IS THERE A RELATION BETWEEN THE SYNTAX AND THE FITNESS OF AN AUDIO FEATURE?

Gabriele Barbieri **Mirko Degli Esposti**
Dip. di Matematica, Università di Bologna
{gbarbieri,desposti}@dm.unibo.it

François Pachet **Pierre Roy**
Sony CSL Paris
{pachet,roy}@csl.sony.fr

ABSTRACT

Feature generation has been proposed recently to generate feature sets automatically, as opposed to human-designed feature sets. This technique has shown promising results in many areas of supervised classification, in particular in the audio domain. However, feature generation is usually performed blindly, with genetic algorithms. As a result search performance is poor, thereby limiting its practical use. We propose a method to increase the search performance of feature generation systems. We focus on *analytical features*, i.e. features determined by their syntax. Our method consists in first extracting statistical properties of the feature space called *spin patterns*, by analogy with statistical physics. We show that spin patterns carry information about the topology of the feature space. We exploit these spin patterns to guide a simulated annealing algorithm specifically designed for feature generation. We evaluate our approach on three audio classification problems, and show that it increases performance by an order of magnitude. More generally this work is a first step in using tools from statistical physics for the supervised classification of complex audio signals.

1. INTRODUCTION

The identification of feature sets is a fundamental step in solving supervised classification problems [3]. For problems involving complex signals (e.g. music, images, etc.) the traditional approach is to use “off-the-shelf” features (see, e.g. [9]). However, general-purpose features are not always adapted to solve difficult classification tasks. Another solution is to design manually *ad hoc* features, specific to the problem at hand. Such a task can be conducted only by experts knowledgeable both in the domain (e.g. music) and in signal processing, a difficult, costly and time-consuming task. Moreover, there is no guarantee that humans will find the best possible features.
Feature generation has recently been introduced to address this problem, by generating automatically problem-dependent features, designed to be efficient for any particular supervised classification problem [18]. Feature generation consists in building features by searching in a huge feature space, usually through genetic programming techniques [7]. The *fitness* of a feature is defined as the performance of a classifier using this feature on the problem at hand [8]. These previous works have consistently demonstrated that feature generation outperforms traditional approaches based on feature selection (see e.g. [5, 14]). However, most, if not all, feature generation systems proposed in the literature were shown to necessitate the exploration of a large number of features before finding relevant ones. For instance in [17], in the context of classification of percussive sounds, the feature generation system evaluated about 77500 features to eventually find features which outperformed standard ones.
Although *search performance* is usually not an issue for off-line applications, poor search performance forbids the use of these promising techniques in other contexts. As an example, the multiplication of portable entertainment devices creates a need for application requiring fast classification of *a priori* unknown user data (audio, pictures, gestures, etc.). In these contexts, feature generation cannot be used primarily because of performance issues. We propose a search method that reduces substantially the number of features to actually evaluate.
The main source of inefficiency of feature generation comes from the blind search strategy inherent to genetic programming. Most of the computation time is lost in evaluating irrelevant features, as noted by [5]. This problem worsens in the signal domain, where some features can be costly to evaluate (for instance features involving the computation of complex transforms). In this context, search can take several days.
Search performance can be increased by guiding the search using domain specific heuristics. To find heuristics we have to understand the mathematical structure of the feature space, to estimate a priori what regions are likely to contain relevant features. However, this is impossible to do in general, because we do not have information about the semantics of the features [11]. For this reason we restrict our study to the specific case of *analytical features* [14]. Analytical features (AF) are functional compositions of elementary signal processing operators. An AF is defined by its syntactical form, i.e. a tree of basic operators. A central question of our study, reflected in the title of this paper, is therefore how to exploit this syntax to extract information about a features fitness before actually computing it. As

