POLYPHONIC MUSIC CLASSIFICATION ON SYMBOLIC DATA USING DISSIMILARITY FUNCTIONS

Yoko Anan, Kohei Hatano, Hideo Bannai, Masayuki Takeda
Department of Informatics, Kyushu University
{yoko.anan, hatano, bannai, takeda}@inf.kyushu-u.ac.jp

Ken Satoh
National Institute of Informatics
ksatoh@nii.ac.jp

ABSTRACT

This paper addresses the polyphonic music classification problem on symbolic data. A new method is proposed which converts music pieces into binary chroma vector sequences and then classifies them by applying the dissimilarity-based classification method TWIST proposed in our previous work. One advantage of using TWIST is that it works with *any* dissimilarity measure. Computational experiments show that the proposed method drastically outperforms SVM and k-NN, the state-of-the-art classification methods.

1. INTRODUCTION

Classification of music is one of the most fundamental problems in music information retrieval research and has been studied extensively (e.g., [4, 5, 7, 19, 20, 25, 28]). Music is usually polyphonic in the sense that more than one tone sounds simultaneously and thus a single time interval is made up of two or more simultaneous tones. Classifying polyphonic music pieces seems to be more difficult than classifying monophonic music pieces.

The difficulty in classifying polyphonic music stems from two issues. The first issue is in determining what kind of information needs to be extracted from polyphonic music. Many previous research (e.g., [13, 18]) reduce the classification problem of polyphonic music to that of monophonic music by converting the data in some way. For example, the so-called skyline method converts polyphonic music to monophonic music by choosing the highest pitches among multiple pitches. This approach is effective to an extent, but it does not fully exploit the information which can be obtained from multiple pitches.

The second issue is how to classify the preprocessed data. A major approach of machine learning techniques is to represent data as feature vectors and then applying learning algorithms. There are several known features such as performance worm [9], performance alphabet [26] and others [4, 19, 20, 28]. Then, it is non-trivial to construct effective features from data, since such construction requires much human resource such as experts' knowledge. If we employ kernel-based machine learning approaches such as SVM (e.g., [15–17, 21, 29]), we can avoid the problem of explicitly constructing features by using kernels since kernels implicitly define features. However, the kernel-based has a limitation that the kernel must be positive-semidefinite. Therefore many popular (dis)similarity measures such as the edit distance is not applicable since they are not proved to be positive-semidefinite.

In this paper, we propose a new method for classifying polyphonic music, which is a combination of polyphonic music preprocessing and classification techniques. For preprocessing data, our method employs chroma vector representation which is popular for audio data (e.g., [3]). Unlike previous approaches such as the skyline method, we preprocess the data so that information contained in the original data is kept as much as possible. An advantage of the chosen approach is that it captures concurrent behavior of pitches by encoding them into a new set of strings, and therefore, can extract more information from polyphonic music data than the monophonic music reduction approach.

For classification, we propose a multi-class version of our classification method named TWIST (Tug-of-War between Instances by Soft margin optimization Technique) proposed in [2]. TWIST is based on the theory of Wang et al. [31] for learning with (dis)similarity functions. The theory guarantees that under some mild assumptions, the final classifier constructed from dissimilarity functions is accurate enough for future data. Further, TWIST can use any dissimilarity function which might not be positive semi-definite.

By combining the two approaches described above, we significantly outperform the state-of-the-art methods such as k-nearest neighbor (k-NN) and SVM with string kernels for composer classification tasks of classical piano music and Japanese POP music given in MIDI format.

2. OUTLINE OF OUR METHOD

In this section, we explain the outline of our method. Our method consists of two parts: (i) First of all, we convert polyphonic music data into binary chroma sequences. In our experiments, the original polyphonic data is given as MIDI data. (ii) Next, given labeled binary chroma sequences and a dissimilarity function which measures the discrepancy between them, we use a multi-class version of TWIST to learn a classifier. The details are given in Section 3.

