

Automatic Han Chinese Folk Song Classification Using Extreme Learning Machines

Suisin Khoo, Zhihong Man, and Zhenwei Cao

Faculty of Engineering and Industrial Sciences
Swinburne University of Technology
VIC 3122 Australia
suisin.khoo@gmail.com, {zman, zcao}@swin.edu.au

Abstract. Multilayer feedforward neural networks trained via supervised learning have proven to be successful in pattern recognition. This paper presents the technique of using single hidden layer feedforward neural network as an automatic classifier in music classification. Han Chinese folk songs from five distinct geographical regions in China are studied and encoded using a novel musical feature density map (MFDMap) for machine classification. The extreme learning machine (ELM) and its two variants are employed as the classifiers to categorize the folk songs. Our simulations show that by using a low-pass finite impulse response extreme learning machine (FIR-ELM), we can achieve 80.65% classification accuracy.

Keywords: Musical feature density map, music classification, extreme learning machine, artificial neural network, Han Chinese folk song.

1 Introduction

Automatic music classification has been widely studied in the past decade, but the majority of the works focuses on Western music. Some examples are [1], [2] and [3]. Though Chinese music classification has not been ignored, little effort has been made in this area. An example is the classification of Chinese folk music using Hidden Markov model [4]. Two sets of perceptual features were used for the purpose of classifying Chinese folk music according to the audio taxonomy defined by the authors. As the authors defined Chinese folk music as either vocal or instrumental, they focused mainly on the timbral and rhythmic features.

As far as we are aware, there is only one work for Chinese music classification that involved using an artificial neural network (ANN) as the classifier. Xu, Wang and Yan [5] used a radial basis function neural network to classify Chinese folk songs based on 74 features extracted from the audio signals. The authors achieve 43.208% classification accuracy using Classification Contribution-Ratio Based Selection algorithm for music feature selection.

The single hidden layer feedforward neural network (SLFN) is one of the simplest and most popular types of ANN. Hornik, Stinchcombe and White [6] have proved that a SLFN, using any desired activation function, is a universal approximator. Such

SLFN is able to approximate virtually any function if given a sufficient number of hidden neurons and a sufficiently large training set. The extreme learning machine (ELM) [7] is an emerging technology that utilized a SLFN. It has been proved to outperform conventional SLFNs, especially the gradient-based SLFNs [7]. Nonetheless, ELM technique has limited attention in music classification research. The only research that employs ELM as a music classifier is [8], where a four genre (classical, dance, pop and rock) music classification is performed. Khoo, Man and Cao [9] employed the regularized extreme learning machine (R-ELM) as a geographical based classifier for Han Chinese folk song.

In this paper, we propose using the finite impulse response extreme learning machine (FIR-ELM) as the music classifier for Han Chinese folk songs. The FIR-ELM is a variant of ELM which incorporated two improvements to the ELM. It introduced two balancing parameters to adjust the balance of the empirical risk and the structural risk of the neural network and improved the limitation of the arbitrary characteristic in both ELM and R-ELM by introducing the filtering technique to the input layer of the SLFN. It is more robust especially in solving complex problem such as the one discussed in this paper.

The corpus for our music classification task is the Han Chinese folk songs from five distinct regions in China. There are two distinct approaches for music representation: audio and symbolic. In this paper, we employ a novel symbolic musical feature representation method, the musical feature density map (MFDMap). Unlike audio representation, our MFDMap is very similar to musical notations and hence more appropriately model the human perception of music. In addition, the characteristic of symbolic representation facilitates the possibility of building intelligent systems for music theory teaching and learning.

This paper is organized as follows. In Section 2, we will describe our proposed musical feature density map and the data set used for the simulations. A brief overview of the ELM, the R-ELM and the FIR-ELM will be presented in Section 3. In Section 4, the simulation results will be presented and discussed. Finally, we will conclude our work in Section 5 and propose ideas for further research.

2 Music Representation

In this paper, musical features were extracted from Kern files. There is no particular reason for choosing this format except that the database we employed is coded in Kern format. It is important to note that from a practical perspective, the features discussed in this section could just as easily be extracted from other existing symbolic formats such as MIDI notation.

