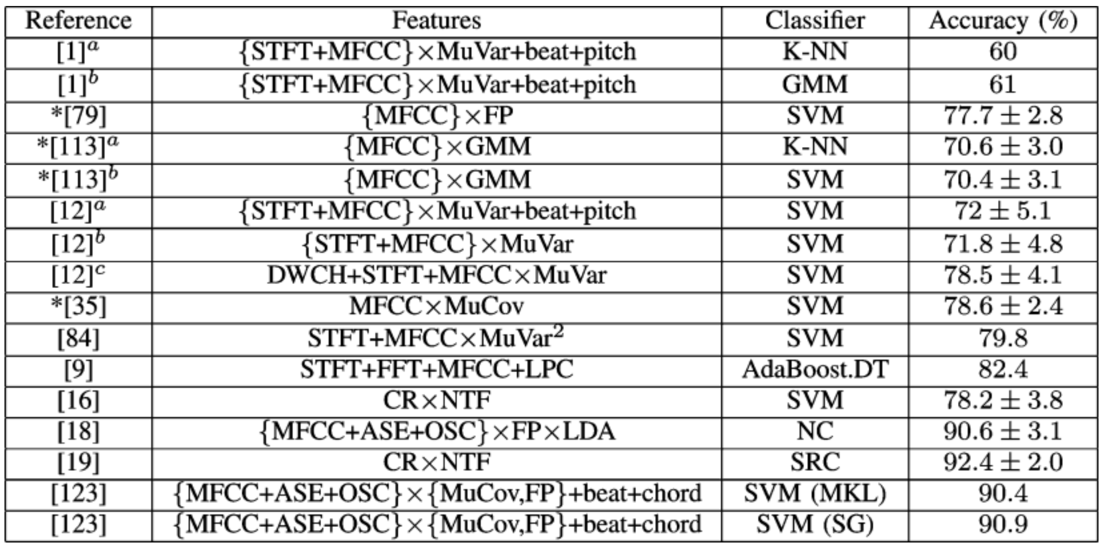
4.1 Problem and Dataset Description

The experiments are based on the famous standard dataset used by Tzanetakis and Cook [13]. This dataset consists of 1000 30s long audio files which have been annotated a genre. The audio files cover 10 different genres with 100 files per each genre. The ten represented genres are: Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae and Rock.

Given this dataset, we aim to train three different classifiers and apply a 10-fold cross validation scheme to classify the dataset into the given genres. Performance of each classifier is analyzed. In the end we combine the three classifiers by a soft-voting strategy and analyze the performance of the ensemble.

4.2 State of the Art

In our context, state of the art refers to the performance of the classifiers applied on the dataset that we are using. The following table, taken from [11], compares the performance of different algorithms using different type of features. The first results come from the creators of the dataset who achieved up to 60% classification accuracy. The most used classifier are support vector machines. The MFCC feature set is used in almost every occasion. Most of the documented classifiers achieve mean classification accuracy in the range of 70-80%.



Reference	Features	Classifier	Accuracy (%)
[1][a]	$\{$STFT+MFCC$\}\times$MuVar+beat+pitch	K-NN	60
[1][b]	$\{$STFT+MFCC$\}\times$MuVar+beat+pitch	GMM	61
*[79]	$\{$MFCC$\}\times$FP	SVM	77.7 ± 2.8
*[113][a]	$\{$MFCC$\}\times$GMM	K-NN	70.6 ± 3.0
*[113][b]	$\{$MFCC$\}\times$GMM	SVM	70.4 ± 3.1
[12][a]	$\{$STFT+MFCC$\}\times$MuVar+beat+pitch	SVM	72 ± 5.1
[12][b]	$\{$STFT+MFCC$\}\times$MuVar	SVM	71.8 ± 4.8
[12][c]	DWCH+STFT+MFCC$\times$MuVar	SVM	78.5 ± 4.1
*[35]	MFCC$\times$MuCov	SVM	78.6 ± 2.4
[84]	STFT+MFCC$\times$MuVar^2	SVM	79.8
[9]	STFT+FFT+MFCC+LPC	AdaBoost.DT	82.4
[16]	CR$\times$NTF	SVM	78.2 ± 3.8
[18]	$\{$MFCC+ASE+OSC$\}\times$FP$\times$LDA	NC	90.6 ± 3.1
[19]	CR$\times$NTF	SRC	92.4 ± 2.0
[123]	$\{$MFCC+ASE+OSC$\}\times\{$MuCov,FP$\}$+beat+chord	SVM (MKL)	90.4
[123]	$\{$MFCC+ASE+OSC$\}\times\{$MuCov,FP$\}$+beat+chord	SVM (SG)	90.9

