Learning to recognize horn and whistle sounds for humanoid robots

Niels W. Backer Arnoud Visser

University of Amsterdam, P.O. Box 94323, 1090 GH Amsterdam

Abstract

The efficiency and accuracy of several state-of-the-art algorithms for real-time sound classification on a NAO robot are evaluated, to determine how accurate they are at distinguishing horn and whistle sounds in both optimal conditions, and a noisy environment. Each approach uses a distinct combination of an audio analysis method and a machine learning algorithm, to recognize audio signals captured by NAO's four microphones. A short summary of three audio analysis preprocessing methods is provided, as well as a description four machine learning techniques (Logistic Regression, Stochastic Gradient Descent, Support Vector Machine, and AdaBoost-SAMME) which could be used to train classifiers which would distinguish whistle and horn signals from background noise. Experimental results show that for each of the acquired data sets, there are multiple high-accuracy solutions available. Actually, the accuracy and precision results were all so high, that a more challenging dataset is needed to determination which method is optimal for this application.

1 Introduction

Although a large portion of AI robotics research is focused mostly on topics of high-level cognitive tasks, like path planning and robot mapping & localization, one of the most important aspects of natural intelligence is its ability to react to trigger-events in the environment. Fluent interaction with environmental signals is not only a requirement for effective Human-Robot Interaction (HRI), but also an important component for systems that have real-world applications. For these reasons, a sound recognition challenge was issued this year by the RoboCup Standard Platform League (SPL) technical committee, which organizes the yearly robot football world cup. The goal of this challenge is to make the humanoid NAO robots used in the SPL recognize audio signals, more specifically predefined start/stop signals and whistle signals, emitted from the sideline of the pitch.

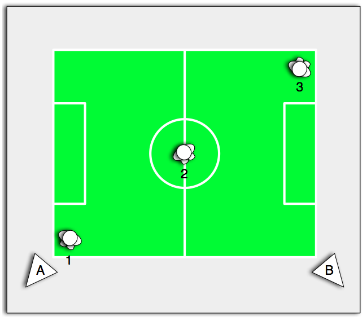


Figure 1: The NAO audio recognition challenge setup. Predefined signals will be emitted in alternating sequences from speakers A and B.

Since the RoboCup is a very noisy event, and NAO's microphones are quite small and of mediocre quality, this challenge is not very straightforward. Because audio classification has been plagued by noise in the past, this challenge calls for a qualitative assessment of a range of techniques applied to signal recognition.

Most digital audio signal processing techniques have a similar method structure. Firstly, the audio signal is converted from analog to digital, after which signal processing techniques are applied to convert the signal to a more compact representation, which contains the features which could be recognized. A classic example of such filter is the Fast Fourier Transform (FFT), which is a very computationally efficient method to find the spectral properties of the signal. An alternative method is the Discrete Wavelet Transform (DWT), which extracts a wavelet power spectrum. Similar to the FFT, DWT can be used to extract frequencies from a signal, but has a distinct advantage for analyzing temporal sequences: It captures both frequency and location (in time) information. The last audio-analysis technique covered in this study is Mel Frequency Cepstral Coefficients (MFCC). Audio analysis using the Cepstral representation of the short-term power spectrum can be used to identify pitch and formant frequencies in speech, and has in fact become a very conventional way to process human speech before recognition.

The second phase is the classification (or recognition) step. While a large number of applications for classifying simple stationary signals have used a frequency mask or single-layer perceptron, the aim of this study is to provide a performance comparison of machine learning algorithms as well. The optimal outcome would be to find a configuration which is impervious to noise, which is a considerable obstacle for common classification methods. In order to combat the effect of noise on audio recognition, we will inspect classification algorithms, some well-established, some quite new, such as logistic regression, stochastic gradient descent, support vector machines, and AdaBoost-SAMME.

The study is organized as follows: Section 2 describes related work. Section 3.1 describes how five datasets were recorded with the Nao robots. Section 3.2 thereafter a short summary of audio analysis techniques is given. Section 3.3 will discuss the machine learning techniques, after which research methods will be discussed, followed by results, a discussion, and notes on future work.