3. QUANTIFYING DISSIMILARITY BETWEEN MUSIC PIECES

The key to successfully using TWIST is in choosing how to quantify dissimilarity between music pieces. We consider dissimilarity measures d that are combinations of a preprocessing p of music pieces into particular sequential representation, and a string dissimilarity measure δ. That is, d is given by $d(x, y) = \delta(p(x), p(y))$, where x, y are music pieces.

One popular sequential representation of polyphonic music is the chroma vector sequence representation. A chroma vector is a twelve-element vector with each dimension representing the intensity in a very short time interval, associated with a particular semitone regardless of octave. Chroma vectors model important aspect of music audio and have been widely used in music retrieval [22, 23], music classification [1, 10], and several other applications in music information processing [24].

On the other hand, many string (dis)similarity measures have been proposed, such as the edit distance [30], the longest common subsequence (LCS) length [14], the normalized compression distance (NCD) [6], which are used in many applications such as automatic spelling correction, information retrieval, gene information analysis and so on. String kernels such as the n-gram kernel [16], the mismatch kernel [15], the subsequence kernel [21] are also string similarity measures.

Consider applying such a string similarity measure to music pieces that are given in the form of sequences of binary chroma vectors (or pitch sets). A naive approach would be to use the so-called *skyline method*, where the highest pitch is chosen among multiple pitches in each time interval. It is essentially a reduction to monophonic music processing.

Another approach is a direct computation of dissimilarity regarding the binary chroma vectors as just symbols. There are $2^{12} = 4096$ symbols. For the edit distance, we have to define the weights associated with edit operations, namely, the weight $w(a, \varepsilon)$ of deleting a and the weight $w(\varepsilon, a)$ of inserting a for any symbol a, and the weight $w(a, b)$ of replacing a with b for any distinct symbol pair (a, b). The simplest way is to use the unit weight function such that $w(a, \varepsilon) = w(\varepsilon, a) = 1$ for any symbol a and $w(a, b) = 1$ for any distinct symbol pair (a, b) [1]. Another possible way would be to set $w(a, b) = 1 - \theta(a, b)$ where $\theta(a, b)$ is the angle between the vectors a and b for any distinct symbol pair (a, b), and $w(a, \varepsilon) = w(\varepsilon, a) = 1$ for any symbol a. An alternative way is to quantify resemblance between chroma vectors based on musical knowledge. Harte et al. [12] proposed such a method: It converts 12-dim. binary chroma vectors into 6-dim. real-valued vectors, called the tonal centroid vectors (TC vectors in short). They claim in [12] that close harmonic relations such as fifths and thirds appear as small Euclidian distances. Thus the Euclidian distance of two TC vectors or the cosine of the angle between them could be a good dissimilarity (similarity) measure between original chroma vectors.

Using TC vector conversion Ahonen et al. [1] took another approach. The component values of TC vectors are quantized to 0 or 1 to produce 6-dim. binary vectors, which we call the *binary TC vectors*. The resulting sequences are thus strings over an alphabet of size 64, and NCD with `bzip2/PPMZ` is used to quantify their dissimilarity.

In Section 5 we compare by computational experiments the performance of all combinations of sequential representation and (dis)similarity measure mentioned above.

4. CLASSIFICATION METHOD

In this section, we briefly sketch TWIST [2] which is designed for binary classification. Then we show how to extend TWIST for multi-class classification tasks.

4.1 binary classification

TWIST employs a dissimilarity-based learning framework of [31], which we call TW (Tug-of-War). We first explain the TW framework below.

Let X be the instance space. We call a pair (x, y) of instance $x \in \mathrm{X}$ and label $y \in \{-1, 1\}$ an *example*. Suppose that we are given p positive examples $(x_1^+, +1) \ldots,$ $(x_p^+, +1)$ and n negative examples $(x_1^-, -1), \ldots, (x_n^-, -1)$. We are also given a dissimilarity function $d(x, x')$ is a function from $\mathrm{X} \times \mathrm{X}$ to $\mathbb{R}^+$.