Figure 4: Performance comparison on the standard dataset

4.3 Pre-processing

The procedure of feature extraction differs for ESNs and the other two classifiers. We feed continuous live data to the network while the features fed to the SVM and logistic regression algorithm are 'global' features computed for the whole song. We document below the two different strategies used in the pre-processing phase.

- ESNs - For each audio file, the features described in 2.3 are computed for each analysis window of 100ms long. We are calculating 20 MFCCs, 20 delta-cepstral coefficients and 8 single features. After the feature extraction procedure, each audio file is represented as a matrix $\mathbf{A}$ of size $48 \times f$, where f is the number of frames of length 100ms for each audio file.
- SVMs & Logistic Regression - The extraction of 'global' features for these two classifiers is based upon the above procedure for ESNs. We compute the mean and the variance of the 48 aforementioned features (calculated for each analysis window) over each texture window of 2s. Therefore, 96 features are computed for each texture window. In the next step, the mean of these features is computed over the whole song segment, providing a global feature set of size 96. In the end, we append to this feature set the covariance matrix of the 20 MFCCs calculated over each analysis window. The usage of the covariance matrix of MFCCs led to better results using SVMs as a classifier [15]. Finally, we have computed for each song a feature set of size 496 which is fed to both the SVM and logistic regression algorithms.

4.4 Cross Validation of Parameters

A 10-fold cross validation is used to determine the parameters of the classifiers that achieve the best performance. The dataset is divided in 10 random smaller sets of 100 audio files each. Each of these subsets is used iteratively as a validation set, while the remaining data is used for training each of the classifiers. For each classifier configuration, the mean classification accuracy over the 10 validation sets is computed and the configurations with the highest accuracy are chosen as the optimal ones. After the parameters of each classifier are decided, the weights of each classifier in the ensemble system are decided in the same fashion.

4.5 ESN Procedure

The feature set fed to the network is of size 48 so our network consists of $K = 48$ input units. We are using the network to classify each musical piece into one of the 10 genres. Therefore, the network will $L = 10$ output units.

Constructing the Network

Weight matrices $\mathbf{W^{in}}$, $\mathbf{W}$ and the bias vector $\mathbf{b_N}$ are randomly initialized with real numbers in the interval $[-0.5, 0.5]$. The spectral radius λ_{max} of the weight matrix $\mathbf{W}$ is computed and we scale the matrix with the inverse of this factor. After this operation, $\mathbf{W}$ will have a spectral radius of 1.

Training the Network

For each run, the network is trained using the state update equation 9. For each musical piece, we ignore the first 30 frames from its matrix representation $\mathbf{A}$. Upon starting training a new musical piece the network state is initialized to 0. The system states are collected in a matrix $\mathbf{X}$ of size $N \times (n_{train} - n_0)$. In our case $n_{train} = 900 \cdot f$ and $n_0 = 900 \cdot 30$, where f is the number of frames for each musical piece. The output teacher matrix $\mathbf{Yt}$ will be of size $L \times (n_{train} - n_0)$. Each column vector of $\mathbf{Yt}$ consists of zeros and a 1 on the right genre. By using the equation 11 the output weights matrix $\mathbf{W^{out}}$ is computed.

Testing the Network

Using state update equations 9 and 10, the network is tested on each validation set. For each musical piece in the validation set, the computed output vectors are saved in a matrix of size $L \times (f - 30)$. The mean of this matrix over each row is computed resulting in a L dimensional vector. This vector is saved as the output of the network which is fed to the ensemble system. The index of the highest value is chosen as the predicted genre. For each run we compute the classification accuracy by dividing the number of correct predictions by 100.

