Charting Perceptual Spaces with Fuzzy Rules

Iván Paz
Soft Computing research group at Intelligent Data Science and Artificial Intelligence Research Center
Universitat Politècnica de Catalunya-BarcelonaTech
Barcelona, Spain
ivanpaz@cs.upc.edu

Àngela Nebot
Soft Computing research group at Intelligent Data Science and Artificial Intelligence Research Center
Universitat Politècnica de Catalunya-BarcelonaTech
Barcelona, Spain
angela@cs.upc.edu

Enrique Romero
Soft Computing research group at Intelligent Data Science and Artificial Intelligence Research Center
Universitat Politècnica de Catalunya-BarcelonaTech
Barcelona, Spain
eromero@cs.upc.edu

Francisco Mugica
Soft Computing research group at Intelligent Data Science and Artificial Intelligence Research Center
Universitat Politècnica de Catalunya-BarcelonaTech
Barcelona, Spain
fmugica@cs.upc.edu

***Abstract*—Algorithmic music nowadays performs domain specific tasks for which classical algorithms do not offer optimal solutions or require user's expertise. Among these tasks is the extraction of models from data that offer an understanding of the underlying behavior, providing a quick and easy to use way to explore the data for first (sometimes on-the-fly) insights. Learning rules from examples is an approach often used to achieve this goal. However, together with the aforementioned requirements algorithmic composition needs to create new material so that it is perceived as consistent with the material of the data. In addition, the input data sets are usually small because the human is the bottleneck when generating them. In this contribution we present a fuzzy rule induction algorithm focused on generalizing a set of data, complying with the previous requirements, that offers good results for small data sets. For its evaluation -in a field where there are no benchmarks available - data sets obtained during user tests were used. The visual representation offered by the fuzzy chart helps to reduce the cognitive complexity of the devices used in algorithmic music. The results obtained show that this approach is promising for future developments.**

***Index Terms*—fuzzy rules, algorithmic music, perceptual spaces, classification, parametric devices**

I. Introduction

Algorithmic composition experienced a rebirth from the moment when laptops were powerful enough to run programs that allow changing them as they run [1], [2]. Accompanying this process, new needs inherent of the application domain arose. For example, in sound design (crafting sound with specific characteristics using tools such as synthesizers, audio processors, generative algorithms, etc.) interpretable models that show the relationships between devices' parameter values and the produced sound are required [3]. In live coding (live interaction with code to produce music or visuals [4]) quick and easy to use ways to explore data (produced by algorithms) for first insights are required. Although learning rules from examples is an approach often used to create interpretable models, it has not been used whitin these contexts. This is due to the fact that many algorithms require user's expertise, or produce poor results for small datasets (for example approaches to extract fuzzy rules from clusters [5]). Other complicated structures offer greater flexibility but are often computationally expensive [6], [7]. Algorithms that offer good results for small data depend on the order of the input variables, and require prior knowledge of the explored system to give optimal results [8], [9]. On the other hand, rule learning algorithms have been little explored lately, in part by the focus on results offered by algorithms such as neural networks [10], which nevertheless produce good results, are not very interpretable.

In this contribution we present an algorithm that extracts fuzzy rules by working on the top of the rule learning algorithm [11] "RuLer" (Section II-A). It extracts interpretable models from data that generalize the input and cover the whole input space (unlike the RuLer), also offering good results for small data sets. The algorithm works on combinations of parameters that, when applied to a device, produce an output to which a perceptual label is associated. Thus, the result can be seen as a chart of the perceptual space. To evaluate the model, user tests were carried out since there are no benchmarks available. This is not a lack of data but rather a restriction inherent to the field. For example, in sound design parameter devices are tweaked while searching for interesting sounds. As discussed in [3], a problem when creating sound is the diversity of objectives: "How does one codify what is beautiful, good, or suitable (for the context)? Maybe some ugly music is wanted to provide contrast, and at another time something beautiful (whatever that means)."

The rest of the paper is structured as follows: Section II presents the algorithm, Section III presents the user tests, and Section IV discusses the results and present the conclusions together with some possible paths for further research.