2.1 Solfege – To Characterize Melody Progression of Notes

One of the main characteristics of folk songs is their method of transmission. Unlike other styles of music, such as classical music, folk songs are transmitted orally. Much existing research that adopts symbolic representation overlooks this important characteristic of folk songs. They usually employ the exact pitch representation of the song

melody. As folk songs are transmitted orally, the key of the tune in which a person sings may vary. This means that the tune of a folk song may be sung in many different pitches but the melody progression of all notes remains unchanged. In this case, exact pitch representation will interpret the different versions independently as different songs instead of as the same song, which results in inaccuracy.

Instead of using the exact pitch representation, we proposed using solfege representation of the song melody. Solfege is a solmization technique that is commonly used for sight singing. Each note in a musical scale is assigned to a unique syllable. These syllables are usually written as numerical notations in musical scores of Chinese folk songs. Solfege notation can be easily computed using the available pitch and key information. It is a consistent and robust feature. Despite the different key of each version of the same folk song, the solfege representation always remains the same. The solfege representation can be computed using the following equation:

$$\text{solfege}_i(n) = \text{solfege}(n) + (i \times 7) \quad (1)$$

for $n = 1, 2, 3, \dots, N$, where N is the number of notes in a folk song and i is the number of octaves above or below the Middle C.

2.2 Interval – To Characterize Musical Flows

From a broad geographical view, the Han Chinese folk songs can be divided into the northern and southern style, each of which is associated with the two major rivers in China: the Yellow River of the north and the Chang Jiang of the south. In this division, the environment, climate and landscape structure play a significant role in forming the characteristics of the folk songs from the different regions. For example, the cold, dry and windy climate in the north that affects the agricultural activities and lifestyle of the people is reflected by a more intense and disjunct progression of melody in folk songs from that area. Conversely, the folk songs from the southern regions are more lyrical and conjunct. This important characteristic of the musical flow and movement of folk songs is reflected in our encoding scheme through the measurement of the musical interval and changes in duration between adjacent notes.

The musical interval is the measurement of the pitch ratio between two adjacent notes. We measure the interval in terms of the number of semitones between two adjacent notes. The intervals of a folk song melody with N number of notes can be calculated using the following equation:

$$\text{interval}(n) = \text{pitch}(n) - \text{pitch}(n-1) \quad (2)$$

for $n = 2, 3, 4, \dots, N$, where N is the number of notes.

2.3 Duration – To Characterize Rhythmic Pattern

Rhythmic patterns define the distinctive identity between music of diverse style and form. For example, a melody with lots of short notes within a short length of time presents an agitated impression but a chunk of lengthy notes usually creates a more

graceful feeling. We encoded the duration of each musical note in a folk song to characterize its rhythmic pattern. The numerical notation we employed to represent different lengths of note is the number of crotchets (♩) per measurement. We encoded 19 different note durations that are common in Han Chinese folk songs.

2.4 Duration Ratio – To Characterize Rhythmical Changes

As mentioned previously, the geographical characteristics of the origin of a folk song have significant influences in the texture of the folk song. Duration ratio measures the rhythmical changes. A large ratio reflects a drastic change in rhythmic progression which portrays a rougher, more intense and disjunct texture. On the other hand, a smaller ratio depicts a smooth, more continuous and lyrical texture. The duration ratio can be computed using the following equation:

$$\text{duration ratio}(n) = \frac{\text{duration}(n)}{\text{duration}(n-1)} \quad (3)$$

for $n = 2, 3, 4, \dots, N$, where N is the number of notes in each folk song.

2.5 Musical Feature Density Map

In the previous sections, useful musical features are extracted for folk song classification. As the folk songs are of varied length, in order to perform the classification, we need to specify a unified length for all. Most existing research papers, for example both [1] and [5], employ a windowing method. In this method, a window of specific size is defined and a song is sliced into many fragments of the same size. These windows of music fragments are then used as the input to the music classifier.

