RIGA TECHNICAL UNIVERSITY

Information Technology and Management Science

ISSN 2255-9094 (online)
ISSN 2255-9086 (print)
December 2018, vol. 21, pp. 6–14
doi: 10.7250/itms-2018-0001
https://itms-journals.rtu.lv

Methodology for Knowledge Extraction from Trained Artificial Neural Networks

Andrey Bondarenko[1], Ludmila Aleksejeva[2]
[1, 2] *Riga Technical University, Riga, Latvia*

Abstract **– Artificial neural networks are widely spread models that outperform more basic, but explainable machine learning models like classification decision tree. However, their lack of explainability severely limits their area of application. All mission critical areas or law regulated areas (like European GDPR) require model to be explained. Explainability allows model validation for correctness and lack of bias. Thus, methods for knowledge extraction from artificial neural networks have gained attention and development efforts. The present paper addresses this problem and describes a knowledge extraction methodology which can be applied to classification problems. It is based on previous research and allows knowledge to be extracted from trained fully connected feed-forward artificial neural network, from radial basis function neural network and from hyper-polytope based classifier in the form of binary classification decision tree, elliptical rules and If−Then rules.**

Keywords **– Artificial neural network, feed-forward neural networks, knowledge acquisition, knowledge extraction, radial basis function neural networks.**

I. Introduction

Artificial neural networks are well known to any machine learning practitioner. ANNs are used (usually) as a non-linear classifier or a regression model. ANNs are widespread and in many cases outperform more classical explainable models like linear regression or C4.5 decision trees. Due to black-box nature of ANNs, their application is limited in mission-critical areas, such as healthcare, finance, law, atomic power and others. For example, thanks to GDPR law in the EU (effective as of 28 May 2018), all life changing decisions (like loan rejection) performed by algorithm should be explainable. Additionally, an ability to explain how a classification decision is made can lead to new insights and generation of deeper understanding of the problem under consideration. Figure 1 lists steps required to perform knowledge extraction.

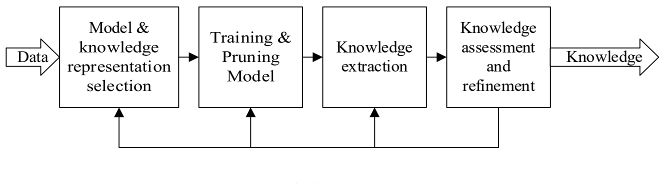
Data
Model & knowledge representation selection
Training & Pruning Model
Knowledge extraction
Knowledge assessment and refinement
Knowledge

Fig. 1. Knowledge extraction steps.

The first step is to understand the end goal – the way knowledge will be used; based on that and presence or absence of the trained model (classifier), the model can be selected or appropriate knowledge algorithm can be applied. Next step is model training (if needed) and trained ANN pruning (highly welcome step, although optional), afterwards knowledge extraction algorithms can be applied. The final step is knowledge assessment and refinement, where knowledge in the form of extracted rules can be assessed and refined. Apart from description of algorithms, the current paper presents guidelines for knowledge representation schema selection (which is correlated to available algorithms), training and extraction and, finally knowledge assessment and refinement. All these steps are covered in Section V.

The following paper is organized as follows: Section I contains introduction; Section II covers knowledge extraction overview and preliminaries, and Section III provides an overview of algorithms. Section IV holds methodology description for knowledge extraction, and Section V contains conclusions about the developed methodology.

II. Knowledge Extraction Overview

A. Types of Knowledge Extraction Algorithms

Let us make a review of the knowledge extraction algorithms. According to [1], there are three main types of algorithms:

- decompositional;
- pedagogical;
- eclectic.

Decompositional knowledge extraction algorithms are trying to extract knowledge using internal structure of the model – ANN, SVM or any other black-box model. It means that decompositional algorithms designed for knowledge extraction from a specific family of ANN are not applicable to other types of classifiers. On the other hand, such methods usually have higher performance (in terms of accuracy and precision) and generally should run faster.

Pedagogical knowledge extraction approaches do not make assumptions in regard to inner structure of the model being processed. Usually such algorithms use the trained classifier model as a black box to sample input space and get dense outputs. This allows the algorithm to build dense representation of decision boundary and capture it in the form of M of N rules or decision trees. It is easy to note that such algorithms are model agnostic, i.e., they can be applied not only to the trained ANNs, but to other types of classifiers as well.

Eclectic approaches are a combination of both decompositional and pedagogical approaches. More detailed description of different knowledge extraction algorithms is out of scope of the present paper. For examples of decompositional algorithms a reader can refer to [2]–[9]; algorithms using a pedagogical approach are described in [10]–[13], and eclectic knowledge extraction algorithms are presented in [14]–[18]. As a summary of our analysis, we present comparison of different rule types in each algorithm class. The estimation is made as a result of review of existing publications (see a list of references above). Based on the properties that are important for us (three first), decompositional algorithms have been selected as the most prospective ones. Later on, a pedagogical algorithm for elliptic rule extraction has been developed to extend knowledge extraction workflow applicability. Accuracy, portability and tunability have been selected as the most important properties, as a combination of high accuracy and high tunability contributes to the most powerful knowledge extraction approach giving a user flexibility to extract both precise and comprehensible rules. Table I (the higher value is the better) contains properties that can help other researchers make their choice based on their own prioritization of algorithm class properties.