we will see, this relation is complex.
In this paper, we first show that predicting directly the fitness from the syntax is difficult. We propose to model features from a more fine-grained perspective: borrowing techniques from spin glass theory [10] we introduce the notion of *spin patterns* to model partial statistical information about basic operators. We show that spin patterns contain probabilistic information about the fitness of features that use a given operator. We also show how these patterns can be used to predict feature fitness. We then propose a feature generation algorithm guided by these predictions. Our algorithm can be viewed as a variant of the *simulated annealing* [4]. The comparison between simulated annealing and genetic programming is a well studied topic [2], however it is hard to establish what is the more efficient optimization method in the general case [19]. In our context we use simulated annealing because it is easier to guide by the information obtained from the spin patterns.
Two versions of the algorithm are proposed. The basic version searches for individual features, and the extended version searches for feature sets. **Even if it is well known that individual features are not able to solve difficult classification problems [3], we present the basic version because it well describes the theoretical aspects of the algorithm.** We evaluate our algorithms on three audio classification problems. We show that our algorithms find features and feature sets which are as good or better than features found by a standard feature generation algorithm, but with a significantly improved search performance (an order of magnitude).
This paper is structured as follows: In Section 2, we introduce analytical features and syntactic neighborhood. In Section 3, we study the prediction of feature fitness from their syntax. In Section 4, we introduce the notion of spin pattern for operators. We illustrate these patterns on a simple audio classification problem. In Section 5, we introduce our search algorithms. In Section 6, we describe the performance of our algorithm on 3 audio classification problems, and compare it to a standard feature generation algorithm.

2. ANALYTICAL FEATURES

Analytical Features are expressed as a functional term, taking as only argument the input signal (represented here as x). This functional term is composed of basic operators. Given a library of basic operators $\mathcal{L}$, an **analytical feature** f is an application $f = O_1 \circ \ldots \circ O_N$ such that $O_i \in \mathcal{L}$. $\mathcal{S}(f) = \{O_1, \ldots, O_N\}$ is the **syntactical form** of f. The **length** $l(f)$ is the number of operators making up the feature. $\mathcal{F}$ is 0the set of all possible analytical features built on $\mathcal{L}$.
For instance, the following feature (A) computes the MFCC (*Mel Frequency Cepstrum Coefficients*) of successive frames of length 100 (samples) with no (0) overlap, and then computes the variance of this value vector:

$$A = Variance(MFCC(Split(x, 100, 0))).$$

Problem	$I(d_{Fit}, d_{Syn})$
PAN	0.032
INS	0.015
MG	0.015

Table 1. Estimation of the mutual information $I(d_{Fit}, d_{Syn})$ between the distances d_{Fit} and d_{Syn} evaluated on three audio classification problem.

The **neighborhood** of f is the set $V_f = \{g \in \mathcal{F} | d_{Syn}(f, g) \leq 1\}$, where $d_{Syn}(f, g)$ is the Levenshtein distance.

2.1 Feature Fitness

Given a classification problem, the fitness $\lambda_D(f)$ of an AF measures the capacity of the feature to distinguish elements of different classes.
There are several ways to assess the fitness of a feature. We follow here a wrapper approach [6], by which features are evaluated using a classifier built on-the-fly during the feature search process [14]. The fitness of the feature is defined as the performance of a classifier built with this unique feature.
We use Support Vector Machines. To evaluate the performance of the classifier (or more precisely its average F-measure) we use 10-fold cross validation on the training database.

3. PREDICTING FEATURE FITNESS

We define the distance $d_{Fit}(f, g)$ based on the fitness of f and g:

$$d_{Fit}(f, g) = |\lambda_D(f) - \lambda_D(g)|\,.$$

We study here experimentally the relationship between d_{Syn} and d_{Fit} on concrete problems. In this study we consider three audio classification problems. The problem PAN consists in discriminating between six percussive sounds [15], INS consists in discriminating between sounds played by eight different instruments [12] and MG consists in discriminating between six musical genres [13].
We compute a population P of 1000 features randomly generated from $\mathcal{F}$. For each problem and for each couple (f, g) in P, we compute distances $d_{Fit}(f, g)$, $d_{Syn}(f, g)$. Note that d_{Fit} depends on D_i, whereas d_{Syn} is problem-independent.
We then estimate the mutual information $I(d_{Fit}, d_{Syn})$ [1] between the distances. Table 1 shows that in each case the mutual information is smaller than 0.1: if a relation exists between syntax and fitness, it is somehow hidden and difficult to model.

