Fuzzy ARTMAP Rule Extraction in Computational Chemistry

Răzvan Andonie, Levente Fabry-Asztalos, Bogdan Crivăț, Sarah Abdul-Wahid, and Badi‘ Abdul-Wahid

Abstract—We focus on extracting rules from a trained FAMR model. The FAMR is a Fuzzy ARTMAP (FAM) incremental learning system used for classification, probability estimation, and function approximation. The set of rules generated is post-processed in order to improve its generalization capability. Our method is suitable for small training sets. We compare our method with another neuro-fuzzy algorithm, and two standard decision tree algorithms: CART trees and Microsoft Decision Trees. Our goal is to improve efficiency of drug discovery, by providing medicinal chemists with a predictive tool for bioactivity of HIV-1 protease inhibitors.

I. INTRODUCTION

Several neural architectures have been successful for QSAR (Quantitative Structure-Property Relationship) and QSAR (Quantitative Structure-Activity Relationship) tasks. Among these are Fuzzy ARTMAP (FAM) architectures [1]–[3]. The FAM [4], which was used for estimating boiling points of alphatic hydrocarbons and aqueous solubility of organics, has been shown to be superior to the back-propagation neural network approach as well as other regression-based statistical correlations reported in the literature [1].

In previous work [5]–[7] we investigated the use of a Fuzzy Neural Network (FNN) architecture for biological activity (IC$_{50}$) prediction. The fuzzy IF/THEN rules were extracted from the trained network. These rules map chemical structure descriptors to predicted inhibitory values. The GA-FNN is a genetic algorithm optimized FNN, introduced in [6].

In [8], we also focused on the IC$_{50}$ prediction task, using a FAM-type prediction technique, the Fuzzy ARTMAP with Relevances (FAMR). The FAMR is an incremental learning system used for classification, probability estimation, and function approximation, introduced in [9]. The FAMR architecture is able to sequentially accommodate input-output sample pairs. Each training pair has a relevance factor assigned to it, proportional to the importance of that pair during the learning phase. The relevance factors are user-defined, or computed, and are proportional to the importance of the respective pair in the learning process. Our IC$_{50}$ prediction method has two stages. First, we use genetic algorithm (GA) optimization to modify the training dataset. This modification consists of finding the best relevances for the data, according to some fitness criterion. The fitness criterion assesses the FAMR IC$_{50}$ prediction accuracy for a given training/validation dataset with given relevances. In stage two, the final FAMR is trained using the dataset with optimized relevances.

FAM networks have the capability to easily expose the learned knowledge in the form of fuzzy IF/THEN rules; several authors have addressed this issue for classification tasks [10]–[13]. The final goal in generating such rules would be to explain, in human-comprehensible form, how the network arrives at a particular decision, and to provide insight into the influence of the input features on the target. To the best of our knowledge, no author has discussed FAM rule extraction for function approximation tasks, such as IC$_{50}$ prediction.

We contribute the following:

i. We discuss rule extraction from FAMR models, both in the context of function prediction and classification (Section III-B).

ii. We introduce a rule generalization algorithm (Section V-B) and a rule inference procedure (Section V-C), able to improve the rules extracted from a neural network.

iii. We analyze the generated rules, when predicting biological activity, and relate them to Lipinski’s Rule of Five which predicts bioavailability (Section VI).

iv. We compare our results to standard decision tree algorithms and to a different neuro-fuzzy model, the GA-FNN (Section V-C).

Section II introduces some FAM basic notation and reviews the FAMR approach for function approximation. In Section III, we discuss FAM and FAMR rule extraction. The computational chemistry problem is presented in Section IV. Our new approach for extracting and post-processing rules is applied to an experimental case in Section V. We discuss the chemical significance of our approach in Section VI and make the concluding remarks in Section VII.

II. THE FAMR MODEL FOR FUNCTION APPROXIMATION

We assume throughout this paper that the reader is familiar with the FAM neural network architecture, its learning phase, and its network parameters, as introduced in [4]. We only review the basic FAMR notation and present its function approximation capabilities. Details can be found in [9].
FAM networks map subsets of $\mathbb{R}^n$ to $\mathbb{R}^m$. Potentially, all FAM variations can be used for function approximation, since the FAM has been proven to be a universal function approximator [14].