For each pair of positive instance x_i^+ and negative instance x_j^- $(i = 1, \ldots, p, j = 1, \ldots, n)$, we define the base classifier $h_{i,j} : X \to \{-1, +1\}$ as follows:

$$h_{ij}(x) = \mathrm{sgn}(d(x_j^-, x) - d(x_i^+, x)),$$

where $\mathrm{sgn}(a) = 1$ if $a > 0$ and -1 otherwise. The base classifier h_{ij} classifies an instance x as positive if x is more dissimilar to the negative instance x_j^- than the positive instance x_i^+ (in other words, x is more similar to x_i^+ than x_j^-) and it classifies an instance x as negative, otherwise. The behavior of the base classifier seems like a tug-of-war, which is why we call the framework TW. In the TW framework, the final classifier is a weighted voting of base classifiers,

$$\mathrm{sgn}[\sum_{i=1}^{p} \sum_{j=1}^{n} w_{ij} h_{ij}(x)]$$

for some weights w_{ij}s. TW has a theoretical guarantee that, under some natural assumptions, there exist weights such that the associated final classifier is accurate enough for future instances. A heuristics is used to determine weights in the original paper [31].

Our previous work [2] uses a more robust method for finding weights than the above mentioned heuristics. We named the method of [2] TWIST, which is an abbreviation of "Tug-of-War of Instances by Soft margin optimization Technique". TWIST employs the 1-norm soft margin optimization to determine weights w_{ij}s. The 1-norm soft margin optimization is a standard formulation of classification

[1] The edit distance with this weight function is often called the Levenshtein distance.

problems in Machine Learning (see, e.g., [8, 32]), which is known to provide a robust classifier. In our case, the 1-norm soft margin optimization is formulated as follows:

$$\max_{\rho,b,\boldsymbol{w},\boldsymbol{\xi}^+,\boldsymbol{\xi}^-} \rho - \frac{1}{\nu}\sum_{k=1}^{p}\xi_k^+ - \frac{1}{\nu}\sum_{k=1}^{n}\xi_k^- \tag{1}$$

sub.to

$$\sum_{i=1}^{p}\sum_{j=1}^{n} w_{ij}h_{ij}(x_k^+) + b) \geq \rho - \xi_k^+ (k = 1, \ldots, p),$$

$$-\sum_{i=1}^{p}\sum_{j=1}^{n} w_{ij}h_{ij}(x_k^-) + b) \geq \rho - \xi_k^- (k = 1, \ldots, n),$$

$$\boldsymbol{w} \geq \boldsymbol{0},\ \sum_{i=1}^{p}\sum_{j=1}^{n} w_{ij} = 1,\ \boldsymbol{\xi}^+,\ \boldsymbol{\xi}^- \geq \boldsymbol{0},$$

where each y_k is $+1$ or -1. An additional advantage of 1-norm soft margin optimization is that the resulting weights are likely to be sparse since we regularize 1-norm of the weights This property is also useful for feature selection tasks.

4.2 multi-class classification

We explain how to extend TWIST for multi-class classification. We employ the standard reduction method from multi-class to binary classification, one-versus-rest. The one-versus-rest method solves K-class classification by reducing it to K binary classification problems. For each class k ($1 \leq k \leq K$), the associated binary classification problem is constructed by assuming the label k is positive and other labels are negative. Then a learning algorithm is applied for each binary classification problem and it outputs the classifier $h_k : X \to \mathbb{R}$ for each class k. The final classifier of the one-versus-rest method is given as

$$\arg\max_{k=1,\ldots,K} h_k(x).$$

5. COMPUTATIONAL EXPERIMENT

We evaluated the performance of TWIST in composer classification of music pieces in comparison to those of the classification methods k-NN and SVM. A suitable dataset should contain enough number of music pieces for each composer, and the pieces for all composers have roughly the same conditions (the length, genre, instrument, etc. of the piece). Although various MIDI datasets are publicly available, datasets suitable for composer classification are rare. For example, the classical music dataset of the RWC Music Database [2] consists of 50 pieces written by 24 composers, only 2.08 pieces for each composer on the average.