4.6 SVM Procedure

Each song is fed to the SVM algorithm as a pair (x, y), where $\mathbf{x}$ is a feature set of size 496 and $\mathbf{y}$ is a label from 1 to 10. For each validation round, the SVM is trained on 900 songs as described in 3.2. In the testing phase, we predict the genre for each of the 100 songs in the validation set. The classification accuracy is computed for each run as in 4.5. Besides the class prediction, we compute for each song probability vectors which will be fed to the ensemble system.

4.7 Logistic Regression Procedure

Each song is fed to the logistic regression model as described in 4.6. For each validation round, the model is trained on 900 songs as described in 3.3. In the testing phase, we compute a probability vector for each of the 100 songs in the validation set. This vector will be fed to the ensemble system. The class with the highest probability is chosen as the predicted genre. The classification accuracy is computed for each run as in 4.5.

4.8 Ensemble Learning Procedure

For each song in the validation set, we have computed three probability scores as explained in 4.5, 4.6, 4.7. We combine the probability scores by a soft voting strategy as documented in 3.4. We compute accordingly the classification accuracy for each run by dividing the number of correct predictions by 100.

5 Results of the Experiment

This section gives a summary of the evaluation criteria and presents the chosen optimal parameters for the system. We also document the results of our experiment.

5.1 Evaluation Criteria

5.2 Results

6 Conclusions

6.1 Future Work

References

[1] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine*, 23(2):133–141, March 2006.

[2] Mantas Lukoeviius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127 – 149, 2009.

[3] H. Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007. revision #183563.

[4] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[5] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[6] Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004.

[7] Hyeoun Park. An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. *Journal of Korean Academy of Nursing*, 43(2):154–164, 2013.

[8] T. Li and M. Ogihara. Toward intelligent music information retrieval. *IEEE Transactions on Multimedia*, 8(3):564–574, June 2006.

[9] L. Rabiner and H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[10] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. 2004.

[11] Z. Fu, G. Lu, K. M. Ting, and D. Zhang. A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia*, 13(2):303–319, April 2011.

[12] Fabien Gouyon and Simon Dixon. A review of automatic rhythm description systems. *Computer music journal*, 29(1):34–54, 2005.

[13] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, Jul 2002.

[14] Laura Williams Macy. *Grove music online*. Macmillan Reference, 2001.

[15] M. Mandel and D. Ellis. Song-level features and svms for music classification. *Proc. Int. Conf. Music Information Retrieval*, 2005.

[16] Madiha Jalil, Faran Awais Butt, and Ahmed Malik. Short-time energy, magnitude, zero crossing rate and autocorrelation measurement for discriminating voiced and unvoiced segments of speech signals. In *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on*, pages 208–212. IEEE, 2013.

[17] Kshitiz Kumar, Chanwoo Kim, and Richard M Stern. Delta-spectral cepstral coefficients for robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4784–4787. IEEE, 2011.

[18] Theodoros Giannakopoulos. pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12), 2015.

[19] H. Jaeger. The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *German National Research Center for Information Technology GMD Technical Report*, 148, Jan 2001.

[20] H. Jaeger. Short term memory in echo state networks. *GMD - German National Research Institute for Computer Science*, 152, Jan 2002.

[21] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20:335–352, April 2007.

[22] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[23] Tristan Fletcher. Support vector machines explained. *Tutorial paper*, 2009.

[24] Martin Law. A simple introduction to support vector machines. *Lecture for CSE*, 802, 2006.

[25] Robert Berwick. An idiots guide to support vector machines (svms). *Retrieved on October*, 21:2011, 2003.

[26] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[27] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C Weng. A note on platts probabilistic outputs for support vector machines. *Machine learning*, 68(3):267–276, 2007.

[28] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[29] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[30] Chih-Jen Lin, Ruby C Weng, and S Sathiya Keerthi. Trust region newton method for logistic regression. *Journal of Machine Learning Research*, 9(Apr):627–650, 2008.

[31] Thomas G Dietterich. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.