A FAM consists (Figure 1) of a pair of fuzzy ART modules, $ART_a$ and $ART_b$, connected by an inter-ART module called Mapfield. The fuzzy $ART_a$ module contains the input layer, $F^a_1$, and the competitive layer, $F^a_2$. A preprocessing layer, $F^a_0$, is also added before $F^a_1$. The following notations apply: $M_a$ is the number of nodes in $F^a_1$, $N_a$ is the number of nodes in $F^a_2$, and $w^a$ is the $ART_a$ weight vector between $F^a_1$ and $F^a_2$. Analogous layers and notations appear in $ART_b$.

The ART modules create stable recognition categories in response to arbitrary sequences of input patterns. The $ART_a$ and $ART_b$ vigilance parameters, $\rho_a$ and $\rho_b$, control the matching mechanism inside the modules.

All input vectors are complement-coded by the $F^a_0$ layer. Each input vector $a = (a_1, \ldots, a_n)$ ($a_i \in [0,1]$, $1 \leq i \leq n$) produces the normalized vector

$$A = (a, 1 - a) = (a_1, \ldots, a_n, 1 - a_1, \ldots, 1 - a_n)$$

During learning, the Mapfield weights are updated: the strength of the weight projecting from the selected $ART_a$ category to the correct $ART_b$ category is increased, while the strengths of the weights to other $ART_b$ categories are decreased. A Mapfield vigilance parameter $\rho_{ab}$ calibrates the degree of predictive mismatch necessary to trigger the search for a different $ART_a$ category. If the weight projecting from the active $ART_a$ category through the Mapfield to the active $ART_b$ category is smaller than $\rho_{ab}$ (vigilance test), then the system responds to the unexpected outcome through the so-called match tracking. This triggers an $ART_a$ search for a new input category. After choosing an $ART_a$ category whose prediction of the correct $ART_b$ category is strong enough, match tracking is disengaged, and the network is said to be in a resonance state. In this case, Mapfield learns by updating the weights $w_{jk}^{ab}$ of associations between each $j$-th $ART_a$ category and each $k$-th $ART_b$ category.

As the $ART_a$ learning algorithm uses the fuzzy MIN operator, the $ART_a$ weight vector $a$ is a monotonically decreasing quantity and settles at the lower bound of the input vector or the weight vector $(A \wedge w^a_j)$, where $w^a_j$ is the winning $F^a_2$ weight vector. With complement coding, the category boundaries have a hyper-rectangle shape where the lower vertices are represented by the $ART_a$ weight vectors while the upper vertices are represented by the complement-coded part. In the case of “fast learning” (i.e., when the $ART_a$ learning rate $\beta_a = 1$ at all times), the hyper-rectangles grow monotonically in all dimensions, but their size is bounded above [15].

The main difference between the FAMR and the original FAM is the updating scheme of the $w_{jk}^{ab}$ weights. The FAMR uses the following iterative updating [9]:

$$w_{jk}^{ab(new)} = \begin{cases} w_{jk}^{ab(old)} & \text{if } j \neq J \\ w_{jk}^{ab(old)} + \frac{q_t}{Q_t} \left( 1 - w_{jk}^{ab(old)} \right) & \text{if } j = K \\ \frac{w_{jk}^{ab(old)}}{1 - \frac{q_t}{Q_t}} & \text{if } j \neq K \\ \end{cases}$$

where $q_t$ is the relevance assigned to the $t$th input pattern ($t = 1, 2, \ldots$) and $Q_t^{new} = Q_t^{old} + q_t$. The relevance $q_t$ is a real positive finite number directly proportional to the importance of the experiment considered at step $t$. According to [9], this $w_{jk}^{ab}$ approximation is a correct biased estimator of the posterior probability $P(k|j)$, the probability of selecting the $k$-th $ART_b$ category after having selected the $j$-th $ART_a$.