The following datasets were available for our experiments:

Classical. The set of classical music MIDI files described in [27]. It consists of 5 sets of 25 pieces of keyboard music, written by Bach, Beethoven, Chopin, Mozart and Schumann, respectively.

JPOP. A set of Japanese POP (JPOP) music MIDI files for KARAOKE downloaded from a commercial site by YAMAHA. It consists of 5 sets of 25 pieces of JPOP, written by 5 composers (Tomoyasu Hotei, Tetsuya Komuro, Keisuke Kuwata, Takahiro Matsumoto, Kazumasa Oda).

From the MIDI files, we removed the MIDI events other than the NOTE ON/OFF events and quantized the NOTE ON/OFF times with unit time corresponding to the sixteenth note length. The tracks/channels of the MIDI files can then be viewed as sequences of sets of pitches that are "ON" in respective unit time intervals. Each MIDI file in **Classical** consists of two tracks, corresponding to the left and right hand parts. We extracted two pitch-set sequences and merged them into a single sequence. Each MIDI file in **JPOP** consists of a single track with several channels. We chose the channel 0 corresponding to main melody part and obtained a single pitch-set sequence from the channel. We then converted the obtained pitch-set sequences into (a) highest-pitch strings, (b) binary chroma vector sequences, and (c) binary TC vector sequences.

For quantifying dissimilarities between sequences, we adopted several (dis)similarity measures between strings. For TWIST, SVM and k-NN, we used the following two string kernels: n-gram kernel with parameters $n = 2, 5, 10$ and mismatch kernel with parameters $n = 5, 10$ and $m = 1, 2$, where m is the maximum number of errors allowed. For TWIST and k-NN, we also used the following (dis)similarity measures: edit distance, LCS, and NCD with compression programs `gzip` and `bzip2`.

For the edit distance between binary chroma vector sequences, we used the symbol-pair weight functions w of the three types: (i) the unit weight ($w(a, b) = 1$ if $a \neq b$ and $w(a, b) = 0$ if $a = b$); (ii) $w(a, b) = 1 - \cos\theta(a, b)$ for binary chroma vectors a, b; and (iii) $w(a, b) = 1 - \cos\theta(a', b')$ for TC vectors a', b' of binary chroma vectors a, b. For highest-pitch strings and binary TC vectors, we used only the unit weight. We used the cosine values for (ii) and (iii) in the case of LCS. We note that the compression programs `gzip` and `bzip2` used in NCD take data files of byte-sequences as input. We encoded the highest-pitch strings as one-byte-integer sequences, wrote them into data files and then applied the compressors to the files. For binary chroma vector sequences, we wrote them as data files of two-byte-integer sequences to be processed in a byte-wise manner by the compressors.

We evaluated the three classification methods by performing 5-fold cross validation. We used the values 0.05, 0.1, 0.2, 0.3, 0.4, 0.5 for the parameter ν of the 1-norm soft margin optimization formulation (1). For SVM, we used the ν-SVM implementation of LIBSVM (version 3.11) [11]. The values 0.05, 0.1, 0.2, 0.3, 0.4, 0.5 for the parameter ν were used. For k-NN, we used $k = 1, 3, 5$. Accuracies are obtained using the best value of ν for each method and each (dis) similarity measure.

[2] http://staff.aist.go.jp/m.goto/RWC-MDB/

Table 1. Comparison of classification accuracy for dataset **Classical** (in %).

(a) highest-pitch strings.

	edit distance	LCS	NCD		n-gram kernel			mismatch kernel			
	unit weight	unit weight	`bzip2`	`gzip`	$n=2$	$n=5$	$n=10$	$n=5$		$n=10$	
								$m=1$	$m=2$	$m=1$	$m=2$
TWIST	70.40	78.40	80.80	**84.80**	76.00	77.60	72.80	72.80	81.60	79.20	75.20
1-NN	52.80	20.00	51.20	41.60	16.00	20.00	12.00	19.20	18.40	12.00	13.60
3-NN	60.00	18.40	62.40	48.00	17.60	41.60	44.80	28.00	19.20	46.40	39.20
5-NN	51.20	24.80	56.00	39.20	33.60	27.20	32.00	36.00	34.40	31.20	30.40
SVM	N/A	N/A	N/A	N/A	51.20	44.00	24.80	52.00	52.80	26.40	29.60