This research was supported by the National Council of Science and Technology of Mexico (CONACyT) 240514, and project TIN2016-79576-R.

II. Algorithm

The algorithm presented here extracts fuzzy rules from logical if-then rules obtained by the "RuLer" rule learning algorithm [11]. A brief description of "RuLer" is presented in Section II-A. Starting from the rules derived from this algorithm, in this investigation we extend these linguistic rules in fuzzy rules by means of the conformation of the cores and the supports of said rules. It first learns the cores of the fuzzy rules from the input, and then adjusts their supports. These processes are explained in Sections II-B1 and II-B2.

A. Rule Learning Algorithm "RuLer"

The rule learning algorithm "RuLer" extracts logical if-then rules from data. Each input datum is a combination of values that, when applied to the parameters of a particular device, produce sound selected by the user for a specific musical context. A linguistic label that describes the characteristics of the sound (e.g. calm, harsh) or the musical context in which the combination is used (e.g. intro, break), is assigned to each combination. The algorithm works as follows:

1) The algorithm represents each categorized parameter combination as an array of size N. The first $N-1$ entries contain the set of possible values of the corresponding parameter, and the last entry contains the category (or class) assigned to the combination. Each datum is considered a single rule. For example, a rule r = [{3}, {5}, intro] is a single rule. A rule r = [{1,2,3}, {7}, . . . , {3}, intro] sould be interpreted as r[1] = 1 OR 2 OR 3, r[2] = 7, . . . , label = intro. The algorithm iteratively searches for patterns among the rules. It iterates until no new rules are created. This is done in the following way:
   a) Take the first rule of the rule set.
   b) Compare the selected rule with the other rules using the dissimilarity function (Section II-A1). If a pattern is found, i.e the dissimilarity between the two rules is less or equal that a threshod established by the coder, create a new rule using the *create_rule* function (Section II-A2).
   c) Eliminate the redundant rules from the current set. A rule r_1 is redundant with respect to a rule r_2 (of the same class) if $\forall$ $i \in$ {0, . . . N-1}, $r_1[i]$ is a subset of $r_2[i]$. Note that this eliminates those rules from which the new rules were created.
2) Add the created rules at the end of the rule set.
3) It solves the contradictions following a maximum volume heuristic.

1) dissimilarity Function: The *disimilarity* function receives two rules (r_1, r_2) together with a threshold $d \in \mathbf{N}$ and returns True in case the rules have the same category and *dissimilarity*$(r_1, r_2) \leq d$. It returns False otherwhise.

The *dissimilarity* function used counts the number of empty intersections between the corresponding entries in the rules. For example, if r_1 = [{1}, {3,5}, intro] and r_2 = [{1,3}, {7,11}, intro], *dissimilarity*(r_1,r_2) = 1.

2) create_rule Function: This function verifies that no contradictions (i.e rules with the same parameter values but different label) are created during the generalization process. It also allows the user to control the generalization level by defining a percentage of the cases contained in any created rule that should be present in the original data. The function receives two rules and creates a new rule if both conditions are met. The function used creates a new rule by taking the unions of the corresponding sets of the rules received.

3) Domain Specific Functions: Note that the *dissimilarity* and *create_rule* finctions can be changed according to the compared objects and the desired generalization. For example, for harmonic objects, we probably want to use a dissimilarity that looks at the harmonic content. For rhythms, temporal factors need to be addressed. See for example [12] for a comparison of rhythmic similarity measures. However, the dissimilarity presented in Section II-A1 works well for the objects considered in the performed experiment (Section III).

B. Fuzzy Rules

From the logical rules, the cores and supports of the fuzzy rules are determined by means of the *Building cores* and *Building supports* functions, respectively, that are explained below.

1) Building cores: The cores of the fuzzy rules are build by extending the sets contained at the entries of the rules to intervals between its respective maximum and minumim values. For example the rule r_1 = [{1}, {3,5}, intro] wold be r_1 = [[1], [3, 5], intro], and will include all the values in the intervals [1] (only one value) and [3,5].