For functions that are known only at a certain number of points, when these functions map from the vector-valued real domain to the vector-valued real range, the $l$-th component of the predicted output vector, corresponding to an input pattern activating the $J$-th $ART_a$ category can be computed according to eq. (2).

$$\mu_{lJ} = \sum_{k=1}^{N_a} \epsilon_{kl} w_{jk}^{ab}$$

The $\epsilon_{kl}$ is the $l$-th component of the $k$-th $ART_b$ category prototype (the vector representing the category). The typical prototype for the $k$-th $ART_b$ category is its centroid. During the FAMR training process, the $l$-th component of the centroid can be approximated on-line by Kohonen’s learning rule. The approximation formula (3) incorporates an idea from [16]. The $size_b^j$ is the number of output vectors of the $k$-th $ART_b$ category and $b$ is the $l$-th component of $b$, the output vector of the current training pair $(a, b)$. At the end of the FAMR learning phase, we substitute $\epsilon_{kl}$ in eq. (2) with the resulting $c_{kl}^j$, before using the FAMR as a function approximator.

$$(\epsilon_{kl}^b)^{new} = (\epsilon_{kl}^b)^{old} + (b_l - (\epsilon_{kl}^b)^{old})/size_b^j$$

Observation: During on-line training, it may happen that categories “lose” vectors they have previously learned to represent. This is due to the fact that, a learned vector, when processed a second time, may be absorbed by a different category (created later) than it was the first time. Therefore, when computed on-line, $size_b^j$ only approximates the real number of output vectors of the $k$-th $ART_b$ category.

### III. Rule extraction

#### A. Rule extraction from FAM

In a FAM, each $ART_a$ category in the $F^a_2$ field (Figure 1) roughly corresponds to a rule, and the total number of rules is $N_a$. Each rule links antecedents to consequents.

Carpenter and Tan [10], [15] were the first who introduced a FAM rule extraction procedure. For the $j$-th $ART_a$ category, they considered the vector of weights $w_{ij}^a$ ($j = 1, \ldots, M_a = 2n$) to be the numerical representative of the category. They used these weights to define the antecedent of the rule:
the real feature values represented by weights \( w_{ij} \) were quantized into fuzzy categories. The quantization level \( L \) was defined as the number of feature values used in the extracted fuzzy rules. Two quantization schemes were considered: truncation and round-off, each dividing the range \([0, 1]\) into \( L \) equal intervals.

When FAM is used as a classifier, the \( ART_a \) module is replaced by a layer of nodes, corresponding to the output classes. In the classical FAM model [4], \( ART_a \) categories are mapped "one-to-one" to \( ART_b \) categories. Therefore, in a FAM classification process, each input vector \( a \) is mapped to a single output class and the extracted rules have only one consequent, which is the output class label. The FAM rule extraction method proposed in [10], [15] is applied only to classification tasks.

To reduce complexity of the fuzzy ARTMAP, a pruning procedure was introduced by Carpenter and Tan [10], [15]. This pruning selects a small set of good rules from the network. The algorithm evaluates each \( ART_a \) category and its predictive performance on the test set. This results in a confidence factor \( CF_j = \gamma U_j + (1 - \gamma) A_j \), where \( U_j \) is the usage of category \( j \), \( A_j \) is its accuracy, and \( \gamma \in [0, 1] \) is a weighting factor. The usage \( U_j \) is the fraction of training vectors coded by \( ART_a \) category \( j \) divided by a normalizing factor. Accuracy \( A_j \) equals the percent of test vectors predicted correctly by category \( j \) divided by a normalizing factor. \( ART_a \) categories with confidence factors below some threshold were removed from the network [10], [15].

In addition to rule pruning, Carpenter and Tan [10] used antecedent pruning, to eliminate redundant antecedents (by setting the corresponding \( ART_a \) weight value to zero). Their procedure was used in several classification applications [10], [11].

If rule pruning is executed frequently, this method has several computational limitations [12]. All training and validation samples must be stored in order to recompute the usage and accuracy. If any rule is modified, or a new one created, the FAM must be retrained, because samples that previously selected another rule may select the one recently modified or created, and vice versa. This also applies after pruning a rule. As the computational cost of evaluating usage and accuracy increases with the number of samples, rule pruning eventually becomes unfeasible. Addressing this issue, Andrés-Andrés et al. [12] proposed a new rule pruning method for FAM, with advantages for incremental learning: It does not store all previous patterns, and it does not use obsolete information to determine which rules to prune.

The rectangular basis function network (RecBFN) [17] uses rectangular basis functions for activation of the hidden neurons and constructs hyper-rectangles from data, where each hyper-rectangle covers a region that belongs to one class. Since the \( ART_a \) categories of FAM also have a geometric interpretation as hyper-rectangles in the input space, Tan et al. [13] combined the FAM with RecBFN. They obtained a hybrid neural model, capable of learning and revealing fuzzy rules.