2 Related Research

Sound recognition in robots is one of the major steps in creating autonomous systems that can directly interact with humankind. According to the RoboCup Standard Platform League challenges set out for 2014[1], there is a number of reasons to eliminate WLAN communication within RoboCup soccer matches, with 'not very human-like' being the most important for the Artificial Intelligence field.

There have been previous attempts at tackling the recognition of whistles and predefined horn signals [2], also considering noisy environments [11]. These systems have been proven to be reliable, though they only consider one possible system architecture, using common methods. Both solutions utilize the Fast Fourier Transform (FFT), a frequency mask, and a single-layer perceptron. However, a perceptron without hidden layers can only be trained to recognize a small set of patterns, since its architecture only allows for it to discern instances in models with linearly separable patterns. Attempts at phoneme/speech recognition have successfully used neural networks as an alternative to simple perceptrons [8].

Some speech and dialogue based systems for the NAO also exist. An example of such system is the event-based conversational system for Human-Robot Interaction (HRI) [10]. Though not yet fully autonomously usable, it provides a solid basis for future research. One of the biggest issues in robot audio recognition is background noise. HRI approaches using the NAO built-in conversational system have had some difficulties getting around this barrier [9]. However, more recent research with a fairly simple approach, using the FFT of a signal and logistic regression, has obtained very promising results in noisy environments [14].

3 Research Method

In this section, a description will be given of all methods, their implementation details, and optimizations for the task in hand. Firstly, we will discuss data collection on the NAO robot, in particular the several data sets that were recorded for this study. In the next section all the audio-analysis methods will be set out, as well as the instance vectors they produce for the machine learning phase. A more extensive description of the audio-analysis techniques is available in [1]. This paper will focus on the machine learning methods available to train the classifiers.

[1] http://www.informatik.uni-bremen.de/spl/pub/Website/Downloads/Challenges2014.pdf

3.1 Data Collection

All experiments were conducted using audio recorded indoors, on the NAO robot. The types of audio consist of two predefined horn signals, namely a 'start' and a 'stop' signal, both a sequence of $200Hz$ and $320Hz$ sinusoids, and a complex whistle sound, as used by a referee in any regular football match, which has a broad peak between 6 and $7kHz$. The distance between the robot and the audio source, as well as the level of background noise, were both decided upon after preliminary audio data assessment: the signal should be audible to a human being.

Five data sets were recorded using NAO's microphone array. The SPL challenge requires the recognition of both predefined horn signals and a whistle, brought by the team itself. For both types of signals, two distinct datasets were recorded. Each dataset consists of two recordings, one with a low amount of noise and high signal strength (sets 2 and 4), the other with background noise and a low signal strength, recorded further away from the signal source (sets 1 and 3). Some background noise, containing small amounts of human speech, was also recorded, to serve as the negative samples for training (set 10).

Table 1: Data set description. All data sets were recorded at a 48000 Hz sample rate, using pcm16 encoding, on all 4 NAO microphones simultaneously.

Set	Description	# audio samples	Duration (s)
1	Whistle set, with background noise	4450	56
2	Whistle set, minimal noise	4205	53
3	Horn signal set, with background noise	2210	28
4	Horn signal set, minimal noise	2225	28
5	Whistle data, sets 1 and 2 combined	3289	41
6	Horn signal data, sets 3 and 4 combined	4436	55
7	Noisy data, sets 1 and 3 combined	6661	83
8	Clean data, sets 2 and 4 combined	6430	80
9	All other sets combined	7726	97

The microphones on the NAO provide a 48 kHz audio stream from 4 microphones, situated at the front, rear, and sides of the head. Audio frames were taken at a frequency of 20 Hz, or every 50 ms. The recordings had duration between 28 and 97 seconds. The recordings should not contain any clipping, so periods of silence between actual signals were scrubbed, resulting into between 2210 and 7726 samples where an audio signal was present. All those audio samples were manually marked as positive, $y = 1$, or negative, $y = 0$, to distinguish the target signal from the background noise. Combined data sets were then created, namely sets 5, 6, 7, and 8, one for each type of signal, and one per amount of background noise, by concatenating data from the first four sets into a single file. Finally, a data set containing all types of audio was created, namely set 9.

