Brain Inspired Cognitive Systems
August 29 – September 1, 2004
University of Stirling, Scotland, UK

CLASSIFICATION OF RECORDED CLASSICAL MUSIC: A METHODOLOGY AND A COMPARATIVE STUDY

R. MALHEIRO, CISUC – Centro de Informática e Sistemas da Universidade de Coimbra
Departamento de Engenharia Informática, PÓLO II da Universidade de Coimbra,
Pinhal de Marrocos, P 3030, Coimbra, Portugal

R. P. Paiva, CISUC
A. J. Mendes, CISUC
T. Mendes, CISUC
A. Cardoso, CISUC

ABSTRACT

As a result of recent technological innovations, there has been a tremendous growth in the Electronic Music Distribution industry. Consequently, tasks such as automatic music genre classification address new and exciting research challenges.

Automatic music genre recognition involves issues like feature extraction and development of classifiers using the obtained features.

We use the number of zero crossings, loudness, spectral centroid, bandwidth and uniformity for feature extraction. These features are statistically manipulated, making a total of 40 features.

Regarding the task of genre modeling, we follow three approaches: the *K*-Nearest Neighbors (KNN) classifier, Gaussian Mixture Models (GMM) and feedforward neural networks (FFNN).

A taxonomy of sub-genres of classical music is used. We consider three classification problems: in the first one, we aim at discriminating between music for flute, piano and violin; in the second problem, we distinguish choral music from opera; finally, in the third one, we seek to discriminate between all five genres.

The best results were obtained using FFNNs: 85% classification accuracy in the three-class problem, 90% in the two-class problem and 76% in the five-class problem. These results are encouraging and show that the presented methodology may be a good starting point for addressing more challenging tasks.

1. INTRODUCTION

Presently, whether it is the case of a digital music library, the Internet or any music database, search and retrieval is carried out mostly in a textual manner, based on categories such as author, title or genre. This approach leads to a certain number of difficulties for service providers, namely in what concerns music labeling. Real-world music databases from sites like AllMusicGuide or CDNOW grow larger and larger on a daily basis, which requires a tremendous amount of manual work for keeping them updated.

Thus, simplifying the task of music database organization would be an important advance. This calls for automatic classification systems. Such systems should overcome the limitations resulting from manual song labeling, which may be a highly time-consuming and subjective task.

Some authors have addressed this problem recently. Tzanetakis and Cook [1] classify music in ten genres, namely, classical, country, disco, hip-hop, jazz, rock, blues, reggae, pop and metal. They further classify classical music into choir, orchestra, piano and string quartets. Features used encompass three classes: timbre, rhythm and pitch-related features. The authors investigate the importance of the features in training statistical pattern recognition classifiers, particularly, Gaussian Mixture Models and k-nearest neighbors. 61% accuracy was achieved for discriminating between the ten classes. As for classical music classification, an average accuracy of 82.25% was achieved. Golub [2] uses seven classes of mixed similarity (a capella, celtic, classical, electronic, jazz, latin and pop-rock). The features used are loudness, spectral centroid, bandwidth and uniformity, as well as statistical features obtained from them. A generalized linear model, a multi-layer perceptron and a k-nearest classifier were used. The best of them achieved 67% accuracy. Kosina [3] classifies three highly dissimilar classes (metal, dance and classical) using k-nearest neighbors. The used features were mel-frequency cepstral coefficients, zero-crossing rate, energy and beat. 88% accuracy was achieved. Martin [4] addresses the problem of instrument identification. He proposes a set of features related to the physical properties of the instruments with the goal of identifying them in a complex auditory environment.

In our work we aim at classifying five sub-genres of classical music, namely opera, choral music and music for flute, piano and violin. This is due to the fact that there are not many studies regarding specifically classical music. Also, digital music libraries have a great diversity of taxonomies of classical music, which demonstrates its practical usefulness.

Unlike other authors, who use a broad range of generic classes, we chose to focus on specific set of related classes. This seems to be a more challenging problem since our classes show a higher similarity degree, leading to, we think, a more difficult classification problem. We chose a set of features based on those used in [5] and [2], encompassing especially timbre and pitch content, which seemed relevant for the task under analysis: the number of zero crossings, loudness, spectral centroid, bandwidth and uniformity. Rhythmic features were not used. We used a KNN classifier, a GMM, trained with the Expectation Maximization algorithm, and a FFNN classifier, trained via the Levenberg-Marquardt algorithm. For validation purposes we obtained 76% accuracy in the five-class problem, using FFNNs. Our results, though far from ideal, are satisfactory. Comparing to [1], we got a similar accuracy using one more category and a reduced feature set.