Although the windowing method is useful, it imposed a few limitations. Firstly, there is no universal value for the window size. Varied window size might result in diverse classification accuracy. Therefore, an exhaustive testing needs to be carried out to determine the most suitable window size. Next, the size of a window determines the amount of information it encapsulates. If a window size is too small, there might not be enough information to define the characteristic of a class. In addition, more than one window might contain the same data. Finally, dividing a song into fragments might result in loss of continuity and integrity of the music.

In this paper, we propose using a musical feature density map which overcomes the above mentioned limitations. MFDFMap is a map that portrays the density of each musical feature value in a song. Instead of fragmenting the songs, each song is considered as a whole. The MFDFMap consists of 175 map locations: 46 for solfege, 49 for interval, 19 for duration and 61 for duration ratio. The value for each map location is calculated as a percentage:

$$\text{map location}_i = \frac{\text{frequency of musical feature value}_i}{N} \times 100 \quad (4)$$

for $i = 1, 2, 3, \dots, 175$ and N is the total number of notes in a song. Note that instead of N , we use $N - 1$ for interval features and duration ratio features.

One of the advantages of MFDDMap is that it preserves the main characteristic of symbolic representations, that is, it closely models the human perception of music. Most musical styles can be differentiated by the high occurrence of certain musical features or the absence of them. This is effectively modelled in the MFDDMap. The structure of MFDDMap is flexible. It allows all musical features to be encompassed in one map. Any musical features that are not applicable to a particular musical style can be easily removed from the map. MFDDMap is also extendable whereby new musical features can be added to it.

By using the occurrence percentage of each musical feature within a song, MFDDMap encompasses the complete melody instead of segmenting it into chunks of notes. This is important as music is usually continuous. Each musical note is not an independent object but is closely related to the preceding note(s) and the successive note(s). MFDDMap uses the whole song and extracts useful musical features from it, each song has a unique map and the problems of finding a representative segment size can be easily avoided.

2.6 Data Set

We have chosen Chinese folk songs from five regions: Jiangsu, Dongbei, Guangdong, Shanxi and Sichuan as our corpus from the larger database of Han Chinese folk songs from [10]. The melody of each folk song in the corpus is encoded into a MFDDMap of its own. There are a total of 312 folk songs in our corpus. 90% of the songs are randomly assigned as the training set and the remaining form the testing set.

3 Machine Classifier

3.1 Extreme Learning Machine

The major bottlenecks of the gradient-based single hidden layer feedforward neural network are the slow learning speed and the convergence issue. These are caused by the learning algorithms that are employed by the SLFN. Gradient-based algorithms suffer from the problem of choosing the learning step that gives good convergence. In addition, the parameters such as the input weights and the hidden layer biases of these learning algorithms need iterative tuning to obtain better learning performance which results in a time consuming and resource consuming process.

The extreme learning machine is proposed by Huang, Zhu and Siew [7] to overcome these drawbacks. ELM algorithm utilizes SLFN architecture but unlike the gradient-based SLFN, ELM randomly chooses the input weights and analytically determines the output weights of a SLFN. In [7], the ELM was theoretically proven to give good generalization performance at an extremely fast learning speed. The brief overview of the ELM algorithm is described as follows.

For a dataset with N distinct samples $\{(X, T) \mid \mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N], \mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_N]\}$ where $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbf{R}^n$ is the input vector and $\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in \mathbf{R}^m$ is the target vector, the SLFN with $\tilde{N}$ hidden neurons can be written as

$$\mathbf{o}_i = \sum_{j=1}^{\tilde{N}} \beta_j g(\mathbf{w}_j \mathbf{x}_i + b_j) \quad (5)$$

for $i = 1, 2, \dots, N$ where $\mathbf{o}_i = [o_{i1}, o_{i2}, \dots, o_{im}]^T \in \mathbf{R}^m$ is the output vector with respect to $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$ the input vector, $\boldsymbol{\beta}_j = [\beta_{j1}, \beta_{j2}, \dots, \beta_{jm}]^T$ is the output weights vector connecting the j th hidden neuron and the output neurons, $\mathbf{w}_j = [w_{j1}, w_{j2}, \dots, w_{jn}]^T$ is the input weights vector connecting the input neurons and the j th hidden neuron, b_j is the bias of the j th hidden neuron and $g(x)$ is the activation function. This SLFN with $\tilde{N}$ hidden neurons and activation function $g(x)$ is proven to be able to approximate N data samples with zero error, such that