In order to do this, it is necessary to solve the contradictions produced by the possible intersections between intervals that can be created as a result of this process. To solve the contradictions, the algorithm identifies the groups of rules that intersect. Two rules r_1 and r_2 intersect if for all i there exists x in $r_1[i]$ such that $y_1 \leq x \leq y_2$ with y_1, $y_2 \in r_2[i]$. If two rules with different class intersect, it is enough to "break" one parameter to solve the contradiction. For example, the contradiction between the rules (see Fig. 1):

$$r_1 = [\{1,5\}, \{2,4\}, calm]$$

and

$$r_2 = [\{2\}, \{3\}, harsh]$$

can be solved either as (see Fig. 2 top):

$$r_{1a} = [\{1\}, \{2,4\}, calm]$$

$$r_2 = [\{2\}, \{3\}, harsh]$$

$$r_{1b} = [\{5\}, \{2,4\}, calm]$$

or as (see Fig. 2 bottom):

$$r_{1a} = [\{1,5\},\{2\},calm]$$

$$r_2 = [\{2\},\{13\},harsh]$$

$$r_{1b} = [\{1,5\},\{4\},calm]$$

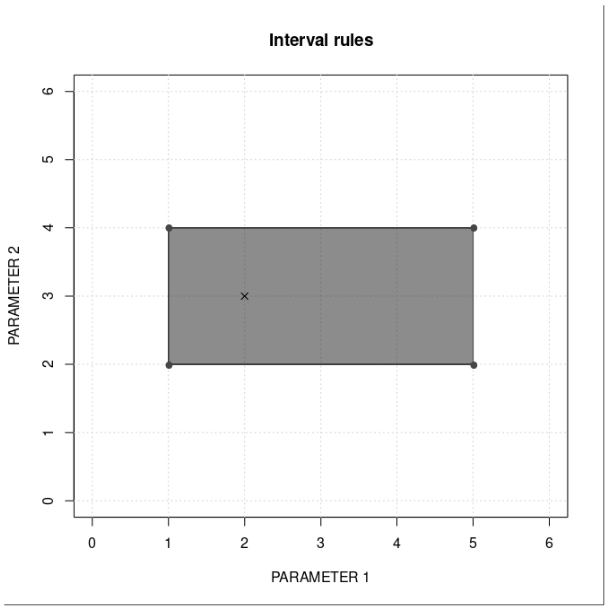

Fig. 1. Rule [{2}, {3}, harsh] intersects rule [{1,5}, {2,4}, calm].

To select the partition, the volume of each set of rules is calculated and the one with maximum volume is selected.

Volume of a Rule Set: The volume of a single rule has two components: Volume (V) and dimension, defined as in Equation 1.

$$V = \sum_{i=0}^{N-1} V_i, \text{ where } V_i = |max_i - min_i| \qquad (1)$$
$$\text{dimension = Number of } V_i \text{ for which Vi} \neq 0.$$

In Equation (1), for each entrance i in the rules, the absolute value between the maximum an minimum values of the set is calculated. For example, if some i has $\{$11,13,15$\}$, V_i = 4, that is $|15 - 11|$. If it has $\{3\}$ V_i = 0. The volume of a set of rules collects the volumes of the individual rules adding those who have the same dimension. It is expressed as an array containing the volume for each dimension. See Table I for an example. When two volumes are compared the greatest dimension wins. For example (Volume = 1, dimension 2) $>$ (Volume = 4, dimension 1). In the same way (Volume = 1, dimension 3) $>$ (Volume = 100, dimension 2; Volume = 100, dimension 1).