(b) binary chroma vector sequences.

	edit distance			LCS			NCD		n-gram kernel			mismatch kernel			
	weight			weight			`bzip2`	`gzip`	$n=2$	$n=5$	$n=10$	$n=5$		$n=10$	
	unit	cosine	TC	unit	cosine	TC						$m=1$	$m=2$	$m=1$	$m=2$
TWIST	80.00	70.40	72.00	81.60	77.60	77.60	91.20	**92.00**	80.00	70.40	49.60	74.40	73.60	61.60	67.20
1-NN	46.40	50.40	44.80	29.60	30.40	27.20	63.20	53.60	20.00	22.40	24.00	20.80	21.60	18.40	17.60
3-NN	35.20	43.20	38.40	25.60	28.80	24.00	68.00	63.20	16.80	4.80	8.00	10.40	16.80	4.80	4.80
5-NN	25.60	40.00	37.60	23.20	23.20	21.60	65.60	56.00	25.60	1.60	9.60	3.20	11.20	4.80	4.00
SVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	57.60	40.00	24.80	44.80	51.20	24.00	23.20

(c) binary TC vector sequences.

	edit distance	LCS	NCD		n-gram kernel			mismatch kernel			
	unit weight	unit weight	`bzip2`	`gzip`	$n=2$	$n=5$	$n=10$	$n=5$		$n=10$	
								$m=1$	$m=2$	$m=1$	$m=2$
TWIST	76.00	78.40	85.60	**88.80**	78.40	70.40	59.20	72.00	73.60	64.80	67.20
1-NN	42.40	28.00	56.00	59.20	20.00	20.00	17.60	19.20	19.20	16.00	17.60
3-NN	32.00	25.60	64.80	65.60	18.40	18.40	17.60	18.40	18.40	15.20	16.00
5-NN	28.00	25.60	57.60	52.00	22.40	1.60	6.40	3.20	12.00	1.60	0.80
SVM	N/A	N/A	N/A	N/A	58.40	46.40	25.60	52.00	54.40	25.60	29.60

The experimental results for **Classical** are summarized in Table 1. TWIST drastically outperforms the other classification methods in all combinations of sequential representation and (dis)similarity measure. TWIST shows the best accuracy when used with combination of the binary chroma vector sequence representation and NCD with `gzip`.

The experimental results for **JPOP** are summarized in Table 2. Again, TWIST defeats the other classification methods in most combinations of sequential representations and (dis)similarity measures. This time TWIST using NCD with `bzip2` shows good accuracies. In fact the best and the second best are achieved by NCD with `bzip2` combined with the highest-pitch strings and with the binary chroma vector sequence representations, respectively.

Now we discuss the effects of preprocessing for classification of **JPOP** and **Classical**. For **JPOP**, the classification accuracy with highest-pitch strings is better than that with chroma vector sequences. This might be due to the fact that **JPOP** is almost like monophonic music. More precisely, each music of **JPOP** is characterized with highest-pitch sequence mostly corresponding to the lead vocal line. On the other hand, each piano music of **Classical** is characterized with a succession of simultaneously sounding pitches. Therefore, chroma vector representation is more advantageous for classification of polyphonic music since the representation keeps more information in the original music.

6. CONCLUSION

In this paper we proposed a polyphonic music classification method as a combination of way of quantifying affinity between music pieces and the classification technique TWIST [2]. The method converts given music data into binary chroma vector sequences, and builds a classifier based on the similarity values between the sequences (or their converted sequences) using a string similarity measure. One advantage is that TWIST works with *any* similarity measure, not necessarily to be positive semidefinite. The results of computational experiments with classical music and Japanese POP music show that TWIST drastically outperforms the well-known classification methods k-NN and SVM with string kernels in all combinations of sequential representation and similarity measure.