$$\sum_{i=1}^N \|\mathbf{o}_i - \mathbf{t}_i\| = 0. \quad (6)$$

Therefore, there exist $\boldsymbol{\beta}_j$, $\mathbf{w}_j$ and b_j such that

$$\mathbf{t}_i = \sum_{j=1}^{\tilde{N}} \boldsymbol{\beta}_j g(\mathbf{w}_j \mathbf{x}_i + b_j) \quad (7)$$

for $i = 1, 2, \dots, N$. The above (7) can then be written compactly in matrix form $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$ where

$$\mathbf{H}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{\tilde{N}}, b_1, b_2, \dots, b_{\tilde{N}}, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) \\ = \begin{bmatrix} g(\mathbf{w}_1 \mathbf{x}_1 + b_1) & g(\mathbf{w}_2 \mathbf{x}_1 + b_2) & \cdots & g(\mathbf{w}_{\tilde{N}} \mathbf{x}_1 + b_{\tilde{N}}) \\ g(\mathbf{w}_1 \mathbf{x}_2 + b_1) & g(\mathbf{w}_2 \mathbf{x}_2 + b_2) & \cdots & g(\mathbf{w}_{\tilde{N}} \mathbf{x}_2 + b_{\tilde{N}}) \\ \vdots & \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \mathbf{x}_N + b_1) & g(\mathbf{w}_2 \mathbf{x}_N + b_2) & \cdots & g(\mathbf{w}_{\tilde{N}} \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \quad (8)$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \boldsymbol{\beta}_2^T \\ \vdots \\ \boldsymbol{\beta}_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}. \quad (9)$$

The ELM algorithm works by first randomly initializing the input weights and hidden layer biases of the SLFN. Then, the hidden layer output $\mathbf{H}$ is calculated. Finally, the output weights $\boldsymbol{\beta}$ are computed using the Moore-Penrose generalized inverse:

$$\boldsymbol{\beta} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}. \quad (10)$$

The major difference between the ELM algorithm and gradient-based learning algorithms is the tuning of the hidden layer. In conventional algorithms, the learning process needs to be repeated many times to tune the input weights but ELM does not require such tuning and its optimal output weights are calculated analytically without any iterative procedure. This results in an extremely fast learning speed and a simpler architecture with fewer parameters to tune. In addition, ELM determines its output weights using the generalized inverse operation which assures a unique solution with the smallest norm. This allows the ELM algorithm to effectively avoid the local minima issue.

Although ELM greatly improved the performance of conventional SLFNs, it still poses a few drawbacks. Firstly, the random assignment of the input weights and hidden layer biases leads to the generation of non-optimal solutions. The arbitrary characteristic of the randomly assigned input weights is similar to the local minima problem in gradient-based algorithm. It results in diverse solutions by an ELM over different trials. Hence, it is difficult to determine the optimal solution in a single run. Repeated trials of testing are required to evaluate the mean classification accuracy of an ELM. In addition, there is no universal value or range of values to set the input weights. Hence, the choice of values is purely empirically based. The next section will further discuss an enhancement to this limitation of ELM.

Next, the design of the output layer weights in ELM gives rise to another issue. ELM uses the generalized inverse of the hidden layer output matrices to determine the output weights. This minimum norm least squares solution of the hidden layer output is an empirical risk minimization (ERM) operation which tends to result in an overfitting model especially if the training set is not sufficiently large. In statistical learning theory, the real prediction risk in learning consists of empirical risk and structural risk. In order to achieve good generalization performance, a model needs to accomplish a good balance between these two risks.