B. Rule extraction from FAMR

Up to this point, FAM rule extraction referred only to classification [10]–[13] and not to function approximation. We now adapt Carpenter and Tan’s rule extraction method for function approximation tasks with the FAMR.

Carpenter and Tan [10], [15] chose the weight vector as the prototype of each \( ART_a \) category. This vector does not reflect the centers, or the centroid, of the category [16]. In our approach, the prototypes are the centroids, approximated by the \( ART_a \) equivalents of eqs. (2) and (3). In their FAM-RecBFN model [13], Tan et al. used a similar center estimation procedure.
In contrast to the FAM, the FAMR Mapfield module performs a “one-to-many” mapping, i.e., an $AR T_a$ category is simultaneously mapped to all $AR T_b$ categories. Therefore, in a classification task, the consequent term of a FAMR rule has $N_b$ variables, each obtained by quantization into fuzzy categories.

For function approximation, the output values are obtained by eqs. (2) and (3), and the consequent is an output vector with quantized components. In $IC_{50}$ prediction, the dimension of the output vectors is one (i.e., $l = 1$) in eqs. (2) and (3), and the consequent is the quantized fuzzy category of a scalar.

IV. THE COMPUTATIONAL CHEMISTRY PROBLEM

HIV-1 protease is an enzyme involved in the maturation of new viral particles. The strength of a potential inhibitor to this enzyme is measured by the $IC_{50}$ value. Current treatments for HIV/AIDS consist of co-administering a protease inhibitor with two reverse transcriptase inhibitors (usually referred to as combination therapy). This therapy is effective in reducing viremia to very low levels, but in 30-50% of patients it is ineffective due to resistance development. Extensive research has resulted in many HIV-1 protease inhibitors [18]–[21]. Due to resistance, poor bioavailability ² profiles, and toxicity associated with these therapies, there is an urgent need for a more efficient development of new drugs.

Our goal is to discover rules that will provide medicinal chemists with new tools to design novel molecules that have good biological affinity and bioavailability. Use of these rules should decrease the number of failures in the late stages of drug development, and thus decrease discovery time and cost.

We train, validate and test a neural network, and then extract the rules. The dataset used, obtained from [22]–[28], consists of 196 compounds with experimentally determined $IC_{50}$ values. The $IC_{50}$ target values for these molecules range from 0.28 to 11,800 nM. Twenty of these molecules are from [25], with $IC_{50}$ values ranging from 1.4 to 120 nM; these molecules are used as an external test set after the training is completed. The remaining 176 molecules are used for training and cross-validation. SYBYL ³ software was used to create the data files that provide molecular descriptors. As a result of normalization, descriptors for all the molecules in the data set fell in the range [0, 1].

Although biological activity data has been obtained for many more chemical structures at various pharmaceutical companies and academic laboratories, they are not available in the public domain. Actually, most classical QSAR studies for a specific enzyme system have been performed on small datasets, due to limited experimentally determined biological activity values in the public domain. The dimensionality (the number of physico-chemical features) characterizing these molecules is relatively high. Our dataset shares these undesired characteristics: it is small, with relatively many features (35), and highly overlapping.

One way to cope with small training datasets is to reduce the number of input features. In [6] we applied feature selection to define a subset of the 35 descriptors. This improved performance. In addition, the generated rules were simpler and easier to analyze.

In the present study, we focus on a model with four descriptors: molecular weight, number of H-bond donors and acceptors, and ClogP. The descriptors were not determined by feature selection, but based on domain knowledge: This is the set defined by Lipinski’s Rule of Five [29], which is accepted by the medicinal chemistry community. This is a rule of thumb for qualitatively predicting bioavailability of a chemical compound. Thus, our method has the potential to link bioavailability with biological affinity.

V. EXPERIMENTAL RESULTS: RULE EXTRACTION AND ANALYSIS

FAMR $IC_{50}$ prediction results, using the train/test datasets and novel compounds, can be found in [8]. In the following, we focus on rule extraction. In order to simplify the presentation, rather than optimizing the FAMR relevances, as we did in [8], we assign the same constant value to all relevances. In all experiments, we use on-line (incremental) learning: the training set is processed only once.