4. SPIN PATTERNS

In this section we introduce a representation of analytical features taking into account the contribution of each sample. Let D be a labeled data set, composed of N audio samples divided in k classes, $C_1, \ldots, C_k$:

$$D = \{(x_1, l_1), \ldots, (x_N, l_N)\}\,,$$

where x_i is the i-th audio sample and $l_i \in \mathcal{C} = \{C_j\}_{j=1,\ldots,k}$ is its label.
Given a feature $f \in \mathcal{F}$, we define its spin configuration as:

$$f \to \sigma^f = \begin{cases} +1 \text{ if } f \text{ classifies } x_i \text{ correctly} \\ -1 \text{ otherwise} \end{cases}$$

If $C : \mathbb{R} \to \mathcal{C} = \{C_j\}_{j=1,\ldots,k}$ is a classifier trained on D with f as a single feature, we have:

$$\sigma_i^f = \begin{cases} +1 \text{ if } C(f(x_i)) = l_i \\ -1 \text{ otherwise} \end{cases}$$

Given $f \in \mathcal{F}$, $\lambda(f)$ is the fitness of f on D, $0 \leq \lambda(f) \leq 1$. By analogy with the spin glass model [10], we can interpret fitness as the Hamiltonian $H(\sigma^f)$ of a spin configuration σ^f induced by the feature f:

$$H(\sigma^f) = -\lambda(f).$$

Given a basic operator $o \in \mathcal{L}$, where $\mathcal{L}$ is the operator library used to construct $\mathcal{F}$, we define

$$\mathcal{F}_o = \{f \in \mathcal{F} | o \in \mathcal{S}(f), l(f) \geq 10\}.$$

So $\mathcal{F}_o$ is the set of all features that contain o with length smaller than 10. Considering only these features we have $0 < |\mathcal{F}_o| < \infty$
The **spin pattern** of a basic operator o is a vector $m(o) = \{m_i(o)\}_{i=1,\ldots,N}$ such that:

$$m_i(o) = \left\langle \sigma_i^f \right\rangle_{\mathcal{F}_o} = \frac{1}{|\mathcal{F}_o|} \sum_{f \in \mathcal{F}_o} \sigma_i^f$$

In practice $m_i(o)$ is an indicator of the probability that a feature containing o correctly classifies sample x_i. Indeed, it is easy to show that, given $f \in \mathcal{F}_o$,

$$Pr(\sigma_i^f = +1) = \frac{1 + m_i(o)}{2}$$

$$Pr(\sigma_i^f = -1) = \frac{1 - m_i(o)}{2}$$

Therefore we are interested in operators whose pattern has as many values as possible which are close to 1. Conversely, *zero spin patterns* configuration ($m_i(o) = 0 \ \forall j$) do not provide any information about the set of features considered. In order to measure the amount of information given by a spin pattern, we can compute its *total magnetization*, defined as follows:

$$M(o) = \frac{\sum_{i=1}^{N} m_i(o)}{N}$$

Figure 1 shows a graphical representation of the spin patterns of two operators. The samples are arranged in a picture composed by N squares. $|m_i(o)|$ is mapped to the darkness of the corresponding square. In this representation, high magnetization features correspond to dark images. In the figure we can observe that the spin pattern of a specific operator like $LpFilter$ has more magnetization than the spin pattern of a more general operator like Abs (the Abs pattern is clearly lighter). The main intuition in guiding search toward an optimal path in the feature space is that there is a rich, non uniform distribution of spin patterns for all basic operators. To support this claim we compute the spin patterns of each operator on the three classification problems presented above. In general, operators yield a spin pattern whose magnetization is significantly higher than 0 (the magnetization of the zero spin pattern, taken as a reference): as we see in figure 2, for each problem, the distributions of the magnetizations for all $o \in \mathcal{L}$ are concentrated near 0.5. This study shows that interesting patterns do exist. Of course we do not know the spin patterns a priori for a given classification problem. **However, we have seen experimentally that we can estimate these patterns using a relatively small number of computations (≈ 1000).**

5. THE ALGORITHM

Our algorithm takes as input a labeled database D and a library of basic operators $\mathcal{L}$. The algorithm searches the feature space $\mathcal{F}$ defined by $\mathcal{L}$, guided by spin patterns, as defined in section 4.