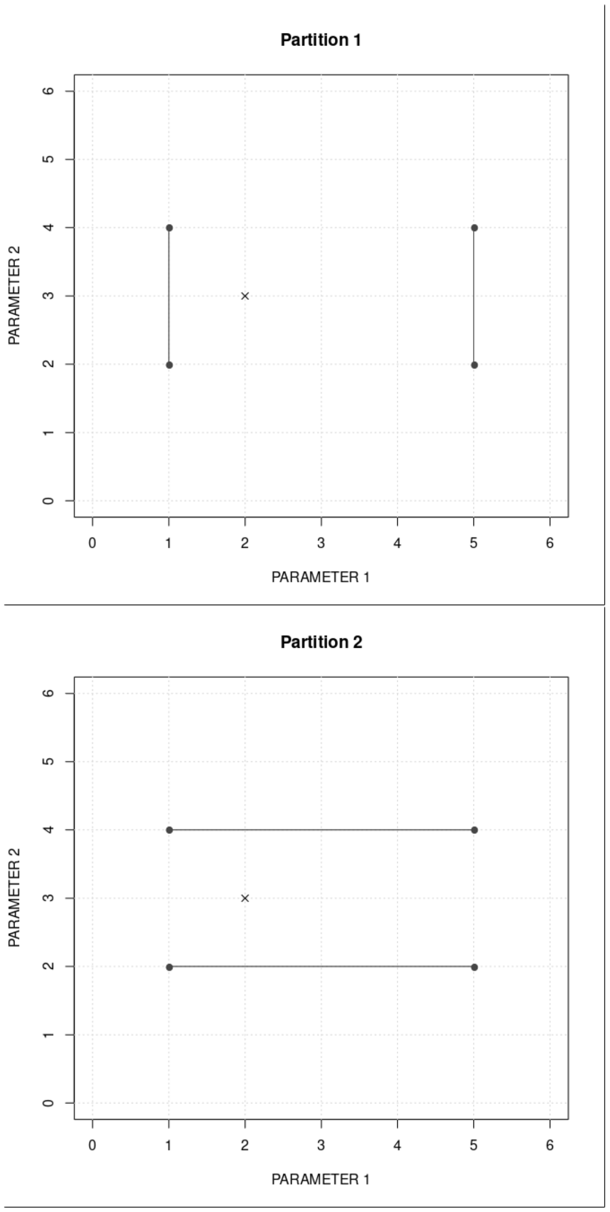

Fig. 2. Two possible ways of solving the contradiction that appears in Fig. 1.

TABLE I
EXAMPLE OF VOLUME AND DIMENSION FOR A SET OF RULES

Rules and Volumes	**Parameter values and category**		
	parameter 1	***parameter 2***	***category***
rule r_1	{1,3}	{2}	calm
Volume r_1	V_1=2	V_2 = 0	Volume 2, dimension 1
rule r_2	{5}	{1,3}	harsh
Volume r_2	V_1=0	V_2 = 2	Volume 2, dimension 1
rule r_3	{5,7}	{5,7}	calm
Volume r_3	V_1=2	V_2 = 2	Volume 4, dimension 2
Global volume: volume 4, dimension 2; volume 4, dimension 1			

[a]Note that rules with different categories contribute to the global volume.

2) Building supports: To build the supports of the fuzzy rules, trapezoidal membership functions are used. Consider the trapezoidal membership function of Equation (2) drawn in Figure 3.

$$\mu_{k,i}(p_i) = \frac{1}{2}[max(0, 1 - max(0, \gamma * min(1, p_i - w_{k,i}))) + max(0, 1 - max(0, \gamma * min(1, v_{k,i} - p_i)))] \tag{2}$$

In this function the index k represents the rule number and the index i counts the parameters. The values $v_{k,i}$ and $w_{k,i}$ are, respectively, the minimum and maximum values of the k^{th} rule i^{th} parameter. The parameter γ controls the "slope" of the trapezoid. Then, for each rule r_k, a trapezoidal membership function is assigned to each parameter p_i in the following way:

1) If the parameter contains an interval, let us say $[a - b]$, then $v_{k,i}$ = a and $w_{k,i}$ = b. The trapezoidal membership function has maximum membership values between the extremes of the interval.
2) If the parameter contains a single value, then $v_{k,i} = w_{k,i}$. Thus, we obtain a triangular membership function centered at that value.