A. Rule extraction

The quantization level influences the quality of the rules. A good compromise is $L = 5$, in accordance with [10] and also our own experiments. In our application, we use round-off quantization for antecedents. The consequent levels are determined by our domain expert, an organic chemist. Domain experts are free to set the range for each quantization level which matches their experience and knowledge [13]. Following are the quantized ranges used in our experiments.

**The descriptor range:** Low: [0, 0.125), Low-Medium: [0.125, 0.375), Medium: [0.375, 0.625), Medium-High: [0.625, 0.875), High: [0.875, 1.0].

**The $IC_{50}$ value range:** Excellent: [0, 20), Good: [20, 50), OK: [50, 100), Mediocre: [100, 500), Terrible: [500, MaxValue]. It should be noted that low $IC_{50}$ is optimal.

Small differences of low $IC_{50}$ values are chemically more significant than the same amount of difference of high $IC_{50}$ values. Therefore, we use the Symmetric Mean Absolute Percentage Error (sMAPE): $200/K \sum |d - y|/(d + y)$ where $K$ is the number of training samples. We also use $r$, the Pearson product-moment coefficient of correlation [30]; for an accurate prediction, one would expect $r$ as close to 1 as possible.

A critical parameter of this system is the value of the vigilance parameter, $\rho_a$. If this value is too low, it results in a single $AR T_a$ category and thus in a single rule. If it is set too high, an excessive number of rules are generated. Not only does this result in overfitting, but it is difficult to gain useful information from numerous rules, and thus the primary purpose of rule extraction is not achieved. We base

²Bioavailability is the rate at which the drug reaches the systemic circulation.
³http://www.tripos.com/
\(\rho_a\) selection on the following requirements: \(a)\) low value of the sMAPE of the test set results, \(b)\) the number of rules in the range \([10, 20]\), \(c)\) high Pearson’s \(r\) value of the training set. A optimal value is \(\rho_a = 0.84\).

The remaining parameters of the FAMR are: \(\rho_b = 0.85, \rho_{ab} = 0\) (no match tracking). The learning parameters in the two ART modules are: \(\beta_a = \beta_b = 0.45\). These parameters were obtained from our previous published results [8].

All rules are described in the form: (Molecular Weight, hDonors, hAcceptors, ClogP) \(\Rightarrow\) IC\(_{50}\). From the trained FAMR we obtain the following set of rules:

- \(O_1\) : (Low-Medium, Low, Low, Low-Medium) \(\Rightarrow\) Terrible
- \(O_2\) : (Low-Medium, Low, Low, Medium) \(\Rightarrow\) Mediocre
- \(O_3\) : (Low-Medium, Low-Medium, Low-Medium) \(\Rightarrow\) Excellent
- \(O_4\) : (Low-Medium, Low-Medium, Low-Medium, Low-Medium) \(\Rightarrow\) OK
- \(O_5\) : (Low-Medium, Low-Medium, Medium, Medium) \(\Rightarrow\) Excellent
- \(O_6\) : (Low-Medium, Low-Medium, Medium, Medium) \(\Rightarrow\) OK
- \(O_7\) : (Medium, Low-Medium, Medium, Medium) \(\Rightarrow\) Excellent
- \(O_8\) : (Medium, Low-Medium, Medium, High) \(\Rightarrow\) Excellent
- \(O_9\) : (Medium, Low-Medium, Medium, High) \(\Rightarrow\) Excellent
- \(O_{10}\) : (Medium-High, Low-Medium, Medium, High) \(\Rightarrow\) Excellent
- \(O_{11}\) : (Medium-High, Medium-High, Medium-High, Medium-High) \(\Rightarrow\) Mediocre
- \(O_{12}\) : (Medium-High, Medium-High, Medium-High, Medium) \(\Rightarrow\) Terrible
- \(O_{13}\) : (Medium-High, High, Medium-High, Medium) \(\Rightarrow\) Mediocre

B. Rule generalization

In the following, we analyze the quality of the set of rules \(\{O_1, \ldots, O_{13}\}\), produced by the FAMR model. First, we define the following rule metrics (introduced in [31]):

- **Confidence**: The rule \(X \Rightarrow Y\) has confidence \(c\) if \(c\%\) of the molecules that contain \(X\) also contain \(Y\).
- **Support**: The rule \(X \Rightarrow Y\) has support \(s\) if \(s\%\) of the molecules contain \(X \cup Y\).
- **Minimum support**: The minimum acceptable confidence for a rule.
- **Minimum support**: The minimum acceptable support for a rule.