This paper is organized as follows. Section 2 describes the process of feature extraction and the features used. In Section 3, a short overview of the followed methodologies is presented: KNN, GMM and FFNNs Experimental results are presented and analyzed in Section 4. Finally, in Section 5 some conclusions are drawn, as well as possible directions for future work.

2. FEATURE EXTRACTION

Based on the classification objectives mentioned above, and taking into account the results obtained in similar works, we gave particular importance to features with some significance for timbral and pitch content analysis. We used no rhythmic features, since they did not seem very relevant for the type of music under analysis. However, we plan to use them in the future and evaluate their usefulness in this context.

We started by selecting 6 seconds' segments from each musical piece (22khz sampling, 16 bits quantization, monaural). Since for training issues the segment samples used should have little ambiguity regarding the category they belong to, we selected relevant segments from each piece. The purpose was not to use long training samples. Instead, short significant segments are used, mimicking the way humans classify music, i.e., short segments [6] using only music surface features without any higher-level theoretical descriptions [7].

After collecting a relevant segment for each piece, the process of feature extraction is started by dividing each 6s signal in frames of 23.22 with 50% overlap. This particular frame length was defined so that the number of samples in each frame is a power of 2, which is necessary for optimizing the efficiency of Fast Fourier Transform (FFT) calculations [8] (Section 2.2). This gives 512 samples per frame, in a total of 515 frames.

Both temporal and spectral features are used, as described below.

2.1. TIME-DOMAIN FEATURES

We use two temporal features: loudness and the number of zero crossings. Loudness is a perceptual feature that tries to capture the perception of sound intensity. Only the amplitude is directly calculated from the signal. Loudness, i.e., the perception of amplitude, can be approximated as follows [2] (1):

$$L(r) = \log_2\left(1 + \frac{1}{N}\sum_{n=1}^{N}\left|x(n)\right|\right) \tag{1}$$

where L denotes loudness, r refers to the frame number, N is the number of samples in each frame, n stands for the sample number in each frame and $x(n)$ stands for the amplitude of the n-th sample in the current frame.

The number of zero crossings simply counts the number of times the signal crosses the time axis, as follows [5] (2):

$$Z(r) = \frac{1}{2}\sum_{n=1}^{N}\left|\operatorname{sgn}(x(n)) - \operatorname{sgn}(x(n-1))\right| \tag{2}$$

where Z represents the number of zero crossings. This is a measure of the signal frequency content, which is frequently used in music/speech discrimination and for capturing the amount of noise in a signal [1].

2.2. FREQUENCY-DOMAIN FEATURES

The spectral features used, computed in the frequency domain, are spectral centroid, bandwidth and uniformity. Therefore, the process starts by converting the signal into the frequency domain using the Short-Time Fourier Transform (STFT) [9]. The signal is divided in frames, as stated above. The signal for each frame is then multiplied by a Hanning window, which is characterized by a good trade-off between spectral resolution and leakage [8].

Spectral centroid is the magnitude-weighted mean of the frequencies [2] (3):

$$C(r) = \frac{1}{N}\frac{\sum_{k=1}^{N} M_r(k)\cdot\log_2 k}{\sum_{k=1}^{N} M_r(k)} \tag{3}$$

where $C(r)$ represents the value of the spectral centroid at frame r and $M_r(k)$ is the magnitude of the Fourier transform at frame r and frequency bin k. This is a measure of spectral brightness, important, for instance, in music/speech or musical instrument discrimination.

Bandwidth is the magnitude-weighted standard deviation of frequencies [2], as follows (4):

$$B(r) = \sqrt{\frac{\sum_{k=1}^{N}\left(C(r) - \log_2 k\right)^2 M_r(k)}{\sum_{k=1}^{N} M_r(k)}} \tag{4}$$

where $B(r)$ represents the spectral bandwidth at frame r. This is a measure of spectral distribution: lower bandwidth values denote a concentration of frequencies close to the centroid (which is the energy-weighted mean of frequencies), i.e., a more narrow frequency range.