Deng, Zheng and Chen [11] proposed to overcome the drawback in the output weights by introducing a regularization term into the ELM algorithm. A weight factor γ for empirical risk is inserted to regularize the proportion of the empirical risk and the structural risk. Their improved ELM is called the regularized extreme learning machine and it used the following equation to calculate the output weights β :

$$\beta = \left(\frac{\mathbf{I}}{\gamma} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}. \quad (11)$$

3.2 Finite Impulse Response Extreme Learning Machine

As mentioned in the previous section, the ELM poses two major limitations: the robustness issues and the effect of the structural and empirical risks. R-ELM was proposed to enhance the limitation in the output weights by introducing a balancing parameter in the optimization of the error function. However, this improvement does not significantly reduce the structural and empirical risks since the input weights and the hidden layer biases are still randomly assigned. The finite impulse response extreme learning machine proposed by Man et al. [12] served to further improve the ELM algorithm, particularly to overcome the robustness issue of the input weights and the hidden layer biases.

In the FIR-ELM, the input weights are designed such that the hidden layer of the SLFN serves as a pre-processor to remove the effects of the input disturbances. As the output of a linear hidden neuron in a SLFN is the sum of the weighted input data, each of these hidden neurons can be treated as an FIR filter. Hence, based on the FIR filter design techniques in signal processing, the hidden layer of a SLFN can be designed as a group of low-pass, high-pass, band-pass or band-stop filters, or any other types of filter to pre-process input data with disturbance or undesired frequency

components. The main advantages of such design are that the input disturbances and undesired frequency components can be removed and that both the empirical and structural risks of the SLFN can be greatly reduced. The output weight matrix in FIR-ELM is designed to further balance the empirical risk and structural risk. It is calculated based on minimizing an objective function that includes both the weighted sum of the output error squares and the weighted sum of the output weight squares of the SLFN.

It should be noted that there is one main difference between FIR-ELM and both the ELM and R-ELM. The SLFN in both ELM and R-ELM uses nonlinear hidden neurons and linear output neurons. However, in FIR-ELM, the SLFN uses both linear hidden neurons and linear output neurons. In order to assure that the SLFN with linear hidden neurons has the capability of universal approximation, an input tapped-delay-line memory with $n - 1$ delay units is added to the input layer. Fig. 1 depicts the SLFN architecture of the FIR-ELM, where D is the $n - 1$ time-delay element that is added to the input layer to form the tapped-delay-line memory, the input sequence $x(k), x(k - 1), \dots, x(k - n + 1)$ represents a time series consisting of the present observation $x(k)$ and the past $n - 1$ observations of the process, the hidden layer has $\tilde{N}$ linear neurons and the output layer has m linear neurons.

The definition for the SLFN with $\tilde{N}$ hidden neurons and m output neurons using FIR-ELM algorithm is similar to the ELM from equation (5) to equation (9). However, instead of randomly assigned the input weights, input weights w_{ij} for the i th hidden neuron of the SLFN in FIR-ELM is obtained as follows:

$$w_{i1} = \hat{h}_{id}[0], w_{i2} = \hat{h}_{id}[1], \dots, w_{in} = \hat{h}_{id}[n-1] \quad (12)$$

where

$$\hat{h}_{id}[k] = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{-j\omega(n-1)/2} e^{j\omega k} d\omega = \frac{\sin[\omega_c(k - (n-1)/2)]}{\pi(k - (n-1)/2)} \quad (13)$$

for $0 \leq k < n - 1$ is the impulse response of a truncated low-pass filter for the i th hidden neuron and ω_c is the cut-off frequency of the low-pass filter. It is noted that the similar design methods can be used to design the hidden neurons as high-pass,

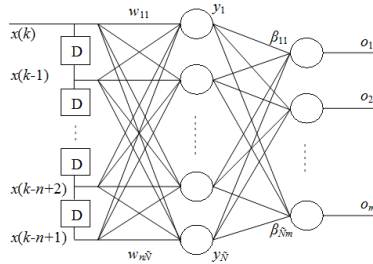


Fig. 1. The FIR-ELM network architecture with linear neurons and time-delay elements

band-pass or band-stop filter, or other types of filter for the purpose of pre-processing the input data.