TABLE I
COMPARISON OF KNOWLEDGE EXTRACTION ALGORITHM TYPES

Property	**Algorithm class**		
	Decompositional	***Eclectic***	***Pedagogical***
Classification accuracy	4	3	3
Portability (not specific to classifier)	1	2	3
Tunability	4	3	3
Algorithm consistency (several runs – the same result)	1	3	2
Speed	1	3	3
Knowledge representation variety	4	3	3
Scalability (Big data)	3	3	3
Algorithm complexity (computational)	1	3	3

It is believed that decompositional approaches are able to produce most precise knowledge. On the other hand, pedagogical methods are model-agnostic ones and can be applied to different types of classifiers (hence portable). As decompositional methods are accessing inner structure of ANN, they have much more power in how precise or comprehensive the extracted knowledge will be, and all in all such an algorithm can be fine-tuned at a more fine-grained level than a pedagogical one, which treats ANN as a black-box / oracle predicting class values for given inputs. Algorithm consistency – an ability to produce the same result over several runs – is a weak point for all algorithms, but currently decompositional ones are the weakest algorithms in this regard. Speed of execution is somewhat discussible because pedagogical algorithms need to generate additional inputs to precise location and form of classification hyperplane, but this can be done in an intelligent way, while decompositional algorithms will deal with full inner complexity (number of trainable parameters – weights or neurons) of ANN. Knowledge representation variety assessment again is somewhat discussible, but we believe that a decompositional approach gives an edge in terms of how easy it is to construct an algorithm that will be able to produce one or the other type of knowledge. In terms of scalability, all types of algorithms are equally parallelizable. As far as algorithm complexity is concerned, at present the count of neuron output values is what consumes most of the time within decompositional approaches. On the other hand, intelligent generation of additional input data can lower the amount of computations dramatically. Hence, an algorithm will only try to generate rules explaining outputs in terms of inputs. Therefore, decompositional algorithms are generally assumed to be more computationally complex.

B. Comprehensibility vs Performance Tradeoff

Another important decision to be made when considering knowledge extraction is comprehensibility vs. classification accuracy (or any other performance measure) tradeoff. There is a direct correlation between accuracy of knowledge and complexity. The more accurate knowledge is, the more complex it will be; thus, it will be less comprehensible. If knowledge to be extracted should be more comprehensible, its complexity should be low; hence, accuracy (performance) will be lower (in comparison with more complex knowledge). Thus, before knowledge extraction a user must prioritize understandability and performance. Complex knowledge (hence more accurate) can be used for model embedding (i.e., as If-Then rules) into some sort of system (embedded systems or, for example, DBMS).

C. Knowledge Representation

Knowledge representation is next important choice to be made. Some knowledge representation schemes are more comprehensible at the expense of expressive richness (count of rules needed).

Propositional If−Then / If−Then−Else rules. Rules of this type consist of If conditional clause and Then classification clause. Optionally, a rule can contain Else classification clause. The If condition part of a propositional rule is a boolean combination of logical conditions on the input variables. Condition part can contain conjunctions, disjunctions and negations. An example of such a rule is as follows:

$$If\ X = x\ and\ Y = y, Then\ Class = A,$$

with $\boldsymbol{X}, \boldsymbol{Y}$ being input variables and $\boldsymbol{x}, \boldsymbol{y}$ – possible values of these variables. For continuous input variables, the conditions are usually specified in the form of range restrictions on the values, e.g.:

$$X \in [c_1, c_2]\ and\ Y > c_3\ with\ c_1, c_2, c_3 \in R.$$

M-of-N rules. This type of rules is closely related to propositional rules. They are expressions of the form:

If (at least | exactly | at most) M of the N conditions C_1, C_2, ... C_N are satisfied then Class A_1.

Equation rules. This type of rules is very similar to oblique rules, but defines boundaries using equations of slightly more complex hyperplanes, like ellipsoids or parabolas. Example:

$$If\left(\frac{(x-h)^2}{r_x^2}+\frac{(y-h)^2}{r_y^2}\leq 1\right) Then\ Class = A$$
$$Else\ Class = B.$$

One major drawback of these rules is that they are more difficult to understand and comprehend, especially in the case of high dimensional input space.

Fuzzy rules. Rules of this type are very similar to propositional If–Then–Else rules, the only difference is that instead of Boolean logic, fuzzy rules are multi-valued. In fuzzy rules, the universe of discourse of the variables is a fuzzy set. Example of fuzzy classification rule is as follows:

$$If\ (X\ is\ Medium)\ and\ (Y\ is\ High)\ Then\ Class = A$$
$$Else\ Class = B.$$

Here high, medium and low are variables of a fuzzy set. In fuzzy sets, each element is associated with the corresponding grade of membership. Like propositional rules, fuzzy rules are generally easy understandable as they operate with linguistic concepts.

Strengths and weaknesses of knowledge representation schemas are summarised in Table II.

TABLE II
COMPARISON OF RULE TYPES

Criteria	If–Then / Decision Tree	M-of-N	Oblique / Equational	Fuzzy
Understandability	4	2	1	3
Compactness	2	1	4	3
Ease of use (embeddability)	4	3	2	1
Expressive power	3	1	4	2
Total	**13**	**7**	**11**	**9**

If–Then rules and decision trees are most welcome as they are easily embeddable, have good comprehensibility and acceptable expressiveness and compactness. Each decision tree can be easily represented as a set of If-Then rules. They both can be easily understood by domain experts and as they have "built in" inference engine – they are easily embeddable (for example, as a set of If-Then clauses inside regular SQL database management system).

Equational and oblique rules are most expressive while least interpretable by human experts. Expressiveness that means lower count of rules can be used to describe the same problem (in comparison with other rule types). Last two groups – fuzzy rules and M-of-N rules – are of less interest as they are bound to fuzzy neural networks, which are less common and are not easily embeddable (require fuzzy inference engine). M-of-N rule understandability is another discussible property. We have used a ranked approach making assessment in Table II – any single row has unique marks. The higher mark – the better.