Uniformity gives a measure of spectral shape. It measures the similarity of the magnitude levels in the spectrum and it is useful for discriminating between highly pitched signals (most of the energy concentrated in a narrow frequency range) and highly unpitched signals (energy distributed across more frequencies) [2]. Uniformity is computed as follows (5):

$$U(r) = -\sum_{k=1}^{N} \frac{M_r(k)}{\sum_{k=1}^{N} M_r(k)} \cdot \log_N \frac{M_r(k)}{\sum_{k=1}^{N} M_r(k)} \tag{5}$$

For each frame, the five features described are extracted. Then, first-differences are calculated, based on the feature values in consecutive frames, e.g., $L(r)$ - $L(r\text{-}1)$. These five new features plus the five features described before constitute our set of 10 basis features.

Classical music is usually characterized by accentuated variations in the basis features throughout time. Therefore, statistical manipulations of the basis features are calculated in order to cope with this aspect.

The means and standard deviations for the ten basis features are calculated in 2 seconds' chunks, leading to 20 features. The final features that compose the signature correspond to the means and standard deviations of the 20 intermediate features computed previously. We get a total of 40 features ($2\times2\times10$).

In order to avoid numerical problems in the classification models used (Section 3), all the features were normalized to the [0, 1] interval [10].

3. MUSIC GENRE CLASSIFICATION

Music genre classification can be regarded as a pattern recognition problem. In fact, the objective is to separate patterns corresponding to the different classes, using the extracted features.

In the present work, we compare three different strategies: a non-parametric approach, using the *k*-nearest neighbors algorithm; a statistical approach, based on Gaussian Mixture Models; and a neural network approach.

Regardless of the followed strategy, the available samples must be divided into two subsets: one training set, for modeling purposes, and one test set, for validation purposes. The model is evaluated by computing an error measure using the validation set.

3.1. K-NEAREST NEIGHBORS

The *k*-nearest neighbors algorithm is a simple non-parametric classifier. In this method, each sample is labeled according to the majority of its *k* nearest neighbors.

```
1. Compute the distance from feature vector $p$ to every
   training sample
2. Get the $k$ samples that are closest to $p$
3. Select the winner: the most represented class among
   the classes associated to the $k$ samples returned
```

Algorithm 1. *K*-Nearest Neighbors classification algorithm.

The algorithm works as follows (Algorithm 1). Given a feature vector *p*, we determine the *k* vectors in the feature space that are closest in distance to *p*. Here, we use the Euclidian distance. Then, we classify *p* according to the most represented category in the *k* samples returned.

In the present case, we use k = 1, 3 and 5.

3.2. GAUSSIAN MIXTURE MODELS

In statistical pattern recognition, the classification problem consists on the estimation of a probability density function (*pdf*) for the feature-vectors of each class. The Gaussian Mixture Model is a general methodology for the estimation of an unknown *pdf*.

In the GMM, one assumes that the *pdf* of each class consists of a mixture of multi-dimensional Gaussian distributions, characterized by their means and co-variance matrices. In this work, we use diagonal, spherical and probabilistic principal component analysis (PPCA) co-variance matrices [11].

The main modeling issue is, then, to determine the best parameters for each distribution. A likelihood function is used, which measures how well each *pdf* fits the data set. Therefore, the best model parameters are the ones that maximize this likelihood function. The maximum likelihood estimation can be carried out iteratively using the Expectation Maximization (EM) algorithm, summarized in Algorithm 2 [12].

```
1. Define the number of classes to use
2. Initialize the model parameters
3. For all the training examples
   a) Expectation stage: compute the a posteriori
      probabilities of the current sample in each class
   b) Maximization stage: adjust parameters towards
      likelihood maximization
```

Algorithm 2. Expectation Maximization algorithm.

In short, in the expectation stage we compute the "expected" classes of all samples, i.e., the *a posteriori* probabilities of each sample in each class. Then, in the maximization step, the parameters are adjusted towards likelihood maximization, given the class membership distributions. The described procedure is repeated for all the training examples.

In the present work, the centers of each cluster, i.e., the Gaussian means, are initialized with the *k*-means clustering algorithm [11]. As for the covariance matrix, all the entries are initially set to 1.

3.3. FEEDFORWARD NEURAL NETWORKS

Artificial Neural Networks (ANN) [13] are computational models that try to emulate the behavior of the human brain. They are based on a set of simple processing elements, highly interconnected, and with a massive parallel structure. ANNs are characterized by their learning, adapting and generalization capabilities, which make them particularly suited for tasks such as function approximation.