Rules \(\{O_1, \ldots, O_{13}\}\) have support between 0.0% and 16.47%, and confidence between 0.00% and 100.00%. In order to remove irrelevant rules (pruning), we introduce a minimum confidence criterion of 25% and a minimum support criterion of 2.5%. Rule \(O_3\) does not meet these criteria and was removed from the set.

The resulting set of rules shows the following characteristics: \(i)\) all rules are complete (i.e., contain one predicate for each descriptor), a consequence of the rule extraction algorithm, and \(ii)\) certain descriptor fuzzy categories do not appear in any rule.

To further analyze this rule set, we introduce two new measures:

- **Coverage**: The percentage of molecules which have the following property: There exists at least one rule for which the molecule’s descriptors fall within the range of all antecedents.
- **Accuracy**: The percentage of molecules which have the following property: There exists at least one rule for which the molecule’s descriptors fall within the range of all antecedents and, in addition, the output falls within the range of the consequent.

Assuming that some rules are too specific to the training set (overfitting), we attempt to generalize them, by applying a greedy Rule Generalization Algorithm (RGA). The RGA is applied to each rule \((X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) \Rightarrow (Y = y)\) in the set:

**Rule Generalization Algorithm (RGA).** Relax the rule by replacing one predicate \(X_i = x_i\) with a wild card value \((X_i = *)\). If the newly formed rule improves the confidence of the original rule, keep it in a pool of candidates. This procedure is applied for all the predicates in the rule, resulting in at most \(n\) generalized rules which have better confidence than the original rule, where \(n\) is the number of predicates in the original rule. If the candidate pool is not empty, replace the original with the pool candidate which maximizes the confidence. The algorithm is applied one more time to the first generalization and it stops when the candidate pool is empty (no better generalization can be found).

In the worst case, the number of predicate replacements for each rule is in \(O(n^2)\). Any relaxation of a rule increases (or does not change) the support of that rule. Therefore, relaxing a rule improves both its confidence and support.

Here is an example of applying this algorithm to rule (Low-Medium, Low-Medium, Low-Medium, Medium) \(\Rightarrow\) Excellent. For this rule, we have: support=6.25% and confidence=90.9%. The first derived generalization is (*, Low-Medium, Low-Medium, Medium) \(\Rightarrow\) Excellent. This generalization has a support of 8.52% and a confidence of 93.3%. After applying the algorithm to the generalized rule, we obtain a larger generalization: (*, Low-Medium, *, Medium) \(\Rightarrow\) Excellent, with support = 13.06% and confidence 95.65%.

After applying the algorithm to the rule set \(\{O_1, \ldots, O_{10}\} - \{O_3\}\), the following generalized rules are obtained:

- \(G_1\) : (*, Low-Medium, *, *) \(\Rightarrow\) Excellent
- \(G_2\) : (*, Medium, Low-Medium, *) \(\Rightarrow\) Excellent
- \(G_3\) : (Medium, *, Medium, *) \(\Rightarrow\) Excellent
- \(G_4\) : (*, Low, Low, Medium) \(\Rightarrow\) Mediocre
- \(G_5\) : (*, Medium-High, Medium-High, *) \(\Rightarrow\) Terrible
Certain rule generalizations overlap, so the resulting set has fewer rules (duplicate generalizations have been removed). This generalization heuristic works only if there is at least one predicate in the rule which, when removed, can lead to strictly better confidence.

C. Inference of new rules

As certain descriptor values do not appear in any rule, simple one-predicate rules are produced to cover those slices of the descriptor space. For each such \( x_i \) value of the \( X_i \) descriptor, we compute the conditional probabilities for each consequent \( IC_{50} \) category, \( P(Y = y_i | X_i = x_i) \), and then select the \( IC_{50} \) label which yields the maximum probability conditioned by \( X_i = x_i \).

The new rules have the form \( (X_i = x_i) \Rightarrow y_i \) and are filtered according to the minimum confidence (25%) and minimum support (2.5%) criteria described above. The following new rule passes the criteria:

\[ I_1 : (\text{Low}^*, *, *) \Rightarrow \text{Terrible} \]

The combined rule set \( \{G_1, \ldots, G_5\} \cup \{I_1\} \) is our end result.