Due to its simplicity, ease of use and embedability (embedded inference mechanism), If-Then rules (or classification decision tree) have been selected as the main knowledge schema representation type. To extend applicability of the proposed knowledge extraction workflow, elliptical rules have been tested to describe radial basis function neural networks (RBFNN). Although the algorithm itself is pedagogical, it can be applied to any kind of classifier to describe it with a set of elliptical rules.

III. ALGORITHMS

The following algorithms have been developed and used in the presented knowledge extraction methodology:

- As the first step – an algorithm for neuron pruning is presented [19], [20].
- Algorithm for binary classification decision tree extraction from the trained multilayered fully connected artificial neural network [21].
- Algorithms for extraction of elliptical rules from the trained radial basis function neural network [22], [23] and algorithm for If–Then rule extraction from piecewise linear classifier [24], [25] are covered.

Let us describe all three proposed algorithms. We will start with a decision tree extraction algorithm that occupies a central part (due to its general applicability) in the methodology.

A. Decision Tree Extraction

The developed algorithm allows extracting a classification tree from the trained ANN. This algorithm assumes a multilayered feed-forward fully connected neural network with sigmoidal activation functions (although we do not see any problems in applying this algorithm to RBF neurons).

Decision tree extraction highly welcomes ANN pruning, optional neuron output discretization and mandatory neuron output clusterization steps before the final decision tree extraction step. Overall binary decision tree extraction algorithm is depicted in Fig. 2.

As we can see, the first step (after training) is pruning. It is a highly desirable step applied before knowledge extraction. It not only raises generalization abilities of the trained ANN, but also reduces the number of neurons, hence, lowers computational requirements and complexity of knowledge, which will be extracted. Thus, by controlling the "aggressiveness level" of pruning procedure, one can control extracted rule complexity/comprehensibility.

Selection of the most appropriate pruning algorithm is a complex task. It highly depends on context. In Table III, a reader can find justification of choice of a sensitivity-based pruning approach.

According to the criteria listed, a sensitivity-based pruning approach has been selected as it represents compromise between factors specified as important. The aim is to select an efficient and easily implementable method, which could be used as the first step in knowledge extraction pipeline.

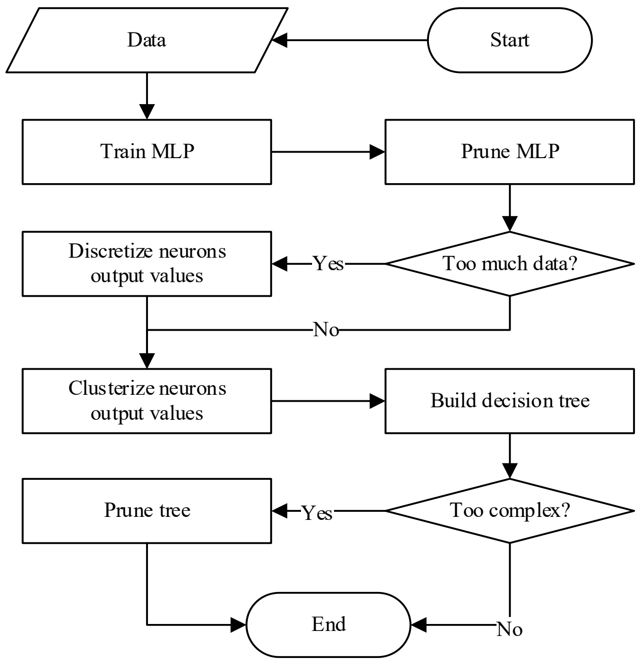

Fig. 2. Decision tree extraction algorithm.

TABLE III

COMPARISON OF PRUNING ALGORITHMS

Criteria	Sensitivity based	Sensitivity analysis II OBD/OBS	Magnitude based	Weight decay	Mutual information	Significance based	Interactive pruning
Simplicity	4	1	5	2	3	1	2
Execution time	0	1	3	2	3	1	1
Memory footprint	3	0	3	3	1	2	2
No special training procedure	2	1	2	0	2	2	2
Classification precision / generalization	3	2	0	n/a	n/a	3	n/a
Pruned neuron / weight count	3	3	0	n/a	n/a	4	n/a
Applicability to RBFNN	2	1	1	2	0	2	n/a
Total score	**17**	**9**	**14**	**9**	**9**	**15**	**7**

Sensitivity-based method is coupled with retraining, fail-retry mechanism, which will not stop an algorithm until specific count of subsequent failed pruning iterations is encountered (thus allowing it to escape local minimums), and memory pocket, which allows storing last best known neural network structure that showed best performance. Another feature is an error-rise tolerance parameter, which allows accepting a classification error decrease to a certain degree over the course of pruning. Table IV depicts conceptual parts of existing sensitivity-based pruning algorithms and highlights the proposed improvements.

TABLE IV

COMPARISON OF THE PROPOSED PRUNING ALGORITHM WITH THE CLOSEST RIVALS

Pruning algorithm specific	N2FPA	N2PS	Proposed
Neuron/weight pruning approach	Sensitivity	Magnitude	Sensitivity
Pruned network error-rise threshold usage	+	+	+
Saving pre-pruned network state	–	+	+
Pruning retry mechanism	–	–	+

The developed pruning approach allows both node and weight pruning. In sets of experiments it was shown [19] that weight pruning could produce ANN with higher accuracy, but at the cost of longer computation. But overall node pruning is a default option to be used as difference is rather small. For details of algorithm implementation and experimental results, as well as an extensive list of pruning algorithms please refer to [19], [20].