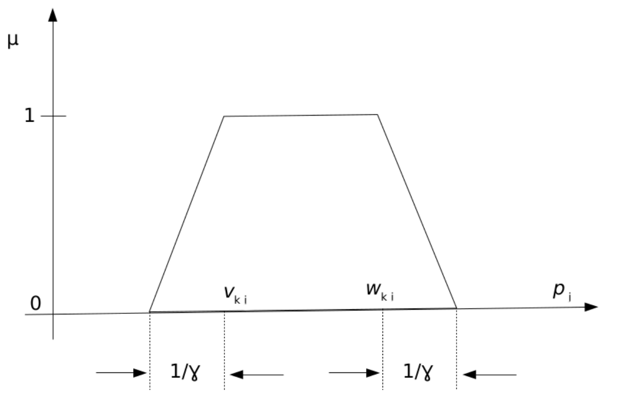

Fig. 3. Membership function for the classifier. The values $v_{k,i}$ and $w_{k,i}$ are, respectively, the minimum and maximum values of parameter i in rule k. The parameter γ controls the "slopes".

C. Fuzzy Classifier

To test the extracted models a fuzzy classifier was built. For that purpose, all features are scaled into $[0, 1]$. To classify a new parameter combination proceed as follows: Let P be the combination sent to the classifier. P is a combination of p_i parameter values. Then, for each rule r_k, it calculates the membership values $\mu_{k,i}(p_i)$ for each parameter p_i. The firing strength $\tau_k(P)$, defined in [13], of a rule r_k, which measures the degree to which the rule matches the input parameters, is defined as the minimum of all the membership values obtained for the parameters (see Equation (3)), i.e:

$$\tau_k(P) = min\,\{\,\mu_{k,i}(p_i)\,\} \tag{3}$$

Once the firing strength has been calculated for all rules, the assigned class (i.e. category; Equation (4)) will be equal to the class of the rule with maximum firing strength.

$$Class(P) = \text{Class of } R_c \text{ where C} = \underset{k}{arg\,max}\,\{\tau_k(P)\} \tag{4}$$

An example of the classification process for a hypothetical system with only two rules and two parameters is shown in Figure 4. The parameter combination $P = (p_1, p_2)$ is sent to the classifer. To classify this combination, for each rule, the degree of membership of each p_i given by the i^{th} membership function of the rule is calculated and the minimum of these values is taken. Then, a set containing the minimum value of each rule is formed. Finally, the output of the system is the class associated with the rule corresponding to the maximum value of that set. If more than one rule has the same firing strength, a random decision is taken.

III. Model evaluation

As mentioned in section I, there are no benchmarks available in this field, since the parameter combinations that codify what is beautiful, good, or suitable depend on the specific context. Therefore, to evaluate the extracted models the data generated during the user test described in [14], was used. It is available at [15]. The device explored was a single band impulse oscillator. In the data collection, 10 students of the Real Time Interaction class of the Master's Degree in Sound and Music Computing of the Pompeu Fabra University (winter term of 2016-2017) were involved.

Single Band Limited Impulse Oscillator: The single band limited impulse oscillator "Blip", produces a fundamental frequency to which a certain number of upper harmonics, all with equal amplitude, are added [16]. It is available as a unit generator (UGen) in the SuperCollider programing language. It has three parameters: *freq*, the fundamental frequency; *numharm* which controls the number of upper harmonics added; and *amp*, the output's amplitude. The parameters can range in the following way: the fundamental frequency can take values between 0Hz and 20KHz which is the upper audible limit. The number of added harmonics range from 0, which leads to a pure tone frequency of the chosen fundamental, to any (theoretical) number of harmonics. However, considering the aliasing processes, this number is limited by the sample rate and the chosen fundamental frequency. Finally, the signal amplitude is normalized between 0 and 1.