Feedforward Neural Networks (FFNN) are a special class of ANNs, in which all the nodes in some layer *l* are connected to all the nodes in layer *l*-1. A FFNN is composed of the input layer, which receives data from the exterior environment, typically one hidden layer (though more layers may be used [14]) and the output layer, which sends data to the exterior environment. In our case, the input layer contains 40 nodes, corresponding to the number of extracted features, and the output layer contains 2, 3 or 5 nodes, according to the classification problem (Section 4). The number of hidden nodes varies from 10 to 30. The actual number is determined experimentally by comparing the results obtained in each structure.

The main advantage of applying ANNs to engineering problems comes from their ability to learn complex input-output mappings by adapting themselves to the data, namely, the weights of the links connecting each pair of neurons. However, their optimal determination is still an open problem, and so, iterative hill-climbing algorithms are used. The main limitation of this approach comes from the fact that only local optima are obtained: only occasionally the global optimum can be found. In the context of ANNs, these iterative optimization algorithms are called training algorithms.

The most widely used training algorithm for FFNNs is backpropagation [13]. Here, there is a forward pass where the inputs are presented to the network and the output values are computed. The error between each target value and the corresponding output value is then calculated. Then, a backward pass is performed, where the weights are adjusted towards error reduction, using the gradient descent method. This process is repeated iteratively until the error is below a given threshold. The procedure described is summarized in Algorithm 3.

The gradient descent method has some limitations regarding convergence properties: the algorithm can get stuck in a local minimum and the selection of the learning rate is usually not trivial (if its value is too low, learning is slow; if it is too high, the network may diverge). Therefore, some variants are used, e.g., learning with a momentum coefficient or defining an adaptive learning rate [13].

Here, we use the Levenberg-Marquardt algorithm, which has the advantage of being significantly faster (10 to 100 times faster [10]) at the cost of higher memory consumption, due to the computation of a Jacobian matrix in each iteration. Also, this algorithm converges in situations where others do not [15].

After the model is trained and validated, using the training and test data respectively, each new sample is classified according to the class associated to the highest output node.

```
1. Define the FFNN structure
   a) number of inputs = number of features
   b) number of hidden nodes
   c) number of outputs = number if classes
2. Initialize the model parameters, i.e., network weights
3. While the error is too high, do for all the training
   samples
   a. Forward stage: compute the error measure
   b. Backward stage: adjust weights according to the
      error measure, using the Levenberg-Marquardt
      algorithm.
```

Algorithm 3. FFNN training algorithm.

4. EXPERIMENTAL RESULTS

As stated before, our goal is to classify classical music into five sub-genres: flute, piano, violin, choral and opera. These can be organized in a hierarchical manner, as depicted in Figure 1. The presented taxonomy is defined only for the sake of clarity: the practical classification performed was not hierarchical.

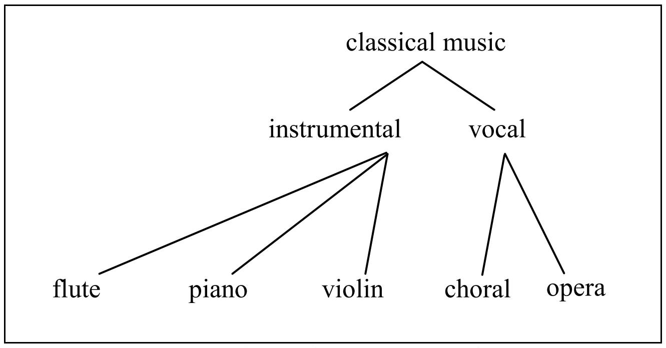

Figure 1. Classical music genre classification.

We collected a database of 300 monaural musical pieces (60 for each genre), sampled at 22050 Hz, with 16 bits quantization. For each piece, 6 seconds' segments were extracted, based on their relevance for its genre, as stated in Section 2.

Our first goal was to discriminate between three genres of instrumental music: music for flute, piano and violin. The 6s' segments extracted were chosen so as to include soles from each instrument by single or several players in unison, in isolation (monophonic segment) or with an orchestra in the background (polyphonic segment). For example, in the case of violin, we extracted a segment from "Spring" in Vivaldi's Four Seasons.

Our second goal was to discriminate between genres of vocal music: chorals and opera. Typically, the musical pieces used for opera were vocal soles, essentially performed by tenors, sopranos and mezzo-sopranos (Callas, Pavarotti, etc.), whereas for choral music segments of simultaneous distinct voices were used, without many of the stylistic effects used in opera (vibrato, tremolo). Many of the used pieces were also *a cappela*, i.e., only human voices, no instruments.