Table V describes the influence of the proposed pruning algorithm parameters on the end goal. In addition, it highlights differences between pruning of different entities – neurons and weights.

There are two critical parameters that provide a user with an ability to influence pruning aggressiveness. These parameters are as follows:

- error rise threshold (an error rise level after which neuron removal is rolled back) and
- pruning retry count (when neuron/weight removal is cancelled – we need to stop or will retry pruning).

Error rise threshold ideally should allow for a small growth in an error, if precision is less important than comprehensiveness, then it can be set even higher. In our experiments, 5 % threshold value has been used.

Last pruning parameter is retraining epoch count – the number of retrain epochs after neuron/weight removal. Choosing a retrain epoch count value to be too high will waste computational resources, while choosing it to be too small can trap algorithm in local minimum.

The proposed sensitivity-based neuron pruning [20] will result in a faster pruning procedure than weight pruning. There is a chance that weight pruning will produce ANN with higher accuracy and generalization abilities, but a pruning procedure will run longer (as there is a much higher count of weights than neurons). Large retrain epoch count will allow pruning to run longer. Large error rise tolerance threshold will allow pruning more neurons and hence will produce smaller ANN with less neurons, which will result in smaller decision tree and more understandable knowledge. If more precise knowledge is required, an error rise threshold should be as small as possible. Small retraining epoch count alone can speed up pruning, but can lead to situations when a network will not be able to fully restore its accuracy after pruning; hence, a classification error will rise too much and a pruning procedure will stop.

TABLE V
INFLUENCE OF PRUNING PARAMETERS ON PRUNING SPEED, EXTRACTED KNOWLEDGE ACCURACY

Requirement	Solution	Consequences
Faster pruning	Set a small retrain epoch count	Pruning can be trapped in local minimum
	Apply neuron pruning	Results should be as good as for weight pruning, but there is a change that weight pruning would be better
	Set a small error rise tolerance threshold (fast stopping)	Few (if any) pruned nodes – complex and precise decision tree
	Set a small failed pruning retry count	Faster pruning, but early pruning stopping can occur
Smaller decision tree (more pruned neurons)	Set a large retrain epoch count	Less chance for pruning to get into local minimum
	Set a larger error rise tolerance threshold	Will prune neurons/weights even if they are somewhat important
	Set a large failed pruning retry count	Longer pruning, but less chance of getting into local minimum
More precise knowledge (less pruned neurons)	Set a large retrain epoch count	Less chance for pruning to get into local minimum
	Set a small error rise tolerance threshold	Less pruned neurons or weights
Better classifier generalization	Set a large failed pruning retry count	Longer pruning, but less chance of getting into local minimum
	Apply weight pruning	Longer pruning, not always, but can get better results than neuron pruning

Fully connected neural network layers are widespread and used not only to build fully connected ANNs classifiers, but also as a part of many deep learning architectures, be it long-short term memory (LSTM) or convolutional NN.

In addition, a binary classification decision tree is easily understandable knowledge format. Finally, a decision tree can be easily translated into a set of If-Then rules. The first step for knowledge extraction (which is optional, but highly welcome) is neuron output value discretization and clusterization. Neuron output value discretization algorithm proposed by Rudy Setiono in [2] is shown in Fig. 3.

Neuron output value discretization is highly recommended in cases when one has a large dataset because in the next step (neuron output value clusterization) a high number of unique neuron output values within each separate neuron to be clustered will worsen computation time.

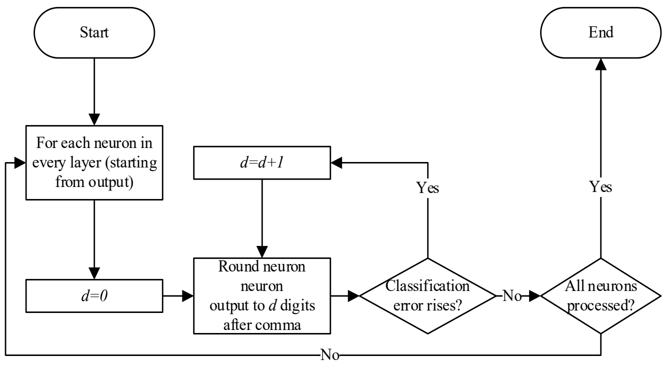
Start
End
For each neuron in every layer (starting from output)
d=d+1
Yes
Yes
d=0
Round neuron neuron output to d digits after comma
Classification error rises?
No
All neurons processed?
No

Fig. 3. Neuron output value discretization.

Neuron output value discretization is a simple approach to round up and lower amount of data points to be processed. Next step is neuron output value clusterization (see Fig. 4).