Configuration for the Experiments: For the experiments the values were restricted within the following ranges. The frequency took values in the interval $[0Hz, 400Hz]$, the number of upper harmonics in $[0, 100]$, and the amplitude in $[0, 1]$. This is a bounded space where the perceptual properties (described next) are clearly expressed. Although this is a simple generator, the different aural possibilities produced by its parameter combinations clearly define distinct perceptual subspaces. For the experiment, three perceptual properties were used. These are outputs perceived as **rhythmic**, **rough**,

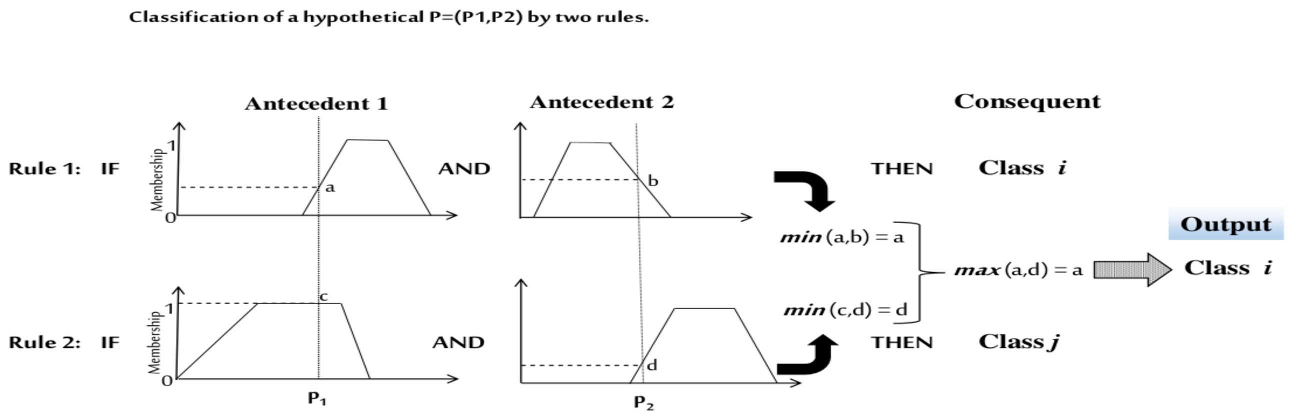

Fig. 4. Example of the classification process for a system of two rules with two parameters p_1 and p_2. A new combination $P = (p_1, p_2)$ is sent to the classifier. For the first rule $\mu(p_1) = a$ and $\mu(p_2) = b$. The minimum of these values is a. In the case of the second rule $\mu(p_1) = c$, $\mu(p_2) = d$ and $min(c, d) = d$. Finally, the $max(a, d) = a$ and therefore the class assigned to the instance is $Class\ i$, which is the class associated with the first rule.

and **pure-tone**. For further reference of the Blip generator, the reader is referred to [17].

Perception Limits of the Perceptual Properties: Generally speaking, outputs perceived as rhythmic are characterized by low fundamental frequencies (less than 20Hz). The number of upper harmonics controls the pitch of the produced pulse (or beat). A low number of harmonics produce constant beats with low pitch, while a higher number of upper harmonics produce the same constant beating, but with higher pitch. Outputs perceived as rough have fundamental frequency values between 15 and 35 Hz. The number of upper harmonics plays a similar role as in the case of rhythmic outputs.

Finally, outputs perceived as pure-tones are produced by any frequency greater that 20Hz without upper harmonics. However, combinations of low-mid to high frequencies (e.g greater than 80Hz) with medium to high number of upper harmonics (for example, between 40 and 50) can sometimes be perceived as pure-tones. This phenomenon is accentuated due to the effects of comparison on the perception. For example, when hearing something highly rough and afterwards hearing a slightly rough combination, the latter can be perceived more like a pure tone. Also, it is worth saying that the regions described are not crisp, and the perception may vary from one user to another, as well as with the just mentioned *perception by comparison* effect. A graph with the collected data is shown in Fig. 5