Finally, we compare our FAMR rule extractor to the GA-FNN [6] and to the following standard decision trees implementations: CART (WEKA implementation) trees [32] and Microsoft SQL Server 2008 Decision Trees [33]. For the decision trees, rules are extracted from each non-root node. Naturally, the decision-tree derived rules have 100% coverage. The complete results are presented in Table I.

The FAMR \( \{G_1, \ldots, G_5\} \cup \{I_1\} \) rule set has very good coverage and accuracy. For the test set, the \( \{G_1, \ldots, G_5\} \cup \{I_1\} \) rules have the same accuracy as the rule set derived from classic decision trees system (the test set consists of 20 molecules, so a difference of 5% translates to one incorrect prediction). This is rather surprising, considering that decision trees are a dedicated tool for rule generation, whereas the FAMR was essentially designed as a primary prediction/classification model.

VI. CHEMICAL INTERPRETATION

Our rules address the problem of biological activity. Lipinski’s Rule of Five addresses the problem of bioavailability. Biological activity and bioavailability are different characteristics of molecules, but equally important in drug design. It is extremely difficult to design drugs that have both good biological activity and good bioavailability.

For medicinal chemists, our \( \{G_1, \ldots, G_5\} \) rule set provides the most insight into the descriptor ranges promoting good biological activity. The next step would be to connect this rule set to Lipinski’s Rule of Five [7]:

\( \text{molecular weight} < 500, \text{H bond donors} < 5, \text{H bond acceptors} < 10, \text{ClogP} < 5 \Rightarrow \text{Good Bioavailability} \)

This rule could be matched, for example to rule \( G_1 \).

However, we also have incompatibilities: rule \( I_1 \) is not comparable with Lipinski’s rule. Rule \( I_1 \) connects low molecular weight to poor biological activity. On the other hand, Lipinski’s rule connects low molecular weight (less than 500) with good bioavailability. It appears that, in this particular case, \( I_1 \) is too general.

Clearly, more work is required to elucidate a connection between our generated rules and Lipinski’s rule. These preliminary findings suggest that our approach is promising.

VII. CONCLUSIONS

In Carpenter and Tan’s approach [10] and later, in [12], the pruning operations were performed on the FAM network, modifying its structure. We operate here directly on the inferred set of rules. This gives us more flexibility and, unlike in [10], we do not have to retrain the network after modifying the rule set. Computationally, our approach is attractive, because the number of rules is generally much smaller than the number of training vectors. The trade-off is that we do not generate the corresponding simplified FAM for improving the prediction phase. In our previous experiments [8], we have obtained very good prediction results using the GA-optimized FAMR. Therefore, in the present study, we did not focus on prediction accuracy, but on generating optimal rules.

The relatively small training set we used results in rule overfitting. This explains the poor GA-FNN rule extraction results, where no post-processing was applied. We could balance overfitting in the FAMR by heuristic generalization and adding one-predicate rules. The combined method FAMR+rule generalization+inference-of-new-rules proves to be efficient. In our future experiments, we aim to verify this method on different sets of chemical descriptors and on other datasets.

REFERENCES

<table>
<thead>
<tr>
<th>Method/rule set</th>
<th>Training set coverage</th>
<th>Training set accuracy</th>
<th>Test set coverage</th>
<th>Test set accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAMR: {O_1, \ldots, O_{13}}</td>
<td>57.39%</td>
<td>36.93%</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>FAMR: {G_1, \ldots, G_5}</td>
<td>86.36%</td>
<td>65.34%</td>
<td>90%</td>
<td>75%</td>
</tr>
<tr>
<td>FAMR: {G_1, \ldots, G_5} \cup {I_1}</td>
<td>88.64%</td>
<td>67.61%</td>
<td>90%</td>
<td>75%</td>
</tr>
<tr>
<td>GA-FNN</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>CART</td>
<td>100%</td>
<td>64.2%</td>
<td>100%</td>
<td>75%</td>
</tr>
<tr>
<td>Microsoft Decision Trees</td>
<td>100%</td>
<td>69.32%</td>
<td>100%</td>
<td>80%</td>
</tr>
</tbody>
</table>