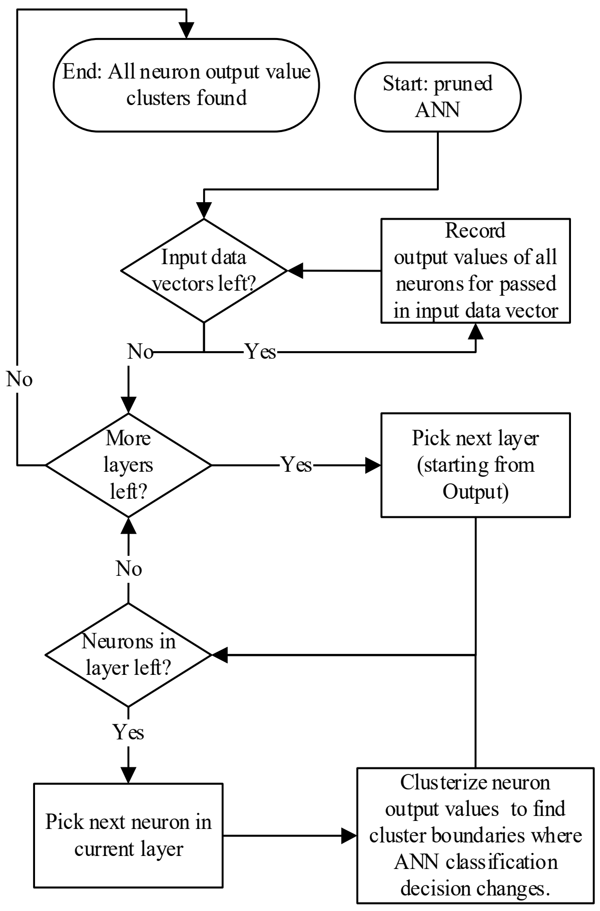
End: All neuron output value clusters found
Start: pruned ANN
Input data vectors left?
Record output values of all neurons for passed in input data vector
No
Yes
No
More layers left?
Yes
Pick next layer (starting from Output)
No
Neurons in layer left?
Yes
Pick next neuron in current layer
Clusterize neuron output values to find cluster boundaries where ANN classification decision changes.

Fig. 4. Neuron output value clusterization.

Neuron output value clusterization algorithm brakes non-linear nature of ANN classifier by substitution of all neuron output values belonging to a single cluster with cluster mid-point – mean output value of the cluster. As clusters are formed with constraints on classification accuracy worsening – such an operation should not lower ANN performance. Doing this operation sequentially neuron by neuron (in random order), layer by layer starting from output layers allows clusterizing all neurons down to an input layer.

Finally, when the number of neuron outputs is sufficiently small (due to output clusterization) it is possible to describe each layer outputs in terms of its inputs using If-Then rules. In his NeuroRule [2], Rudy Setiono proposes such an algorithm. The shortcoming of the method is a necessity to merge sets of rules describing different layers. Such sets of rules must be clustered, pruned and merged.

In contrast, Fig. 5 shows a modified decision tree extraction algorithm, which allows skipping rule generation for each layer and directly generating decision tree using cluster boundaries of input layer output values. A detailed algorithm description and experimental results can be found in [21]. This algorithm is better than NeuroRule algorithm [2], but uses a classical decision tree algorithm applied to inter-cluster boundary points of output values of input layer neurons to build a classification decision tree. Only these cluster boundaries of output values (or midpoints that can serve as a border between two nearby clusters), i.e., input neuron output values that influence a classification decision, i.e., if a neuron output value moves over such a point, the classification decision made by the whole network can change, are allowed to be used as splitting points of input space for decision tree construction.

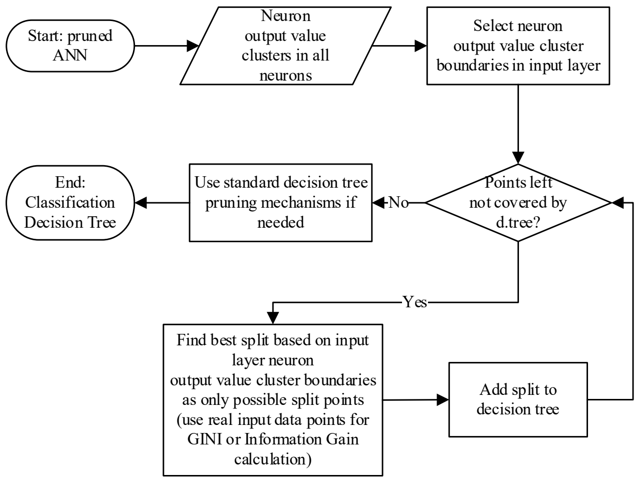
Start: pruned ANN
Neuron output value clusters in all neurons
Select neuron output value cluster boundaries in input layer
Points left not covered by d.tree?
No
Use standard decision tree pruning mechanisms if needed
End: Classification Decision Tree
Yes
Find best split based on input layer neuron output value cluster boundaries as only possible split points (use real input data points for GINI or Information Gain calculation)
Add split to decision tree

Fig. 5. The proposed decision tree extraction algorithm.

This algorithm is not prone to curse of dimensionality, and is able to produce a classification decision tree for multiple classes.

B. Elliptical Rule Extraction from RBFNN and If-Then Rule Extraction from Piecewise Linear Approximation

The idea behind both algorithms is quite simple. First of all, both of them are optimization-based approaches, which somehow limits them as makes them prone to curse of dimensionality, but in return for If-Then rules in case they are extracted from piecewise linear classifier because of convex optimization problem – global optimum will always be found. In regard to RBFNN, elliptical rules are much more expressive than If-Then rules and only few such rules can describe a complex classification decision boundary, which has a large number of If-Then rules. General idea is to recursively search for inscribed ellipsoid or hyper-polytope into the RBFNN classification boundary or piecewise linear classification. First iteration is straightforward, subsequent ellipsoids/hyper-polytopes should be searched with slight modifications in regard to constraints. Experimental evaluation and further algorithm details of elliptical rule extraction from RBFNN can be found in [22], [23]. Details of If-Then rule extraction from piecewise linear classifier can be found in [24], [25].

IV. Methodology of Knowledge Extraction

Based on the presented algorithms [19]–[25], the methodology of knowledge extraction from several classifier models has been developed. According to Fig. 1, the methodology consists of the four steps:

- model and knowledge representation schema selection;
- model training and pruning;
- knowledge extraction;
- knowledge assessment and refinement.

Let us take a closer look at each of the steps listed above.