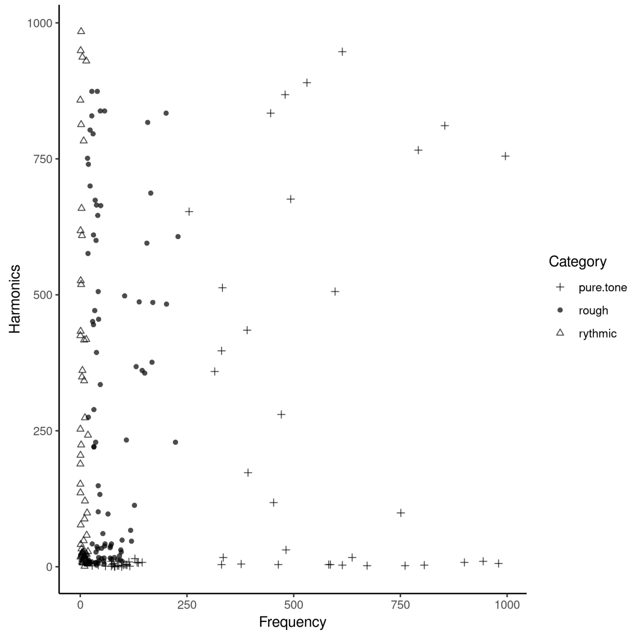

Fig. 5. Data captured during the user tests for the Blip. The x coordinate represent the frequency and the y the number of upper harmonic added.

A. Results

To evaluate the model the extracted rules were compared with the perception windows defined in [17] for the perceptual characteristics used in the experiments. Then, a fuzzy classifier using the data for each user was build, and a 10-fold cross-validation test performed. These results are shown in Sections III-A1 and III-A2, respectively.

1) Extracted Rules: Examples of the extracted rules, for the different categories, are shown in Table II. Looking at the rules in Table II, it can be seen that its parameter values match the expected perceptual windows defined in [17] for each category. The second rule, shows that some users still perceived the output as rough for higher frequencies. It can be interpreted as follows: as long as the number of upper harmonics remain low (3.7), the fundamental frequency can be either 30 or 54 for the combination to be perceibed as rough. This is still acceptable if we consider the effect of the upper harmonics.

If we look at the first rule considering the interval [15.8, 18.15], the possible combinations of the values within this

TABLE II
EXAMPLES OF THE EXTRACTED RULES. THE FIRST THREE ARE RULES EXTRACTED USING d = 1. IT CAN BE SEEN THAT THE PARAMETER RANGES MATCH THE EXPECTED PERCEPTUAL INTERVALS. THE LAST TWO RULES WERE EXTRACTED WITH d = 2. IT CAN BE SEEN THAT THE NEW COMBINATIONS ARE STILL WITHIN THE EXPECTED RANGES

Category	Frequency	Number of harmonics	Amplitude
Rhythmic d = 1	15.8 OR 18.15	3.7	0.25
Rough d = 1	30 OR 54	3.7	0.25
Pure-Tone d = 1	94 OR 154 OR 188	0.37	0.25
Rhythmic d = 2	15.8 OR 18.15	3.7 OR 59 OR 34 OR 28.4	0.25
Rough d = 2	30 OR 54	3.7 OR 20 OR 28.4 OR 6.36	0.25

interval and the rest of the parameters remain within the perceptual window for the category. In this way, the extracted rules are an interpretable model of the relationships between the parameter values and the categories.

2) Fuzzy Classifier: Table III contains the results of a 10-fold cross-validation for different classifiers using the data collected during the experiments. The configuration for each classifier is shown in the third column. The fuzzy classifier built on the top of the RuLer is located just below the k-Nearest Neighbor (configured with k = 1) and on the SVM with the Linear kernel (C = 10). These results are promising considering that for these tests the same gamma was used for all the trapezoidal functions. At this moment modules to calculate independent gammas as well as different forms of aggregation for the results of the triggered rules are being implemented. However, the result is good if we consider that the extracted rules can be easily read as shown above III-A1.