3.2 Audio Analysis Methods

The datesets contained at least 2000 audio samples, which is a sufficient size not to skip any samples because of the analysis processing time, and more than enough samples to produce a precise representation of the signal. These audio samples were normalized to reduce the effect of the audio volume on the final representation.

3.2.1 Fast Fourier Transform

There is a plethora of FFT algorithms, each of which calculates the Discrete Fourier Transform, or DFT, of a signal. The DFT is a discretized form of a method for expressing a function as a sum of periodic components, and for recovering the function from these components. Using an input signal of N samples, the DFT is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i \frac{kn}{N}} \qquad k = 0, \dots, N-1$$

For this study, the Scientific Python (SciPy) implementation of FFT was used[2], specifically `scipy.fftpack.rfft`. This Real Fast Fourier Transform is an implementation of an adapted form of the Cooley– Tukey algorithm [5], which can only be applied to a series of real values, such as a sound signal. By ignoring the complex conjugates, which would yield negative frequencies, it gains a considerable speed boost. This method yields $\frac{N}{2} + 1$ coefficients, of which the log values are taken, which are more suitable for most machine learning algorithms. The FFT is truncated at around 12 kHz, since early analysis shows that higher frequencies are redundant for the task of whistle recognition. This process generates the 680-feature length vectors used for training and testing our classifiers.

3.2.2 Discrete Wavelet Transform

For this study, the implementation of Discrete Wavelet Transform from the open source PyWavelets Python module, `pywt.WaveletPacket`, was used[3]. Like most implementations of DWT, it calculates the transform of a discrete signal by passing it through a series of filters, using convolution. Instead of analyzing sinusoid levels in a signal, DWT attempts to find wavelets, which are wave-like oscillations with an amplitude that begins at zero, increases, and then decreases back to zero. There are many wavelet families, the particular wavelet used here is commonly referred to as the Daubechies 4-tap wavelet, or db4 [6]. Firstly, the signal is decomposed by simultaneously applying a low pass filter and a high pass filter, which output the approximation and detail coefficients, respectively:

$$y_{low}[n] = \sum_{k=-\infty}^{\infty} x[k]h[2n-k]$$

$$y_{high}[n] = \sum_{k=-\infty}^{\infty} x[k]g[2n-k]$$

Where h and g are the low and high filter's impulse responses, respectively. The output of the filters is then subsampled, because according to Nyquist's rule half of the samples can be discarded since half of the frequency band is removed. Hereafter the process of decomposition and subsampling is repeated in a filter bank. The repetition depth, or level, determines the frequency resolution; higher-level coefficients have a high frequency resolution, and a low time resolution. Preliminary analysis shows that a level 4 DWT provides a suitable representation for classification, as well as the fact that the higher-level coefficients contain no useful information. Therefore, the DWT is truncated at the fifth coefficient level, out of 16 levels in total. For each 2400 audio samples, the mean of the resulting 44 coefficient vectors is taken, to produce a 5-feature length vector for classification. Though this may seem to be a small amount of features, results show that it is sufficient for a classifier to attain a very high accuracy.

3.2.3 Mel Frequency Cepstral Coefficients

The Mel-Frequency Cepstral is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency. MFCCs are commonly used as features in speech and speaker recognition systems, since its warped frequency scale approximates the human auditory system's response more closely than the linearly-spaced frequency bands of the FFT or common cepstral representation. For this study, the following approach was used to produce MFCCs:

1. Compute the FFT for an audio sample.
2. Map the obtained FFT onto the Mel scale, using triangular overlapping mathematical windows, also known as producing a triangular bank filter.
3. Compute the log spectrum. This is done by taking the logs of the powers at each of the Mel frequencies.
4. Finally, compute the discrete cosine transform of the list of Mel log powers. This produces the MFCCs.

[2] http://www.scipy.org/

[3] http://www.pybytes.com/pywavelets/

This method of calculating coefficients allows you to pick the size of the MFCC array as you choose. Result analysis shows that all characteristic components of both the horn and whistle signals can be captured in a 13-feature length instance vectors, which was the design choice for this study.