Although the computational experiments were carried out on MIDI files, our classification method can, in theory, be applied to audio files, provided that an appropriate function that quantifies affinity between music audio data

Table 2. Comparison of classification accuracy for dataset **JPOP** (in %).

(a) highest-pitch strings.

	edit distance	LCS	NCD		n-gram kernel			mismatch kernel			
	unit weight	unit weight	`bzip2`	`gzip`	$n=2$	$n=5$	$n=10$	$n=5$		$n=10$	
								$m=1$	$m=2$	$m=1$	$m=2$
TWIST	78.40	80.80	**86.40**	73.60	38.40	31.20	39.20	28.80	48.80	26.40	28.00
1-NN	26.40	21.60	33.60	27.20	19.20	19.20	19.20	19.20	19.20	19.20	19.20
3-NN	37.60	45.60	58.40	44.00	58.40	58.40	20.00	20.00	58.40	58.40	20.00
5-NN	34.40	42.40	43.20	40.00	20.00	20.00	20.00	20.00	20.00	20.00	20.00
SVM	N/A	N/A	N/A	N/A	35.20	25.60	17.60	25.60	36.00	17.60	19.20

(b) binary chroma vector sequences.

	edit distance			LCS			NCD		n-gram kernel			mismatch kernel			
	weight			weight			`bzip2`	`gzip`	$n=2$	$n=5$	$n=10$	$n=5$		$n=10$	
	unit	cosine	TC	unit	cosine	TC						$m=1$	$m=2$	$m=1$	$m=2$
TWIST	80.00	83.20	83.20	76.80	81.60	81.60	**84.80**	79.20	60.80	48.80	28.80	44.00	50.40	36.00	34.40
1-NN	31.20	30.40	34.40	27.20	27.20	30.40	35.20	30.40	19.20	19.20	19.20	19.20	19.20	19.20	19.20
3-NN	44.80	44.80	49.60	53.60	53.60	55.20	51.20	44.80	57.60	56.80	58.40	58.40	49.60	57.60	58.40
5-NN	41.60	41.60	46.40	46.40	46.40	32.80	48.80	40.00	19.20	20.00	20.00	20.00	27.20	20.00	20.00
SVM	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	40.80	40.00	32.00	40.80	40.80	33.60	36.00

(c) binary TC vector sequences.

	edit distance	LCS	NCD		n-gram kernel			mismatch kernel			
	unit weight	unit weight	`bzip2`	`gzip`	$n=2$	$n=5$	$n=10$	$n=5$		$n=10$	
								$m=1$	$m=2$	$m=1$	$m=2$
TWIST	72.80	**83.20**	79.20	80.80	56.00	38.40	40.00	45.60	42.40	28.80	37.60
1-NN	28.80	28.00	31.20	30.40	19.20	19.20	19.20	19.20	19.20	19.20	19.20
3-NN	44.00	55.20	48.80	55.20	57.60	56.00	58.40	58.40	42.40	57.60	58.40
5-NN	44.00	48.00	45.60	40.00	20.00	21.60	20.00	20.00	27.20	20.00	20.00
SVM	N/A	N/A	N/A	N/A	38.40	39.20	32.00	41.60	43.20	33.60	36.00

is available. A future work is to develop such a function for music audio data.

7. ACKNOWLEDGMENTS

We thank anonymous referees for helpful comments and suggestions.

8. REFERENCES

[1] Teppo E. Ahonen, Kjell Lemström, and Simo Linkola. Compression-based similarity measures in symbolic, polyphonic music. In *Proceedings of the 12th International Symposium on Music Information Retrieval (ISMIR'11)*, pages 91–96, 2011.

[2] Yoko Anan, Kohei Hatano, Hideo Bannai, and Masayuki Takeda. Music genre classification using similarity functions. In *Proceedings of the 12th International Symposium on Music Information Retrieval (ISMIR'11)*, pages 693–698, 2011.

[3] Mark A. Bartsch and Gregory H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.

[4] James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 65:473–484, 2006.

[5] Rudi Cilibrasi, Paul Vitányi, and Ronald de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.