Finally, our third goal was to discriminate between all of the five genres referred above.

For training purposes, we used 40 pieces from each genre, whereas for validation the remaining 20 were used (a total of 200 pieces for training and 100 for validation). Special care was taken so that the training samples for each genre were diverse enough.

Below, we present the classification results obtained using the three followed approaches: *k*-nearest neighbors, GMMs and FFNNs.

4.1. FIRST CLASSIFICATION: THREE GENRES

In this case, musical pieces were classified into flute, piano and violin pieces.

Regarding the KNN classifier (Table 1), we achieved an average classification accuracy of 80%, with $k = 5$. More specifically, we got 80% accuracy for flute, piano and violin. As for the GMM classifier (Table 2), we achieved an average classification accuracy of 75%, using a PPCA co-variance matrix. Detailing the results, we got 65% accuracy for flute pieces, 90% for piano and 70% for violin. The best results were obtained with the FFNN classifier (Table 3), where we achieved an average classification accuracy of 85%, with 20 neurons in the hidden layer. More specifically, we got 90% accuracy for flute, 80% for piano and 85% for violin.

KNN 80%	Flute	Piano	Violin
Flute	80	15	10
Piano	15	80	10
Violin	5	5	80

Table 1. Instrumental music confusion matrix: KNN.

GMM 75%	Flute	Piano	Violin
Flute	65	0	10
Piano	25	90	20
Violin	10	10	70

Table 2. Instrumental music confusion matrix: GMM.

FFNN 85%	Flute	Piano	Violin
Flute	90	10	5
Piano	5	80	10
Violin	5	10	85

Table 3. Instrumental music confusion matrix: FFNN.

Analyzing the classification errors, we noticed that they occur when the instruments are played in an unusual manner, not included in the training samples. For instance, with the FFNN classifier, two violin pieces that were misclassified as piano, had in common the fact of being extremely slow and having small amplitude variations. However, the output values for the violin class were high (above 0.7), which comes from the fact that the timbral features correctly detected the presence of violins.

4.2. SECOND CLASSIFICATION: TWO GENRES

In this situation, musical pieces were classified into opera and choral pieces.

Regarding the KNN classifier (Table 4), we achieved an average classification accuracy of 85%, with $k = 3$. More specifically, we got 100% accuracy for choral pieces and 70% for opera. As for the GMM classifier (Table 5), we achieved an average classification accuracy of 85%, using a spherical co-variance matrix. Detailing the results, we got 80% accuracy for choral pieces and 90% for opera. The best results were obtained with the FFNN classifier (Table 6), where we achieved an average classification accuracy of 90%, with 25 neurons in the hidden layer. More specifically, we got 90% accuracy both for opera and choral pieces.

KNN 85%	Choral	Opera
Choral	100	30
Opera	0	70

Table 4. Vocal music confusion matrix: KNN.

GMM 85%	Choral	Opera
Choral	80	10
Opera	20	90

Table 5. Vocal music confusion matrix: GMM.

FFNN 90%	Choral	Opera
Choral	90	10
Opera	10	90

Table 6. Vocal music confusion matrix: FFNN.

With the FFNN classifier, only two choral pieces and two opera pieces were not correctly classified. One of those choral pieces has some instrumental parts, unlike most of the training samples, which are *a capella*. Also, that particular piece has a female voice that clearly stands out, which could be easily classified as opera by most persons. In what concerns the two mistaken opera pieces, we could not find any clear reasons for that behavior. The only conclusion we can draw is that the used features are good enough for well-behaved cases. For more atypical situations, a more thorough feature analysis is required: elimination of redundant features and/or inclusion of necessary extra features.

4.3. THIRD CLASSIFICATION: FIVE GENRES

Here, musical pieces were classified into the five categories listed before: flute, piano, violin, opera and choral music.

Regarding the KNN classifier (Table 7), we achieved an average classification accuracy of 67%, with $k = 5$. More specifically, we got 80% accuracy for flute, 75% for piano and 50% for violin, 65% for choral pieces and 65% for opera. As for the GMM classifier (Table 8), we achieved an average classification accuracy of only 53%, using a diagonal co-variance matrix. Detailing the results, we got 55% accuracy for flute, 70% for piano, 60% for violin, 50% for choral pieces and 30% for opera. In this situation, the best results were obtained, by far, with the FFNN classifier (Table 9), where we achieved an average classification accuracy of 76%, with 30 neurons in the hidden layer. More specifically, we got 75% accuracy for flute pieces, 65% for piano, 85% for violin, 75% for chorals and 80% for opera.