3.3 Machine Learning Algorithms

The machine learning algorithms used in this study were implemented in the Scikit-Learn Python module [13]. Before classification, all analyzed data was scaled in order to produce data with zero mean and unit variance, which is needed for algorithms like support vector machines. This scaling was first applied to the training set, and its parameters were then used to scale the test set, in the same manner.

The data sets were then shuffled, and split with 70% of the data for the training set, and 30% reserved for cross-validation. These examples can be represented as a set of instance vectors $\vec{x}$, and mark y: $\{(\vec{x}_i, y_i)\}_{i=1}^{M}$, where $\vec{x}_i \in \mathbb{R}^M$ and $y_i \in \{0, 1\}$. The amount of samples, M, greatly exceeds the amount of features N: $M \gg N$.

The results in this paper were obtained after each classification algorithm was run 20 times independently. For each classifier run, the data was re-shuffled and split, after which the classifiers were fitted. Their accuracy, precision, recall, F1-score, false positives, false negatives, true positives, and true negatives were stored and averaged over all of the runs.

3.3.1 Logistic Regression

The most efficient algorithm in terms of computing power used for this study is the logistic regression binary classifier with l^2 normalization [3]. This particular form of regularization was selected since previous approaches to this task yielded promising results [14]. Its hypothesis space consists of sigmoid functions, $h_\theta(\vec{x}) = \frac{1}{1+e^{-\vec{\theta}^T \vec{x}}}$, where $\vec{\theta}$ are the adjustable weights, and $\vec{x}$ the sample vectors. In its implementation, $y_i \in \{-1, 1\}$, yet this is solved by automatically setting all instances initially marked 0 to -1 prior to fitting, and resetting them for the evaluation phase. It solves the following unconstrained optimization problem:

$$min_{\vec{\theta}} \left(\frac{1}{2} \vec{\theta}^T \vec{\theta} + C \sum_{i=1}^{N} \xi(\vec{\theta}; \vec{x}_i, y_i) \right)$$

Where ξ is the loss function:

$$\xi(\vec{\theta}; \vec{x}_i, y_i) = log(1 + e^{-y_i \vec{\theta}^T \vec{x}_i})$$

In order to combat overfitting, but retain a high accuracy rate, a C-sweep (inverse regularization, $C = \frac{N}{\alpha}$) was performed.

3.3.2 Stochastic Gradient Descent

The SGD classifier used here is a first-order learning routine, which, in contrast to batch gradient descent, considers a single instance vector at a time. The change of its weights is given by the gradient of the classification error, which is evaluated only for the latest in the training sample sequence. This variant is sometimes known as the On-line Gradient Descent algorithm (OGD). Its weights vector $\vec{\theta}$ is updated like so:

$$\vec{\theta} \leftarrow \vec{\theta} - \eta \nabla Q_i(\vec{\theta})$$

Where $Q_i(\vec{\theta})$ is the value of the loss function at example i, and the adaptive learning rate η is given by

$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)}$$

Where t is the weight number, $t = 1, 2, \ldots, T$. For the task of audio recognition, the decision was made to compute the gradient over a simple hinge loss function. This linear approach has previously successfully been applied to audio recognition, for instance in music annotation [12].

3.3.3 AdaBoost-Samme

AdaBoost-SAMME [15] is a state-of-the-art generalization of the AdaBoost algorithm [7]. This meta-estimator begins by fitting a classifier, a decision tree classifier in this case, on the original dataset, and then fits additional copies of the classifier on the same dataset, but where the weights of incorrectly classified instances are modified such that subsequent classifiers focus more on difficult cases.

The ‘weak classifier’ used for this algorithm, a decision tree classifier, is first built using a greedy, top-down recursive partitioning strategy. It uses the Gini impurity function to calculate the quality of a split, which is a measure of how often a randomly picked instance would be incorrectly labeled if it were randomly labeled according to the distribution of labels it has already encountered. On its own, this decision tree classifier achieves a very low accuracy, but using AdaBoost with a standard learning rate of 1.0, the accuracy and F1-scores soar.