5.1 Individual feature search (IFS)

We describe here the IFS algorithm, that searches for individual features. Section 5.2 describes an extension to feature set search. IFS receives as input a labeled database D and a library of basic operators $\mathcal{L}$. The output is a feature with a high fitness on domain D.
The algorithm is a variant of the simulated annealing algorithm. It is based on a Metropolis procedure [4] that guarantees the convergence to a global optimum.
Starting from a random feature f_0, the algorithm iteratively selects neighbors of the current feature, using the spin patterns. At each iteration, the algorithm selects a new feature in the syntactical neighborhood of the current feature. This choice is done according to the estimation of the spin patterns. The algorithm terminates after a specified number of iterations, or as a result of an interactive user request, and returns the best feature (i.e. the feature with highest fitness) found during the search as output.
In the following subsections we detail the components of the algorithm.

5.1.1 The Spin Pattern Estimator

The *spin pattern estimator* (SPE) is executed only once at the beginning of the algorithm.
The SPE computes a population T of 1000 random features. These features are used to estimate the spin patterns of each operator $o \in \mathcal{L}$.

5.1.2 The Neighbor Selector

The task of the *neighbor selector* is to decide how to move in the feature space. The selector receives as input the cur-

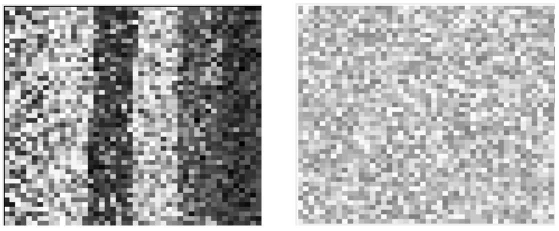
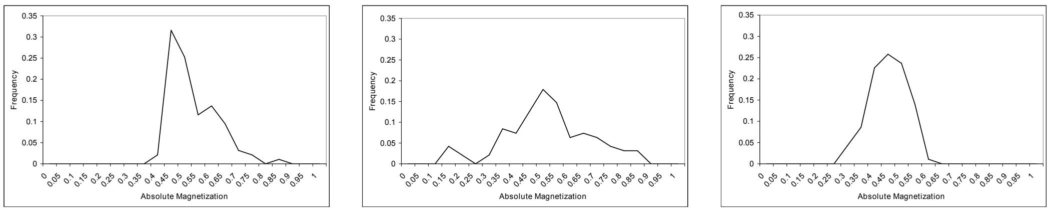

(a) LpFilter (b) Abs

Figure 1. Graphical representation of the spin patterns of the basic operators $Lpfilter$ (left) and Abs (right) evaluated on the problem PAN (classification of percussive sounds). The representation of a domain specific operator like $LpFilter$ is clearly darker. In the spin pattern of $LpFilter$ note the two dark stripes on the center and on the right. Magnetization are 0.46 and 0.28 (resp.)



