Structure Recognition on Sequences with a Neuro-Fuzzy System

Klaus Dalinghaus
Institute of Cognitive Science
University of Osnabrück
kdaling@uos.de

Tillman Weyde
Research Department of Music and Media Technology
University of Osnabrück
tweyde@uos.de

Abstract

We present a neuro-fuzzy system capable of recognizing and assigning rhythmic structure to musical input. The system is thereby based on two stages, grouping of the raw data and alignment of simple groups. Both steps incorporate expert knowledge from the musical domain. Furthermore, they are optimized with learning algorithms from the neural network domain according to training data collected from real-life scenarios. The performance of the system reaches an accuracy of 97%.

Keywords: Neural Networks, Fuzzy-Logic, Music, Alignment of Rhythmic Data.

1 Introduction

Many problems dealing with the alignment of sequences can be solved with the minimal edit distance algorithm or minimizing another well defined cost function on sequences, such as comparison of words when searching documents or alignment of DNA sequences (see [4] and [1]). But for some tasks it is difficult or impossible to find such a cost function, e.g., if the knowledge about the domain is incomplete or vague. This is the case if MIDI[1] data streams are to be compared with respect to their underlying rhythmic pattern. As an example the tempo may vary within a sequence just because of the difficulty of the part which is to be played. However, both, expert knowledge partially describing the system and training data for correctly aligned rhythmic patterns are available, such that a neuro-fuzzy system seems an appropriate tool for this task. The interaction of fuzzy systems and neural networks has grown in the last years because the approach can use the advantages of both systems and decrease the disadvantages of them (see [6], [3], [7]). It is possible to put expert knowledge into the system, to train the system according to given training examples, and, after the training process, to interpret the results. We here propose a neuro-fuzzy system for the alignment of rhythmic structure. Thereby, the system works on two different levels, the *structure level* and the *group level*. That means the first step is to look for a segmentation of the sequences into groups and then, in the second step only single groups are aligned. Due to this separation, a comparably efficient alignment procedure can be used. Moreover, expert knowledge, i.e. vague rules for both parts, segmentation of a sequence and comparison of two segments, can be integrated as fuzzy rules into the system. The interpreted system has been realized based on musical theory and successfully trained real data sets.

[1] Musical Instrument Digital Interface

2 Recognition of Rhythmic Patterns With a Neuro-Fuzzy System

The task of recognizing and assigning rhythmic structure to unquantized musical input is fundamental for developing educational systems and database search on music. Music theory and music psychology have determined features that are of importance in the perception and cognition of music, but a coherent paradigm to support computer models has not yet been established. We now present the precise goal and the neuro-fuzzy system (see [9]). Thereby, the used rules can be seen as a partial implementation of

the grouping rules by Lerdahl and Jackendoff [5] with some rules added for the comparison.

2.1 The Basic Problem

Rhythms are represented as sequences of note objects based on MIDI data. Three values of these note objects are used as input for the system: onset time, duration, and key velocity (loudness). The task is to find an appropriate alignment for two given sequences (task and input). Thereby, appropriate alignments assign parts of similar rhythmic structure to each other. A precise characterization of a good alignment is however not available, and an alignment can be best evaluated based on the two levels mentioned before: the structure level and the group level. First, both sequences are segmented into groups, so called motifs. This grouping depends on the context and on temporal proximity. Additional criteria are the accents on the notes, the group length (amount of notes), and the group duration (amount of time). Motifs are spontaneously recognized, even if they are not equal. They are 'transposable' in respect to time, dynamics and pitch, i.e. their Gestalt is partially invariant to these transformations.

Now the assignment can be done on both levels. Every group of the input is assigned to a task group or to no group. In the latter case an error value for the input group is calculated. Then the assignment of the notes inside the groups can be done. Because the assigned groups need not have the same number of notes, some notes cannot be assigned. These are marked as added or omitted. We get a so called *interpretation* (see figure 1), where information about the matching of the groups is stored, which includes tempo, loudness, added or omitted notes, etc.

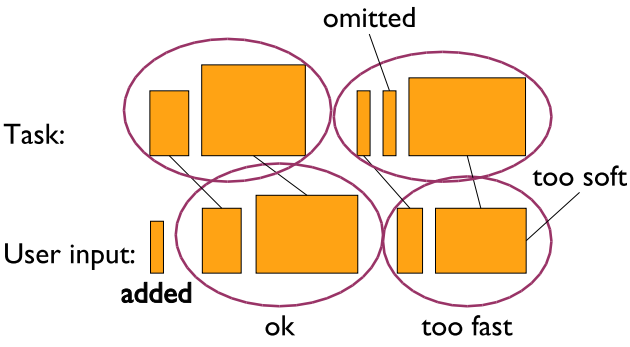

Figure 1: Interpretation of a given example

As next step complex features can be extracted from an interpretation for every group assignment. Four main features are extracted: correctness, tempo, precision, and position. The calculation also determines deviation of the tempo, of the note positions, of the loudness patterns, and the deviation of the groups from their expected positions. In addition some input features on the structure level are calculated, e.g. a value for the correct order of the groups in the input.