3.3.4 Support Vector Machine/C-Support Vector Classifier

The support vector machine used in this study is the C-support vector classifier `sklearn.svm.SVC`; an l^2-regularized SVM. In its implementation, $y_i \in \{-1, 1\}$, yet this is solved by automatically setting all instances initially marked 0 to -1 prior to fitting, and resetting them for the evaluation phase. Similar to logistic regression, this support vector machine solves the following problem:

$$min_{\vec{\theta}} \left(\frac{1}{2} \vec{\theta}^T \vec{\theta} + C \sum_{i=1}^{N} \xi(\vec{\theta}; \vec{x}_i, y_i) \right)$$

Yet with a loss function different from logistic regression, ξ, defined as

$$\xi(\vec{\theta}; \vec{x}_i, y_i) = max(1 - y_i \vec{\theta}^T \vec{x}_i, 0)^2$$

For FFT and MFCC, the error term penalty parameter $C = \frac{N}{\alpha}$ was kept at 1.0 after preliminary assessment - lowering C made model convergence unfeasible, while higher values gravely affected F1-scores. For DWT analysis, C was adjusted to 1.4. In the decision function used for FFT and MFCC, SVC uses a sign function with a third degree polynomial kernel K, without an intercept term:

$$h_\theta(\vec{x}) = sgn \left(\sum_{i=1}^{N} y_i \alpha_i K(\vec{x}_i, \vec{x}) \right)$$

Note that for most types of audio analysis, support vector machines will yield very poor results if the data has not been scaled. SVC is an inefficient algorithm for large data sets, since its fit time complexity is more than quadratic. In order to speed up convergence, a shrinking heuristic was used. This shrinking heuristic removes certain elements that may have already been bounded during the decomposition iterations [4]. Because of the optimizations to this algorithm, convergence time was brought down to a feasible level, while retaining a low criterion for stopping tolerance ($tol = 0.001$).

4 Results

All analysis/classifier combinations were run 20 separate times on each combined data set, with a cross validation stage on the remaining 30% of the data that was not used during the training stage. The accuracy score reflects that of a single classifier, run on all audio samples in from all 4 microphone inputs, unless specified otherwise.

The first of experiments was performed on both signal types separately, in order to ascertain any discrepancies in performance. The results can be found in Table 2.

These results show that although classifier performance varies slightly between different types of audio, it is not clear whether this is due to the signal type itself, or to small imperfections in data set 6, since there is only a 0.2% accuracy rate variation.

In order to obtain a clear view of the difference in classifier performance in varying conditions, each classifier/analysis combination was run on both the noisy and clear data sets. These results show something unexpected: None of the analysis/classifier combinations seems to be heavily affected by signal noise. The accuracy rates on combined sets 7 and 8 (see table 3), show only minor variations.

	FFT	DWT	MFCC
SVM	99.97	**100.00**	**100.00**
SGD	99.99	99.99	**100.00**
ADA	99.97	**100.00**	99.97
LogReg	**100.00**	99.98	99.99

	FFT	DWT	MFCC
SVM	99.73	99.80	99.87
SGD	99.70	99.82	**99.90**
ADA	99.87	99.84	99.87
LogReg	99.88	99.83	99.87

Table 2: Accuracy percentage scores on the whistle data set (set 5 - left) and horn data set (set 6 - right).

	FFT	DWT	MFCC
SVM	97.44	94.99	99.60
SGD	98.74	96.25	99.22
ADA	99.69	98.99	99.65
LogReg	**99.78**	96.35	99.21

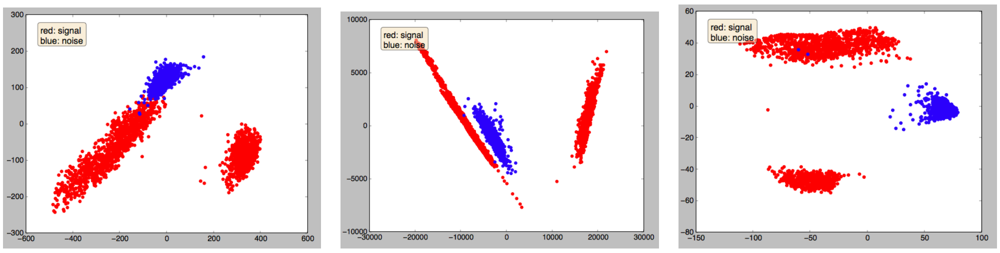
	FFT	DWT	MFCC
SVM	98.66	96.28	99.90
SGD	99.49	96.92	97.88
ADA	**99.88**	**99.88**	99.87
LogReg	99.87	97.88	99.87

Table 3: Accuracy percentage scores on the combined noisy (set 7 - left) and clean data (set 8 - right).