The output weights for the SLFN is then calculated by minimizing both the weighted sum of the output error squares and the weighted sum of the output weights squares of the SLFN:

$$\text{Minimize } \left\{ \frac{\gamma}{2} \|\boldsymbol{\varepsilon}\|^2 + \frac{d}{2} \|\boldsymbol{\beta}\|^2 \right\} \quad (14)$$

$$\text{subject to } \boldsymbol{\varepsilon} = \mathbf{O} - \mathbf{T} = \mathbf{H}\boldsymbol{\beta} - \mathbf{T} \quad (15)$$

where γ and d are constant balancing parameters for adjusting the balance of the empirical risk and the structural risk. Hence, the output weights of the SLFN in FIR-ELM can be obtained as follows:

$$\boldsymbol{\beta} = \left(\frac{d}{\gamma} \mathbf{I} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}. \quad (16)$$

4 Simulations and Results

In order to examine the performance of the finite impulse response extreme learning machine as the machine classifier for Han Chinese folk song classification, we conducted our simulations using three classifiers: the ELM, the R-ELM and the FIR-ELM. The maximum number of hidden neurons employed in all classifiers is up to 2000 neurons. As the input weights in both ELM and R-ELM are randomly assigned, each repetition of the simulation will produce a different result. Hence, the simulations using ELM and R-ELM were repeated 50 times for the mean classification accuracy.

In FIR-ELM, the length of the FIR filter is the same as the size of the MFDMap, which is 175. The balancing parameters are set as $\gamma = 1$ and $d = 0.01$ based on empirical studies. The simulations were performed using four different types of FIR filter: low-pass, high-pass, band-pass and band-stop filters, over a range of cutoff frequencies ω_c ranging from 0.1 to 0.9 with a step size of 0.1. A band width of ± 0.05 is used for the band-pass and band-stop filters. The targets for the three classifiers are set using the 1-of- c method by assigning each of the five geographical regions to one target. For a set of targets, the one representing a particular region is assigned '1' and the remaining targets are assigned '0'.

The comparison of the classification accuracy of the three classifiers is presented in Table 1. Since both ELM and R-ELM possess arbitrary characteristics, the simulation was repeated 50 times to obtain the mean classification accuracy and this is shown in Table 1 together with their respective standard deviations. In FIR-ELM simulation, the low-pass filter performs the best among the four different filters. Hence the classification performance for FIR-ELM shown in Table 1 is the low-pass FIR-ELM with cutoff frequency at 0.3.

Table 1. Classification performance of ELM, R-ELM and low-pass FIR-ELM

Number of Hidden Neurons	ELM		R-ELM		FIR-ELM (low-pass)
	Accuracy (%)	Standard Deviation (%)	Accuracy (%)	Standard Deviation (%)	Accuracy (%)
50	51.94	7.96	55.35	6.16	29.03
100	50.77	7.22	60.97	5.80	48.39
200	44.00	7.37	63.29	5.71	61.29
500	45.03	9.15	68.45	5.91	67.74
1000	58.06	7.90	69.03	5.13	74.19
1200	60.84	5.97	70.39	5.28	80.65
1500	62.13	6.35	68.71	5.30	80.65
2000	64.97	6.29	69.74	5.29	67.74

It is fairly obvious in Table 1 that increasing the number of hidden neurons usually leads to some increase in performance for each classifier. Due to the nature of music, there is great subjectivity that makes music classification a difficult task to accomplish. Although there are differences between folk songs from various geographical regions, the divergence is not easily recognized. In the Han Chinese folk song classification, the differences between the categories are fairly subtle. Sorting these folk songs into regional source is not an obvious task even for humans. The performance shown by these machine classifiers is very encouraging.

In general, the low-pass FIR-ELM is the best classifier among the three. The R-ELM performs slightly better than ELM. The FIR-ELM shows its best performance at 1200 hidden neurons, achieving an accuracy of 80.65% while R-ELM accomplishes 70.39% accuracy and ELM 60.84% accuracy. The performance of the ELM fluctuates a little when the number of hidden neurons is less than 500 but begins to show steady improvement in performance when more hidden neurons are added and reaches 64.97% accuracy with 2000 hidden neurons. The performance of R-ELM on the other hand is fairly steady with a slight drop from 1500 hidden neurons. The low-pass FIR-ELM is the most robust among the three classifiers. Its classification accuracy improves as the number of hidden neurons increases and reaches its saturation point at 1200 hidden neurons. The FIR-ELM manages to maintain this good performance until 1500 hidden neurons before starting to show a hint of deterioration.