2.2 Description With Fuzzy-Logic

Based on the calculated features a rule set can be created which is able to rate a given interpretation. Here is a subset of the rules used in the system:

$$
\begin{aligned}
\mathrm{CQual}(sa) &\leftarrow \mathrm{CTpoQual}(sa) \wedge \mathrm{CPrcsn}(sa) \wedge \\
&\quad \mathrm{CCorrect}(sa) \wedge \mathrm{CPosition}(sa) \\
\mathrm{CTpoQual}(sa) &\leftarrow \mathrm{GTpoQual}(ga_1) \wedge \ldots \wedge \mathrm{GTpoQual}(ga_n), \\
&\quad \text{where } sa = [ga_1, \ldots, ga_n] \\
\mathrm{GTpoQual}(ga_i) &\leftarrow \mathrm{GTpoStbl}(ga_i) \wedge \mathrm{GTpoPlsbl}(ga_i) \\
\mathrm{CPrcsn}(sa) &\leftarrow \mathrm{GPrcsn}(ga_1) \wedge \ldots \wedge \mathrm{GPrcsn}(ga_n), \\
&\quad \text{where } sa = [ga_1, \ldots, ga_n] \\
\mathrm{CCorrect}(sa) &\leftarrow \mathrm{GCorrect}(ga_1) \wedge \ldots \wedge \mathrm{GCorrect}(ga_n), \\
&\quad \text{where } sa = [ga_1, \ldots, ga_n]
\end{aligned}
$$

The first rule says that the quality of a structure assignment (*sa*) depends on the tempo quality, the precision, the correctness, and the positioning of *sa*. The second, fourth, and fifth rule use a special feature which can deal with a variable number of premises. This is necessary to get from the structure level (*sa*) to the group level (*ga*) because the number of groups in an interpretation might vary.

Often t-norms and t-conorms like $\top_{min}$ and $\bot_{min}$ are used to calculate the truth value of logical expressions in fuzzy logic. However, we decided to use a compensatory operator:

$$\mu_{\otimes,q}(\alpha_1, \ldots, \alpha_n) = \left(\frac{1}{n} \sum_{i=1}^{n} \alpha_i^q \right)^{\frac{1}{q}} \qquad q > 0 \qquad (1)$$

This operator is related to the Yager operator (see [10]). It converges to $\top_{min}$ in the limit of $q \to 0$ and to $\bot_{min}$ in the limit of $q \to \infty$, so it is a compensation between both operators. It is not a t-norm because it violates the requirement of the identity and associativity. It can nevertheless be used in a neuro-fuzzy

system where only differentiability of the operator is required. In our system we use $q = \frac{1}{2}$ for the conjunction and $q = 2$ for the disjunction. The compensatory operator give more robust and less extreme values, for example, if only one α_i is 0 or 1.

2.3 Transformation to a Neuro-Fuzzy Network

Using the algorithm in [6] we can transform the rule basis into a neural network. This is done by assigning a neuron to every proposition and to every conclusion. For every rule, connections from the premises to the conclusion are inserted into the network. The neurons are calculating their input no longer as the weighted sum but as the evaluation of the fuzzy operator used in the corresponding rules. Note that the algorithm as described in [6] can only deal with limited rule sets which, in particular, need to be non-recursive. For our rule set, transformation into a form suitable for [6] is possible.

For training of network the standard backpropagation algorithm has to be extended because the neurons are no longer using the weighted sum to calculate their input value. The modified backpropagation rule, which is obtained as gradient decent on the quadratic error on a given training set, can be written as

$$\triangle w_{ij} = \eta \sum_{p \in P} o_i^{(p)} \cdot drvt_{ij}^{(p)} \cdot \delta_j^{(p)} \tag{2}$$

with

$$\delta_j^{(p)} = \begin{cases} f'(net_j^{(p)})(t_j^{(p)} - o_j^{(p)}), & \text{if } j \text{ is outp. neur.,} \\ f'(net_j^{(p)}) \sum_{s=1}^{m} \delta_{k_s}^{(p)} \cdot w_{jk_s} \cdot drvt_{jk_s}^{(p)}, & \text{else} \end{cases} \tag{3}$$

where $drvt_{ij}^{(p)}$ is the derivation of the used input function for neuron j given input pattern p, $o_i^{(p)}$ is the output of neuron i for pattern p, $t_j^{(p)}$ is the target value for pattern p and neuron j, w_{ij} denote the weight from i to j in the network, and f denote the activation function of the neuron. The network is trained with a modification of a simple gradient descent according to the ideas provided in the R-PROP algorithm (see [8]).

2.4 Integration of the Network into the System

Since the rule set enables the system to rate an interpretation, a preprocessing function is needed, that calculates all possible interpretations for a given pair of sequences. After this step, all implausible interpretations are filtered out. The remaining of the interpretations are sorted by a heuristic and complex features are computed for them. The neuro-fuzzy system rates every interpretation based on these features and the one rated best is selected as output.