Table 3 shows a performance comparison between the total combined noisy and clean sets. Note that these sets contain multiple types of signals - start, stop, and whistle signals. Their overall performance decreases slightly when applied to noisy data, but only by a small margin. This implies all methods are mostly impervious to background noise.

4.1 Evaluation

The purpose of this experimental study is to assess several potential system proposals for the recognition of trigger-events in the form of audio signals, and to compare their performance in discerning both low-frequency sinusoids and a high-frequency whistle sound. Figures 2a, 2b, and 2c are scatter plots of $\sim 5\%$ of the processed versions of set 9, of which the dimensionality was reduced using Principal Component Analysis (PCA). These PCA representations clearly show that all audio analysis techniques provide a robust method for discerning each data point, since most of the sets are linearly separable.



(a) FFT

(b) DWT

(c) MFCC

Figure 2: PCA visualization of the combined data set 9 for several audio analysis techniques.

5 Conclusion

Twelve different combinations of audio analysis and machine learning techniques were studied for their effectiveness in recognizing three auditory triggers on a NAO robot using four microphones. The audio analysis was performed using Fast Fourier Transform, Discrete Wavelet Transform, and Mel Frequency Cepstral Coefficients, whereas the recognition task was handled by a C-Support Vector Classifier, Stochastic Gradient Descent, AdaBoost-SAMME, and Logistic Regression.

The results reported herein clearly prove that all twelve combinations are good candidates to solve the challenge of recognizing both the predefined horn signal with a sinusoid shape and an arbitrary whistle signal. Experiments were carried out comparing the accuracy rates, in order to discover the optimal settings for each algorithm in both noisy and quiet environments. Performance was not severely affected by noise, and because these results are based on $50ms$ time frames, while a real signal would be at least several frames long, all of these methods possess the ability to recognize natural audio signals reliably.

References

[1] Niels W. Backer. Horn and whistle recognition techniques for nao robots. Bachelor thesis, Universiteit van Amsterdam, June 2014.

[2] Andrea Bonarini, Daniele Lavatelli, and Matteo Matteucci. A composite system for real-time robust whistle recognition. In *RoboCup 2005: Robot Soccer World Cup IX*, volume 4020 of *Lecture Notes in Computer Science*, pages 130–141. Springer Berlin Heidelberg, 2006.

[3] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[4] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27:1–27:27, 2011.

[5] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[6] Ingrid Daubechies. *Ten lectures on wavelets*, volume 61 of *CBMS-NSF Series in Applied Mathematics*. SIAM Publications, Philadelphia, 1992.

[7] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[8] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference onAcoustics, Speech and Signal Processing (ICASSP'13)*, pages 6645–6649, May 2013.

[9] Dinesh Babu Jayagopi, Samira Sheiki, David Klotz, et al. The vernissage corpus: A conversational human-robot-interaction dataset. In *8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 149–150. IEEE Press, 2013.

[10] Ivana Kruijff-Korbayová, Georgios Athanasopoulos, Aryel Beck, et al. An event-based conversational system for the nao robot. In *Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop*, pages 125–132. Springer New York, 2011.

[11] Gil Lopes, Fernando Ribeiro, and Paulo Carvalho. Whistle sound recognition in a noisy environment. In *Proceedings of Controlo'2010 - 9th Portuguese Conference on Automatic Control*, pages 172–179, September 2010.

[12] Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius O Smith. Learning sparse feature representations for music annotation and retrieval. In *Proceedings of 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 565–570, 2012.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] Kyle Poore, Saminda Abeyruwan, Andreas Seekircher, and Ubbo Visser. Single- and multichannel whistle recognition with nao robots. In *RoboCup 2014: Robot Soccer World Cup XVIII*, Lecture Notes on Artificial Intelligence series. Springer, Heidelberg, 2015. to be publicated.

[15] Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2(3):349—360, 2009.