A. Knowledge Representation Schema Selection

User can already have a pre-trained neural network, and in this case he will choose a knowledge representation schema supported by knowledge extraction algorithms that are capable of processing ANN he has. If this is not the case, then as it has been noted, the methodology incorporates three algorithms for knowledge extraction (If-Then rules, decision tree, and elliptical rules). In addition, a user is provided with pros and cons of other types of rules using Table II. Another important factor to bear in mind is limitations of the developed knowledge extraction algorithms summarized in Table VI. At this point, using pros and cons of the presented knowledge representation schemas (in regard to decision tree, elliptical rules, and If-Then rules) one can make a selection of classifier + knowledge extraction algorithm + knowledge representation triplet selection.

TABLE VI

Model and Knowledge Scheme Selection Based on Feature and Class Number

Given Attributes Number	Given Classes Number	Selected Model
Any	Any	MLP + Decision tree
Less than 4	Any	RBF + Elliptical rules
Less than 11	2	Polytopes + If–Then rules

During initial classifier model selection, dimensionality of dataset to be classified should be considered. Datasets of small dimensionality can be processed by RFBNN. For cases of ten or less input dimensions, it is feasible to build linear convex polytope classifier (or multi-surface-method tree based classifier) and apply the developed optimization-based If–Then rule extraction method. Finally, a binary classification decision tree can be extracted from feed-forward ANN regardless of input dimension count. In addition, there are no theoretical limitations for application of a decision tree extraction algorithm to RBFNN. But, in any case, according to “No Free Lunch” theorem, there could be no single classification model performing equally well on all kinds of datasets. Therefore, feed-forward ANN could not be the best classifier in all cases; hence, RBFNN and Polytope classifiers can be used. It is possible to combine several algorithms covering specific input space sub-space regions.

B. Model Training and Pruning

Model training is out of scope of the present paper, but model pruning has been developed in scope of decision tree extraction from multilayered fully connected ANN (see Fig. 6). Nevertheless, RBFNN networks (in elliptical rule extraction pathway) can be pruned using the developed algorithm. Model pruning is the first step, at which complexity of extracted knowledge can be controlled. The more aggressive pruning will be used, the smaller decision tree will be extracted. Less complex rules / knowledge mean more comprehensibility and less accuracy – depending on a knowledge usage scenario, aggressive pruning can be desirable or not.

C. Knowledge Extraction

When model training (and if required pruning) has taken place, knowledge extraction step can take place. Multilayered feed-forward ANN can be represented as a binary classification decision tree (which can be translated into If-Then rules), and RBFNN classifier can be translated into elliptical rules (in addition, there are no theoretical limitations to apply a decision tree extraction algorithm, but no experiments have been conducted yet). Finally, If-Then rules can be extracted from piecewise linear classifier.

D. Knowledge Assessment and Refinement

Final step, which should be performed, is the assessment of extracted knowledge and mitigation of potential problems. This is the rightmost largest block of Fig. 6.

Figure 6 uses wording like “accuracy too low”, “knowledge too complex” – we must stop here and describe roots of such informal description. As a usual end goal drives process, in knowledge extraction if model understanding is required (for example, how exactly a specific classifier makes a decision about loan rejection) one can assume that getting smaller number of rules is of higher importance; thus, model accuracy drop is acceptable. The extent of accuracy drop that is acceptable is again a case specific value. In one scenario, drop of classification accuracy of 10 % is acceptable, in others a 2 % drop is not an option. The same applies to knowledge complexity – is it complex or does not depend on domain experts’ subjective judgment.

The first step of assessment and refinement is classification performance check. If classification performance is too low, several options exist. First, (Fig. 6 – arrow 1), knowledge extraction parameters can be adjusted; this is applicable to If–Then rule extraction routine as it has a recursion depth threshold and applied to elliptical rule extraction routine as it can have a threshold for count of extracted ellipsoids.

Both thresholds can be increased to get more coarse-grained rules. If this is not the case, then (Fig. 6 – arrow 2) pruning parameters can be relaxed and pruning repeated (with subsequent knowledge extraction).