[6] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

[7] Christopher DeCoro, Zafer Barutcuoglu, and Rebecca Fiebrink. Bayesian aggregation for hierarchical genre classification. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, pages 77–80, 2007.

[8] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

[9] Simon Dixon, Werner Goebl, and Gerhard Widmer. The performance worm: Real time visualisation of expression based on langner's tempo-loudness animation. In *Proceedings of the International Computer Music Conference (ICMC'02)*, pages 361–364, 2002.

[10] Daniel P. W. Ellis. Classifying music audio with timbral and chroma features. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR '07)*, pages 339–340, 2007.

[11] Rong-En Fan, Pai-Hsuen Chen, and Chih-Jen Lin. Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[12] Christopher Harte, Mark Sandler, and Martin Gasser. Detecting harmonic change in musical audio. In *Proceedings of 1st ACM Workshop on Audio and Music Computing Multimedia (AMCMM'06)*, pages 21–26, 2006.

[13] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. String quartet classification with monophonic models. In *Proceedings of the 11th International Society for Music Information Retrieval (ISMIR '10)*, pages 537–542, 2010.

[14] D.S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

[15] Christina S. Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[16] Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: a string kernel for SVM protein classification. In *Proceedings of the Pacific Symposium on Biocomputing (PSB'02)*, pages 566–575, 2002.

[17] Christina S. Leslie and Rui Kang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5:1435–1455, 2004.

[18] Ming Li and Ronan Sleep. Melody classification using a similarity metric based on Kolmogorov complexity. In *Sound and Music Computing*, pages 126–129, 2004.

[19] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, pages 34–41, 2005.

[20] Thomas Lidy, Andreas Rauber, Antonio Pertusa, and José Manuel Iñesta. Improving genre classification by combination of audio and symbolic descriptors using a transcription system. In *Proceedings of 8th International Conference on Music Information Retrieval (ISMIR'07)*, pages 61–66, 2007.

[21] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, Chris Watkins, and Bernhard Scholkopf. Text classification using string kernels. *Journal of Machine Learning Research*, 2:563–569, 2002.

[22] Riccardo Miotto and Nicola Orio. A music identification system based on chroma indexing and statistical modeling. In *Proceedings of the 9th International Society for Music Information Retrieval (ISMIR '08)*, pages 301–306, 2008.

[23] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR '05)*, pages 288–295, 2005.

[24] Laurent Oudre, Yves Grenier, and Cédric Févotte. Chord recognition by fitting rescaled chroma vectors to chord templates. *IEEE Transactions on Audio, Speech and Language Processing*, 19(7):2222 – 2233, 2011.

[25] Carlos Pérez-Sancho, David Rizo, and José Manuel Iñesta Quereda. Genre classification using chords and stochastic language models. *Connection Science*, 21:145–159, 2009.

[26] Craig Saunders, David R. Hardoon, John Shawe-taylor, and Gerhard Widmer. Using string kernels to identify famous performers from their playing style. In *Proceedings of the 15th European Conference on Machine Learning (ECML'04)*, pages 384–395, 2004.

[27] Takuya Sawada and Ken Satoh. Composer classification based on patterns of short note sequences. In *Proceedings of the AAAI-2000 Workshop on AI and Music*, pages 24–27, 2000.

[28] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals. In *Proceedings of the 2nd International Symposium on Music Information Retrieval (ISMIR'01)*, pages 293–302, 2001.

[29] S. V. N. Vishwanathan and Alexander J. Smola. Fast kernels for string and tree matching. In *Advances on Neural Information Processing Systems 15*, pages 569–576, 2002.

[30] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[31] Liwei Wang, Masashi Sugiyama, Cheng Yang, Kohei Hatano, and Jufu Feng. Theory and algorithm for learning with dissimilarity functions. *Neural Computation*, 21(5):1459–1484, 2009.

[32] Manfred K. Warmuth, Karen Glocer, and Gunnar Rätsch. Boosting algorithms for maximizing the soft margin. In *Advances in Neural Information Processing Systems 20 (NIPS'08)*, pages 1585–1592, 2008.