(a) INS (b) MG (c) PAN

Figure 2. Distributions of the absolute magnetizations of the spin patterns of all the operators $o \in \mathcal{L}$

rent feature and the spin pattern estimation of each operator $o \in \mathcal{L}$. The output of the selector is either a new feature, or the feature passed in input.
Given the input feature f and the estimations of the spin patterns, the selector assigns a score to each basic operator $o \in \mathcal{L}$. This score is designed to favor operators that tend to correctly classify the samples that are wrongly classified by f. The score $\mu(o)$ of operator o is defined as:

$$\mu(o) = \frac{\sum_{j=1}^{N} b(\sigma_i^f)\bar{m}_i(o)}{N} \tag{1}$$

where $b(x) = -\frac{1}{2}x + \frac{1}{2}$ is a function that converts a ± 1 spin in a $\{0, 1\}$ boolean value ($b(+1) = 0$, $b(-1) = 1$). $\mu(o)$ induces a score function $\mu(g)$ for a feature $g \in \mathcal{F}$ by averaging on all operators of f:

$$\mu(g) = \frac{\sum_{o \in \mathcal{S}(g)} \mu(o)}{l(g)} \tag{2}$$

A crucial point of our method is that in general the score function $\mu(g)$ is easier to compute than the fitness $\lambda(g)$.
The selector then computes the neighborhood V_f of f (see section 3). V_f is then sorted according to the score function defined in (2). The feature $\tilde{f} \in V_f$ with highest score is chosen by the neighbor selector. The spin configuration $\sigma^{\tilde{f}}$ and fitness $\lambda(\tilde{f})$ of $\tilde{f}$ are then computed.
To avoid local maximum effects, $\tilde{f}$ is accepted as the next feature in a stochastic way, using the Metropolis procedure.
More precisely, if $\Delta\lambda = \lambda(\tilde{f}) - \lambda(f) \geq 0$ f is accepted. The case $\Delta\lambda < 0$ is accepted with a probability $Pr(\Delta\lambda) = e^{\frac{\Delta\lambda}{t_k}}$, where

$$t_n = \left(\frac{t_1}{t_0}\right)^n t_0$$

and $k = |T|$.
If the feature $\tilde{f}$ is not accepted, it is removed from V_f, otherwise it will be reselected by the neighbor selector in the next iteration. Using the Metropolis procedure, we can assume that, for a good choice of parameters T_0 and T_1, the algorithm converges to the global maximum [4]. Following [4], we heuristically set $T_1/T_0 = 0.95$ and $T_0 = 10$.

5.2 The Feature Set Version

As described earlier, a single feature is usually not enough to solve a classification problem and a feature set is required. In principle, building feature sets instead of individual features complexifies drastically the procedure, since all combinations of features must be considered at each step. **For this reason we propose the follow simple extension of IFS that searches feature sets, in order to maintain an affordable computational cost.**
The architecture of the algorithm is essentially the same of the basic version. The *spin pattern estimator* works exactly in the same way. The only module that changes is the *neighbor selector*. In this version it takes as input a feature set of dimension d (d is a fixed parameter) and outputs a feature set instead of a single feature.
Because the spin configuration of a feature f is defined by the classifier built with the values of f, it is possible to define in the same way the spin configuration σ^F and the fitness $\lambda(F)$ of a feature set F, using the classifier produced by the values of F. Therefore we can define the score of an operator $\mu(o)$ and the score of a feature $\mu(g)$ as in the

previous subsection.
In this version of the algorithm, a feature f is randomly chosen $\in F$. A feature $\tilde{f}$ in the neighborhood V_f is selected like in the basic version of the algorithm.
The new feature set $\tilde{F}$ is then built from F by substituting f by $\tilde{f}$. The new feature set is accepted with the same Metropolis procedure described above. Our algorithm is then able to search for feature sets of any dimension.
In the next section we compare the search performance of our two algorithms against the search performance of a standard genetic algorithm.

6. RESULTS

6.1 Individual Feature Search

To assess the search performance of IFS we compare it against the search performance of a standard genetic algorithm on three audio classification problems. These problems are the same presented in the section 3. The genetic algorithm we use is described in [14].
Given a classification problem D, we use our algorithm and the genetic algorithm to search the feature f with highest fitness $\lambda_D(f)$ and we compare the sets of features explored by the two algorithms. We are interested in three quantities: the fitness of the best feature found, how many features are computed before finding the best feature and the distribution of the fitness of the explored features.
To get statistically significant results, we execute both algorithms three times for each classification problem. The results are shown in figure 3 and figure 4. In figure 3 we can observe that IFS finds features at least as good as the features found by the genetic algorithm but converges faster to the solution. In average, it computes less than 3500 features to find the best features. This figure includes the number of features needed to estimate the spin patterns (approximately 1000, as described in section 4). Conversely the genetic algorithm requires more than 48000 features to find the optimal.
As we can see in figure 4 the distribution of the fitness, when using IFS, is concentrated near high values of the fitness. Conversely the genetic algorithm explores blindly the feature space, resulting in a more uniform distribution of the fitness.