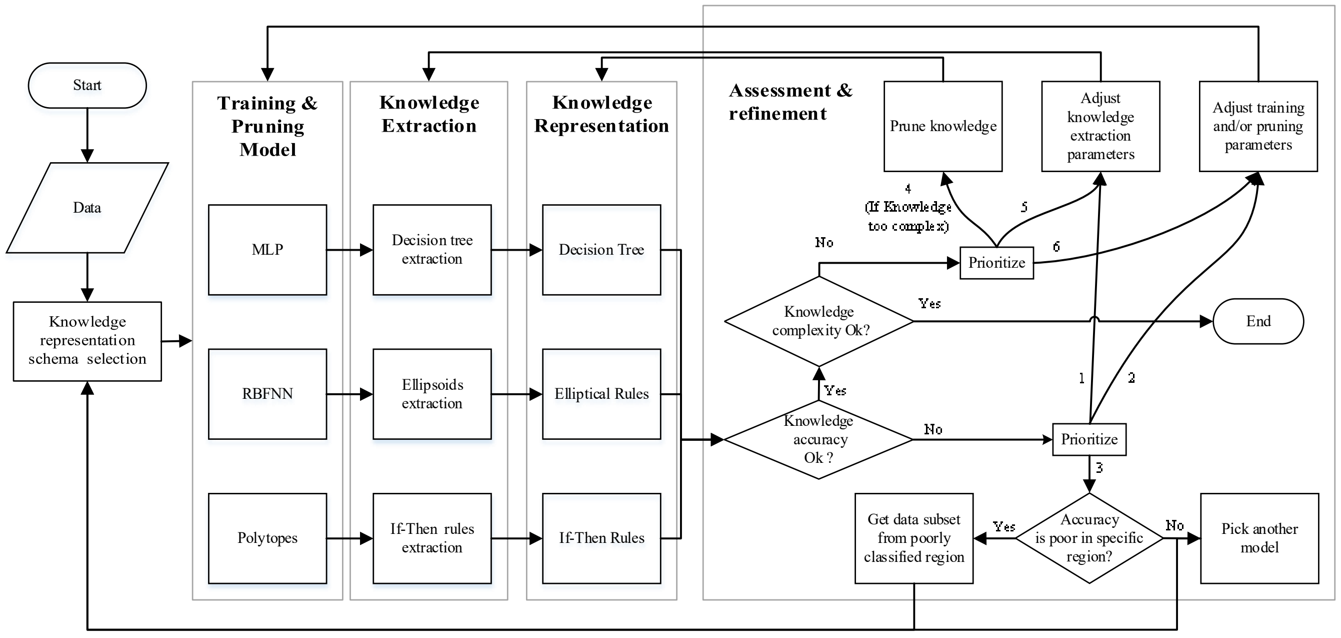

Fig. 6. Methodology for knowledge extraction.

If a pruning procedure is excessive, model accuracy is lowered too much. Finally, if the previous steps do not help, the model itself should be retrained (Fig. 6 – arrow 2), but only if we see that the difference between accuracy of classifier and knowledge accuracy is small, hence by getting a better performing classifier we can obtain better performing knowledge (in terms of classification). In cases, when knowledge performance is good, but knowledge is not comprehensible, i.e., too complex – several options exist. First, (Fig. 6 – arrow 4) knowledge pruning can be performed. In case of a classification decision tree, standard existing decision tree pruning approaches can be applied. In case of If–Then or elliptical rules, manual selection of least important rules should be performed with subsequent elimination. Most obvious criteria for least important rule can be either count of training data samples covered, size of area covered by rule or mixture of both.

If such knowledge pruning fails, then either knowledge extraction parameters should be changed to be more strict (Fig. 6 – arrow 5), or pruning itself has to be tuned (Fig. 6 – arrow 6) for removal of larger count of neurons (or acquisition of smaller number of polytopes). Finally, if nothing helps, the model itself should be retrained; of course, it should be made simpler so that we will extract less complex knowledge.

Alternative course of actions should be performed in case when the previous steps returned no results. The alternative (Fig. 6 – arrow 3) is the assessment of model performance in regard to input data. If there is only a specific small region of input data space that is poorly classified, then for only that specific region another classifier can be trained and knowledge for that data subset extracted. If this is not the case, then the whole classifier should be discarded and a new cycle of model training, pruning, knowledge extraction and assessment performed.

V. Conclusion

The present paper covers the developed methodology of knowledge extraction from trained ANN classifiers, specifically:

- highlights comprehensibility vs accuracy trade-off;
- gives recommendations for model and knowledge representation schema selection;
- for decision tree extraction methodology utilizes ANN pruning algorithm as a knowledge complexity control method;
- algorithms for elliptical and If-Then rule extraction are presented;
- strengths and weaknesses of all algorithms are listed and recommendations are given for selection of one over the other;
- presents an overall workflow combining the above-mentioned algorithms;
- presents a workflow for extracted knowledge extraction and assessment.

Contribution of the present paper is the following: an overview of algorithms capable of knowledge extraction along with their brief description; an overview of the types of knowledge representation schemas, summarization of their strengths and weaknesses; presentation of the developed methodology with guidelines for selection of algorithms and rule representation schemas.

The methodology highlights decisions a user should pay attention to and provides guidelines for extracted knowledge assessment and refinement. The methodology presents a "checklist" for mitigation of problems with extracted knowledge highlighting steps necessary to achieve the end goal. It is specifically underlined that end goals are contradictory in nature – rule understandability vs rule classification performance.

References

[1] H. Jacobsson, "Rule Extraction from Recurrent Neural Networks: A Taxonomy and Review," *Neural Computation,* vol. 17, no. 6, pp. 1223–1263, 2005. https://doi.org/10.1162/0899766053630350

[2] R. Setiono and H. Liu, "NeuroLinear: From neural networks to oblique decision rules," *Neurocomputing*, vol. 17, no. 1, pp. 1–24, 1997. https://doi.org/10.1016/S0925-2312(97)00038-6

[3] J. R. Zilke, E. L. Mencia and F. Janssen, "DeepRED–Rule extraction from deep neural networks," in *T. Calders, M. Ceci, D. Malerba (Eds.): Discovery Science 19th InternationalConference, DS 2016, Bari, Italy, Oct. 19–21, 2016, Proceedings*, LNAI 9956, pp. 457–473, 2016. https://doi.org/10.1007/978-3-319-46307-0_29

[4] M. Sato and H. Tsukimoto, "Rule extraction from neural networks via decision tree induction," in *Proceedings IJCNN'01. International Joint Conference on Neural Networks, Washingtom, DC, July 15–19, 2001*, IEEE, vol. 3, pp. 1870–1875, 2001. https://doi.org/10.1109/ijcnn.2001.938448

[5] S. I. M. Kamruzzaman and M. M. Islam, "An algorithm to extract rules from artificial neural networks for medical diagnosis problems," *International Journal of Information Technology*, vol. 12, no. 8, pp. 41–59, 2006.

[6] E. R. Hruschka and N. F. F. Ebecken, "Extracting rules from multilayer perceptrons in classification problems: A clustering-based approach," *Neurocomputing*, vol. 70, no. 1-3, pp. 384–397, 2006. https://doi.org/10.1016/j.neucom.2005.12.127

[7] P. Shinde, "Data mining using artificial neural network tree," *IOSR Journal of Engineering*, pp. 9–12, 2012.