2.5 Choice of the Training Set

For the training process examples i.e. interpretations of pairs of sequences together with a given rating are needed. Yet it is very difficult to choose an appropriate exact value for a given interpretation. However, it can be stated very easily which of two given interpretations should be rated higher. So we used a training approach based on relative training examples as proposed in [2] (see figure 2).

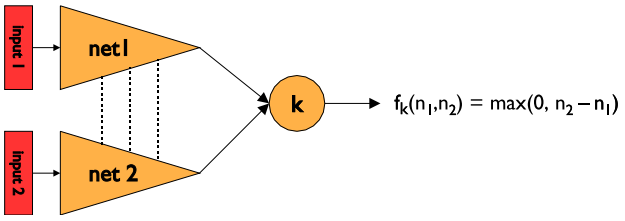

Figure 2: Training with relative examples

Thereby, a training example for the neural network consists of a pair of two different interpretations of the same input and task sequence: one given from an expert and one from the system. The output of the network shall be 0 if the expert ranking is larger than the ranking from the system, and it shall be the difference of the two rankings otherwise. The motivation behind is, that then a constant output 0 would correspond to a perfectly trained system. To handle this procedure we duplicate the network and create a so called comparator neuron k. The two networks use weight sharing.

Now the training process runs as follows: First an expert selects for every example (pair of sequences) the best interpretation. Then for every example the interpretation rated best by the current system is calculated to build the relative training examples. To guarantee that the training algorithm works correctly, we iterate the training process, because even if the expert interpretation is rated better than the system interpretation after training, there could be a third interpretation now rated best by the system.

2.6 Results

For testing the system we used 100 samples consisting of 50 samples as performed by students in original form and with noise added. The expert interpretations were defined by graduate students. Three types of networks were trained with these data: a linear network[2], a multi-layer perceptron[3] and the neuro-fuzzy network. With the linear network only for 28% of the samples the estimated interpretation was rated best. For the other networks 80% of the samples were processed correctly. But even if the system output is not the expected interpretation, it can nevertheless be a musically acceptable which is the case for another 17% of the samples. Overall 97% of the interpretations chosen by the system trained with the multi-layer perceptron or the neuro-fuzzy network are musically acceptable, which is a good result. But it is not possible to extract any knowledge from the calculated weights of the multi-layer perceptron wich is the case for the neuro-fuzzy network.

3 Conclusions

We have proposed a system to solve the task of segmenting rhythmic patterns and it has been solved satisfactorily. Thereby, knowledge modelling in combination with machine learning tools enables us to solve the task without a complete model or exact knowledge of perceptual and other relevant processes. The concept of the presented system is general such that it should be possible to transfer the design to other fields of application: the alignment of phonemes and graphemes in orthography for example. A transfer to another ares as well as automatization of rule acquisition will be the topic of future research.

References

[1] Abdullah N. Arslan and Ömer Egecioglu. Efficient algorithms for normalized edit distance. *Journal of Discrete Algorithms*, 1(1):3–20, 2000.

[2] Heinrich Braun. *Neuronale Netze: Optimierung durch Lernen und Evolution*. Springer Verlag, Berlin, Heidelberg, 1997.

[3] Giovanna Castellanoa, Ciro Castiello, and Anna Maria Fanelli. KERNEL: A matlab toolbox for knowledge extraction and refinement by neural learning. In *Lecture Notes in Computer Science*, volume 2329, pages 970–979. Springer Verlag, Berlin, 2002.

[4] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences : Computer Science and Computational Biology*. Univ. Press, Camebridge, 1999.

[5] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Camebridge, MA, 1983.

[6] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Foundations of Neuro-Fuzzy Systems*. Wiley, Chichester, 1997.

[7] Detlef Nauck and Rudolf Kruse. NEFCLASS – a neuro–fuzzy approach for the classification of data. In K.M. George, Janice H. Carrol, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Applied Computing 1995. Proc. of the 1995 ACM Symposium on Applied Computing, Nashville, Feb. 26–28*, pages 461–465. ACM Press, New York, 1995.

[8] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *ICNN-93: IEEE International Conference of Neural Networks*, pages 586–591, San Francisco, CA, 1993.

[9] Tillman Weyde und Klaus Dalinghaus. Recognition of musical rhythm patterns based on a neuro-fuzzy-system. In *Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining and Complex Systems*, volume 11, pages 679–684, New York, NY, 2001. ASME press.

[10] Ronald R. Yager. On a general class of fuzzy connectives. *Fuzzy Sets and Systems*, 4:235–242, 1980.

[2] The input features responsible for the group quality are linearly connected to a single neuron. The same is done for the segmentation quality. These two neurons and other input features constitute the top layer and are linearly connected to the output neuron

[3] In addition to the structure in the linear network, a fully connected hidden layer is put in between the top layer and the output neuron.