6.2 Feature Set Search

The feature set version of our algorithm builds feature sets of dimension N. Again we test this algorithm against the genetic algorithm used above to find feature sets of dimension 3 for the three reference problems.
The genetic algorithm, after a population has been created and each feature has been individually evaluated, selects a subset of features to be retained for the next population. The output of the genetic algorithm is a feature set F_{GA} of dimension 3: F_{GA} is the set $\{f_1, f_2, f_3 | f_1, f_2, f_3 \in F_{last}\}$ with maximum fitness. F_{last} is the last population.
Here we compare the two feature sets obtained by the two algorithms against an other feature set obtained by applying a feature selection algorithm (in our case, InfoGain

Database	GP	FS	IFS
PAN	0.76	0.78	0.79
INS	0.56	0.56	0.59
MG	0.77	0.79	0.77

Table 2. Comparison between the fitness of the best feature sets obtained by IFS against the best feature sets obtained by the genetic algorithm. In two cases IFS outperforms the genetic one. GP means genetic programming, FS means feature selection.

Database	GP and FS	IFS
PAN	77531	4043
INS	72837	4152
MG	43265	7221

Table 3. Number of features needed to find the best feature sets.

[16]) to the whole set of features explored by the genetic algorithm. In table 2 we observe that our algorithm performs as well as the genetic one and the feature selection one on the three problems. However, the features sets necessitate a smaller exploration: table 3 shows the number of features explored in order to find the best feature set. It can be seen again that our algorithm improves the search performance by an order of magnitude.

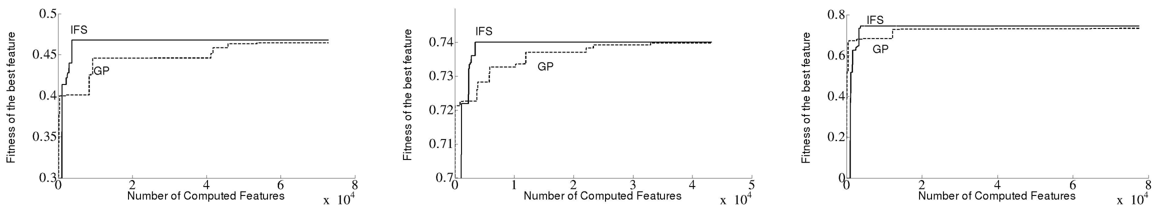
7. CONCLUSION

We have explored the relation between the syntax and the fitness in a supervised classification context. Such a relation seems complex to grasp at a macroscopic level. We have proposed a sample-based approach to model the topology of feature spaces, and exhibited a computable criterion, spin patterns, to guide a feature search algorithm. This algorithm is based on simulated annealing, with a Metropolis procedure, and exploits spin patterns, resulting in a better performance than genetic programming, as measured by the total number of features actually evaluated.
As such, this approach is a promising one to reduce the traditionally high computational cost of feature generation, and increase the applications of this technique. The answer to our title question is therefore: "Yes, there is a complex relationship". Furthermore, we showed how this relationship can be exploited to improve the performance of a feature generation system. More generally, this work represents a first step in applying tools from statistical mechanics of complex systems to supervised classification of complex audio signals.