KNN 67%	Flute	Piano	Violin	Choral	Opera
Flute	80	10	10	20	10
Piano	15	75	0	15	0
Violin	5	5	50	0	10
Choral	0	5	20	65	15
Opera	0	5	20	0	65

Table 7. Mixed classification confusion matrix: KNN.

GMM 53%	Flute	Piano	Violin	Choral	Opera
Flute	55	15	10	30	15
Piano	0	70	0	0	0
Violin	10	5	60	10	40
Choral	30	10	10	50	15
Opera	5	0	20	10	30

Table 8. Mixed classification confusion matrix: GMM.

FFNN 76%	Flute	Piano	Violin	Choral	Opera
Flute	75	20	0	10	10
Piano	5	65	0	15	5
Violin	0	5	85	0	0
Choral	10	5	10	75	5
Opera	10	5	5	0	80

Table 9. Mixed classification confusion matrix: FFNN.

Though interesting, the results obtained for this more complex classification problem are less satisfactory. It is clear that the used features could not separate the five classes in a totally unambiguous manner. Therefore, a deeper feature analysis seems fundamental in order to obtain better results.

5. CONCLUSIONS

The main goal of this paper was to present a methodology and a comparative study for the classification of classical music. Although the results obtained are not sufficient for real-world applications, they are promising.

In the most complex case, where we defined five categories, the classification results were less accurate, particularly for KNN and GMM classifiers. Neural networks always lead to better results, especially in the most complex case. In our opinion, a hierarchical classifier, following the structure in Figure 1, would lead to better results.

In the future, we will conduct a more thorough analysis of the feature space: detection and elimination of redundant features, as well as definition and utilization of other features, which may help to discriminate the more atypical cases. Additionally, we plan to use a broader and deeper set of categories, i.e., more basis classes and subclasses. In case we use categories like waltz, rhythmic features, not used in the present work, will certainly be important.

ACKNOWLEDGMENTS

This work was partially supported by the Portuguese Ministry of Science and Technology (MCT), under the program PRAXIS XXI.

REFERENCES

[1] G. Tzanetakis & P. Cook, Musical Genre Classification of Audio Signals, *IEEE Transactions on Speech and Audio Processing*, 10(5), 2002, 293-302.

[2] S. Golub, *Classifying Recorded Music* (MSc Thesis, University of Edinburgh, 2000).

[3] K. Kosina, *Music Genre Recognition* (MSc Thesis, Hagenberg, 2002).

[4] K.D. Martin, Toward Automatic Sound Source Recognition: Identifying Musical Instruments, *NATO Computational Hearing Advanced Study Institute*, Italy, 1998.

[5] G. Tzanetakis, G. Essl & P. Cook, Automatic Musical Genre Classification of Audio Signals, *Proc. of 2nd ISMIR*, 2001.

[6] D. Perrot & R.O. Gjerdigen, Scanning the dial: An exploration of factors in the identification of musical style, *Proc of the 1999 Society for Music Perception and Cognition*, 1999.

[7] K.D. Martin, E.D. Scheirer & B.L. Vercoe, Musical content analysis through models of audition, *Proc of the ACM Multimedia Workshop on Content-Based Processing of Music*, 1998.

[8] S. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing* (California Technical Publishing, 1997).

[9] R. Polikar, *The Wavelet Tutorial*, (http://engineering.rowan.edu/~polikar/WAVELETS/ WTtutorial.html, available by July 2003).

[10] H. Demuth & M. Beale, *Neural Network Toolbox User's Guide - version 4* (Mathworks, 2001).

[11] R. Duda, P. Hart & D. Stark, *Pattern Classification*, (John Wiley & Sons, New York, 2000).

[12] T.K. Moon, The Expectation-Maximization Algorithm, *IEEE Signal Processing Magazine*, 13 (6), 1996, 47-60.

[13] S. Haykin, *Neural Networks: A Comprehensive Foundation*, (Macmillan College Publishing, 1994).

[14] W. Sarle, *Neural Nets FAQ*, (ftp://ftp.sas.com/pub/neural/ FAQ3.html, 2001)

[15] M. Hagan & M. Menhaj, Training Feedforward Networks with the Marquardt Algorithm, *IEEE Transactions on Neural Networks*, 5(6), 1994.