TABLE III
RESULTS OF A 10-FOLD CROSS-VALIDATION FOR DIFFERENT CLASSIFIERS USING THE DATA COLLECTED DURING THE EXPERIMENTS. FOR EACH CLASSIFIER ITS CONFIGURATION IS SHOWN

Accuracy	**Algorithm**	**Configuration**
93.714286	SVM_Gaussian	(g=0.00000100 C=1000.0)
92.714286	RandomForest	(NTrees=500)
89.785714	k-NearestNeighbour	(k=1)
89.738095	**RuLer fuzzy classifier**	d = 1, ratio = 1, gamma = 1
87.809524	SVM_Lineal	(C=10.0)
86.000000	SVM_Poly2	(g=0.001 C=1.0)
80.571429	NaiveBayes	(nb=50)

IV. CONCLUSIONS

In this contribution, a fuzzy rules extraction algorithm that works on top of the logic rules extraction algorithm "RuLer" (described in Section II-A) is presented. The algorithm allows to extend the validity of the if-then rules extracted by the RuLer. For this, it solves the contradictions that can emerge when the if-then rules are extended and pass from describing points in space to describing intervals. The result forms the cores of the trapezoidal functions to which the supports are added. The results inherit the interpretability of the original rules, and allow to build a chart that offers a quick inside of the distribution of the classes in the parameter space. The acuracy of the fuzzy classifier built for model validation, with user test data, gives promising results. Although the accuracy can be improved, possible places to adjust it are the aggregation of triggered rules and adding a variable gamma. However, the interpretability of the resulting rules offers an extra feature to consider the algorithm for computer music practice and research.

REFERENCES

[1] A. McLean, & R. T. Dean, (Eds.). "The Oxford Handbook of Algorithmic Music." Oxford University Press. 2018.
[2] J. Rohrhuber, & A. De Campo, "Improvising Formalisation: Conversational Programming and Live Coding." New Computational Paradigms for Computer Music. Delatour France/Ircam-Centre Pompidou. 2009.
[3] P. Dahlstedt, "Thoughts on creative evolution: A meta-generative approach to composition." Contemporary Music Review, 28(1), 43-55. 2009.
[4] N. Collins, A. McLean, J. Rohrhuber, & A. Ward "Live coding in laptop performance." Organised sound, 8(3), 321-330. 2003.
[5] F. Klawonn, A. Keller "Fuzzy clustering and fuzzy rules", in: Proceedings of the 7th International Fuzzy Systems Association World Congress (IFSA'97), vol. 1, Academia, Prague, 1997, pp. 193–198.
[6] B. Geva, "Hierarchical unsupervised fuzzy clustering", IEEE Transactions on Fuzzy Systems 7 (6) (1999) 723–733
[7] A. Shigeo, T. Ruck, "A fuzzy classifier with ellipsoidal regions", IEEE Transactions on Fuzzy Systems 5 (3) (1997) 358–368.
[8] F. Castro, A. Nebot, & F. Mugica "On the extraction of decision support rules from fuzzy predictive models." Applied Soft Computing, 11(4), 3463-3475. 2011.
[9] R. Berthold, "Mixed fuzzy rule formation." International journal of approximate reasoning, 32, 67-84. 2003.
[10] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, & M. Norouzi, "Neural audio synthesis of musical notes with wavenet autoencoders." arXiv preprint arXiv:1704.01279. 2017.
[11] I. Paz, À. Nebot, F. Mugica & E. Romero. "Generalizing successful parameter combinations: directed sound design for algorithmic music" Unpublished.
[12] G. T. Toussaint, G. T. "A Comparison of Rhythmic Similarity Measures." In ISMIR." 2004.
[13] L. Kuncheva, "Fuzzy classifier design" (Vol. 49). Springer Science & Business Media. 2000.
[14] I. Paz, À. Nebot, F. Mugica, & E. Romero."Modeling perceptual categories of parametric musical systems." Pattern Recognition Letters, 105, 217-225. 2018
[15] Git Repository https://github.com/ivan-paz/in-construction accessed January 2019.
[16] Blip http://danielnouri.org/docs/SuperColliderHelp/UGens/Oscillators/Blip.html accessed January 2019.
[17] C. Roads, "Sound composition with pulsars." Journal of the Audio Engineering Society, 49(3), 134-147. 2001.