[8] J. M. Benítez, J. L. Castro and I. Requena, "Are artificial neural networks black boxes?," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 1156–1164, 1997. https://doi.org/10.1109/72.623216

[9] H. Tsukimoto, "Extracting rules from trained neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 377–389, 2000. https://doi.org/10.1109/72.839008

[10] M. W. Craven, "*Extracting comprehensible models from trained neural networks*," Ph.D. thesis, University of Wisconsin–Madison, Madison, USA, 1996.

[11] S. Thrun, "Extracting Rules from Artificial Neural Networks with Distributed Representations," in *NIPS'94 Proceedings of the 7th Int. Conf. on Neural Information Processing Systems*, pp. 505–512, 1994.

[12] M. G. K. T. Augasta, "Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems," in *Neural Processing Letters*, vol. 35(2), pp. 131–150, 2012. https://doi.org/10.1007/s11063-011-9207-8

[13] K. Sethi, D. Mishra and B. Mishra, "KDRuleEx: A Novel Approach for Enhancing User Comprehensibility Using Rule Extraction," in *Third International Conference on Intelligent Systems Modelling and Simulation*, Kota Kinabalu, Malaysia, Feb. 8–10, pp. 55–60, 2012. https://doi.org/10.1109/ISMS.2012.116

[14] N. Barakat and J. Diederich, "Eclectic Rule-Extraction from Support Vector Machines," *International Journal of Computational Intelligence*, pp. 59–62, 2005.

[15] M. R. A. Iqbal, "Eclectic Rule Extraction from Neural Networks Using Aggregated Decision Trees," in *7th International Conference on Electrical & Computer Engineering (ICECE)*, pp. 129–132, 2012. https://doi.org/10.1109/icece.2012.6471502

[16] A. B. Tickle, “DEDEC: Decision Detection by rule extraction from neural networks,” Technical Report NRC QUT, Queensland University, 1994.

[17] M. W. Craven and J. W. Shavlik, “Learning Symbolic Rules Using Artificial Neural Networks,” in *Proceedings of the Tenth International Conference on Machine Learning*, pp. 73–80, 1993. https://doi.org/10.1016/B978-1-55860-307-3.50016-2

[18] R. Setiono, “Understanding Neural Networks via Rule Extraction,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, Aug. 20–25, pp. 480–485, 1995.

[19] A. Bondarenko, A. Borisovs and L. Aleksejeva, “Neurons vs Weights Pruning in Artificial Neural Networks,” in *Environment. Technology. Resources: Proceedings of the 10th International Scientific and Practical Conference*, Rezekne, Latvia, June 18–20, 2015, vol. 3, pp. 22–28. 2015. https://doi.org/10.17770/etr2015vol3.166

[20] A. Bondarenko and A. Borisovs, “Artificial Neural Network Generalization and Simplification via Pruning,,” *Information Technology and Management Science*, vol. 17, no. 1, pp. 132–137, 2014. https://doi.org/doi:10.1515/itms-2014-0020

[21] A. Bondarenko, L. Aleksejeva, V. Jumutcs and A. Borisovs, “Classification Tree Extraction from Trained Artificial Neural Networks,” *Procedia Computer Science*, vol. 104, pp. 556–563, 2017. https://doi.org/10.1016/j.procs.2017.01.172

[22] A. Bondarenko and A. Borisovs, “The Extraction of Elliptical Rules from the Trained Radial Basis Function Neural Network,” *Information Technology and Management Science*, vol. 15, pp. 161–165, 2012. https://doi.org/doi:10.2478/v10313-012-0027-2

[23] A. Bondarenko and A. Borisovs, “Elliptical Rule Extraction from a Trained Radial Basis Function Neural Network,” in *The 6th Int. Conf. Applied Information and Communication Technology*, Jelgava, Latvia, Apr. 25–26, 2013. ISSN 2255-8586.

[24] A. Bondarenko and V. Jumutcs, “Extraction of interpretable rules from piecewise-linear approximation of a nonlinear classifier using clustering-based decomposition,” in *Proceedings of 10th WSEAS Int. Conf. Artificial Intelligence,Knowledge Engineering and Data Bases (AIKED 2011)*, Cambridge, UK, Feb. 22, 2011, pp. 145–149, 2011.

[25] A. Bondarenko and A. Borisov, “Knowledge extraction from piecewise-linear approximation of multi-surface classifier,” in *Int. Conf. Information Intelligent Systems*, Kharkiv, Ukraine, April 17−19, 2012, pp. 5–6, 2012.

Andrey Bondarenko received BsCompSc degree (2004) at the University of Latvia, *Mg. sc. ing.* degree (2006) at the Transport and Telecommunication Institute in Riga. At present, he is pursuing PhD degree at Riga Technical University.
He works as a Data Scientist at CTCO in the area of document, image analysis and introduction of machine and deep learning to enterprise systems. He has more than ten years of experience in enterprise-grade IT system development. Research interests include machine learning, deep learning, and intelligent systems of information processing under conditions of a priori uncertainty and incompleteness.
Email: andrejs.bondarenko@gmail.com
ORCID iD: https://orcid.org/0000-0002-4103-6814

Ludmila Aleksejeva is a Professor of the Faculty of Computer Science and Information Technology at Riga Technical University (Latvia). She received her *Dr. sc. ing.* degree from Riga Technical University in 1998.
Current research interests include decision-making techniques using different types of information, intelligent decision support systems, data mining and decision support methods collaboration.
Email: ludmila.aleksejeva@rtu.lv
ORCID iD: https://orcid.org/0000-0003-0900-3868