The learning time, in number of seconds, required for each classifier to achieve a complete learning of the music classification task is shown in Table 2 to demonstrate the performance speed of each classifier. Overall, these classifiers perform at an extremely fast speed. The maximum time required by a classifier at 2000 hidden neurons is 3.3 seconds. The low-pass FIR-ELM starts off being the fastest but eventually become the slowest among the three classifiers when the number of hidden neurons increases. The learning time required by the R-ELM is about the same as the FIR-ELM while ELM performs at a slightly faster speed than both R-ELM and FIR-ELM between 1000 to 2000 hidden neurons.

Table 2. Learning time in number of seconds for ELM, R-ELM and FIR-ELM classifiers

Number of Hidden Neurons	ELM (sec.)	R-ELM (sec.)	FIR-ELM (low-pass) (sec.)
50	0.0163	0.0049	0.0046
100	0.0270	0.0068	0.0066
200	0.0752	0.0131	0.0113
500	0.2274	0.0812	0.1022
1000	0.2910	0.4454	0.5115
1200	0.3147	0.7212	0.7566
1500	0.3562	1.3228	1.3881
2000	0.4172	2.9797	3.2972

5 Conclusion and Future Work

We have introduced our novel symbolic music encoding method known as the MFDMMap. We have demonstrated the feasibility of using our MFDMMap in machine classification of Han Chinese folk songs through simulations using the extreme learning machine, the regularized extreme learning machine and the finite impulse response extreme learning machine. Our simulation results have successfully shown that the finite impulse response extreme learning machine using a low-pass filter is the best classifier among the three classifiers. It is more robust especially in solving a complex task such as music classification.

In this paper, our MFDMMap employed only symbolic representations of the musical features. One possible enhancement is to extend the versatility and capabilities of our MFDMMap to also incorporate musical features extracted from audio signals, especially timbral related features. In addition, the corpus employed by the MFDMMap can be extended to include folk songs of other geographical regions in China.

References

1. Tzanetakis, G., Cook, P.: Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing* 10, 293–302 (2002)
2. Xu, C., Maddage, N.C., Shao, X., Cao, F., Tian, Q.: Musical Genre Classification Using Support Vector Machines. In: *International Conference on Acoustics, Speech and Signal Processing*, pp. 429–432 (2003)
3. McKay, C., Fujinaga, I.: Automatic Genre Classification Using Large High-Level Musical Feature Sets. In: *5th International Conference on Music Information Retrieval*, pp. 525–530 (2004)
4. Liu, X., Yang, D., Chen, X.: New Approach to Classification of Chinese Folk Music Based on Extension of HMM. In: *International Conference on Audio, Language and Image Processing*, pp. 1172–1179 (2008)
5. Xu, J., Wang, P., Yan, L.: Feature Selection for Automatic Classification of Chinese Folk Songs. In: *Congress on Image and Signal Processing*, pp. 441–446 (2008)

6. Hornik, K., Stinchcombe, M., White, H.: Multilayer Feedforward Network Are Universal Approximators. *Neural Networks* 2(5), 359–366 (1989)
7. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme Learning Machine: Theory and Applications. *Neurocomputing* 70, 489–501 (2006)
8. Loh, Q.-J.B., Emmanuel, S.: ELM for Classification of Music Genres. In: *International Conference on Control, Automation, Robotics and Vision*, pp. 1–6 (2006)
9. Khoo, S., Man, Z., Cao, Z.: Automatic Han Chinese Folk Song Classification Using The Musical Feature Density Map. Accepted: 6th International Conference on Signal Processing and Communication Systems (2012)
10. Schaffrath, H.: The Essen Folksong Collection in Kern Format. In: Huron, D. (ed.) *Menlo Park, CA* (1995)
11. Deng, W., Zheng, Q., Chen, L.: Regularized Extreme Learning Machine. In: *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 389–395 (2009)
12. Man, Z., Lee, K., Wang, D., Cao, Z., Miao, C.: A New Robust Training Algorithm for a Class of Single-Hidden Layer Feedforward Neural Networks. *Neurocomputing* 74, 2491–2501 (2011)