8. REFERENCES

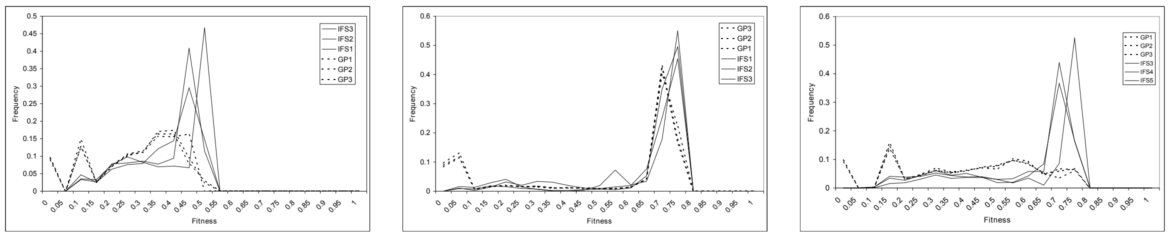
[1] T.M. Cover, J.A. Thomas, *Elements of information theory.* Wiley, New York, 1991.

[2] L. Davis, *Genetic algorithms and Simulated Annealing.* Morgan Kaufman Publishers, Inc.,Los Altos, CA. 1987.



(a) INS (b) MG (c) PAN

Figure 3. Performance of IFS compared to the performance of a standard genetic algorithm (GP). We report here only the best execution for each algorithm.



(a) INS (b) MG (c) PAN

Figure 4. Distributions of the fitness of the features computed by the two algorithms. In dotted line, the distributions of the features computed by the genetic algorithm, in black the features computed by IFS. For IFS, the distributions of the fitness are concentrated near high values, whereas the features explored by the genetic algorithm have more spread distributions.

[3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research,* 3:11571182, 2003.

[4] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, New Series, Vol. 220, No. 4598, pp. 671-680, May 13, 1983.

[5] Y. Kobayashi, "Automatic Generation of Musical Instrument Detector by Using Evolutionary Learning Method", in *Proc. of the 10th International Conference on Music Information Retrieval (ISMIR '09)*, Kobe, Japan, October 2009.

[6] R. Kohavi, G.H. John, "The Wrapper Approach", in *Feature Selection for Knowledge Discovery and Data Mining,* H. Liu & H. Motoda (eds.), Kluwer Academic Publishers, pp 33-50, 1998.

[7] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press. 1992.

[8] S. Markovitch and D. Rosenstein, "Feature Generation Using General Constructor Functions", *Machine Learning*, Vol. 49, pp. 59-98, 2002.

[9] M.F. McKinney, J. and Breebart, "Features for audio and music classification", in *Proc. of the International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 151-158, 2003

[10] M. Mezard, G. Parisi, M.A. Virasoro, *Spin Glasses Theory and Beyond*, World Scientific, Singapore, 1987.

[11] I. Mierswa, and K. Morik, "Automatic feature extraction for classifying audio data", *Machine Learning Journal*, Vol. 58, pp. 127-149, 2005.

[12] University of Iowa Musical Instrument Samples: http://theremin.music.uiowa.edu/MIS.html

[13] http://ismir2004.ismir.net/genre_contest/index.htm

[14] F. Pachet, P. Roy, "Analytical Features: A Knowledge-Based Approach to Audio Feature Generation", *Eurasip Journal on Audio, Speech, and Music Processing*, 2009(1), February 2009.

[15] http://www.csl.sony.fr/pandeiro/

[16] J. R. Quinlan,*C4.5: Programs for Machine Learning,* Morgan Kaufmann, San Francisco, California, USA, 1993.

[17] P. Roy, F. Pachet, S. Krakowski, "Analytical Features for the classification of percussive sound: the case of Pandeiro", in *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR '07)*, pp. 229-232, Vienna, Austria, September 2007.

[18] E. K. Tang ; P. N. Suganthan; X. Yao, "Nonlinear feature extraction using evolutionary algorithm" in *Lecture notes in computer science,* Vol. 3316, pp. 1014-1019, 2004.

[19] A. Vasan, K.S. Raju, "Comparative analysis of Simulated Annealing, Simulated Quenching and Genetic Algorithms for optimal reservoir operation," in *Applied Soft Computing,* Vol. 9, No. 1, pp. 274-281, January 2009.