Contributed article

Rule Extraction by Successive Regularization

Masumi Ishikawa

Department of Brain Science and Engineering

Graduate School of Life Science and Systems Engineering

Kyushu Institute of Technology

680-4 Kawazu, Iizuka, Fukuoka 820-8502, Japan

Acknowledgements: I would like to thank Hirotsugu Yamamoto, Hiroki Ueda and Naoe Fukuyama, former students in my laboratory, for their contributions in this research. This research was partly supported by Grant-in-Aid for Scientific Research(C)(07680404) from the Ministry of Education, Science, Sports and Culture, Japan.

Correspondence:

Masumi Ishikawa

Department of Brain Science and Engineering

Graduate School of Life Science and Systems Engineering

Kyushu Institute of Technology

680-4 Kawazu, Iizuka, Fukuoka 820-8502, Japan

Phone: +81-948-29-7718 Fax: +81-948-29-7709

E-mail: ishikawa@ces.kyutech.ac.jp

Rule Extraction by Successive Regularization

abstract

Knowledge acquisition is, needless to say, important, because it is a key to the solution to one of the bottlenecks in artificial intelligence. Recently knowledge acquisition using neural networks, called rule extraction, is attracting wide attention because of its computational simplicity and ability to generalize. Proposed in this paper is a novel approach to rule extraction named successive regularization. It generates a small number of dominant rules at an earlier stage and less dominant rules or exceptions at later stages. It has various advantages such as robustness of computation, better understanding, and similarity to child development. It is applied to the classification of mushrooms, the recognition of promoters in DNA sequences and the classification of irises. Empirical results indicate superior performance of rule extraction in terms of the number and the size of rules for explaining data.

keywords: rule extraction, successive regularization, rules and exceptions, structural learning, forgetting, classification

List of symbols

symbol	description
J	quadratic criterion in back propagation learning
J_f	total criterion in the learning with forgetting
o_k	output of output unit k
t_k	target of output unit k
η	learning rate
ε	amount of forgetting
λ	regularization parameter
θ	threshold in the learning with selective forgetting
w_{ij}	the connection weight from unit j to unit i
Δw_{ij}	the amount of change of the connection weight w_{ij}
$\wedge$	conjunction or logical product
$\vee$	disjunction or logical sum
$\neg$	negation

1 Introduction

Knowledge acquisition is, needless to say, important, because it is a key to the solution to one of the bottlenecks in artificial intelligence. Numerous studies on knowledge acquisition have been made in machine learning such as ID3 and C4.5[23]. Recently knowledge acquisition using neural networks, called rule extraction, is attracting wide attention because of its computational simplicity and ability to generalize.

The author has proposed a learning algorithm named a structural learning with forgetting(SLF), which generates a skeletal structured network reflecting regularities in data[12]. It was originally aimed at improving generalization performance by pruning unnecessary connections due to forgetting. SLF is also effective in rule extraction and has demonstrated its effectiveness using various databases as will be shown in section 3.

In the present paper the author proposes a novel approach to rule extraction named successive regularization, which is a succession of learning with a decreasing value of regularization parameter[11]. At an earlier stage, learning with a large value of regularization parameter is carried out to obtain dominant rules. At later stages, the regularization parameter is decreased step by step to obtain less dominant rules or exceptions by freezing connection weights corresponding to previously extracted rules. It, therefore, generates a small number of dominant rules at an earlier stage and less dominant rules or exceptions at later stages.

Rule extraction by successive regularization has the following advantages. Firstly, rules can be obtained with higher reliability than by learning with conventional regularization. In conventional regularization, it generates a whole set of rules by using a fixed small value of a regularization parameter. Since the size of resulting networks is large, the resulting network structure heavily depend on initial connection weights due to a local minima problem. In learning with successive regularization, on the other hand, major rules are generated with robustness because the resulting network size is small. Minor rules or exceptions are also generated with higher reliability compared to conventional regularization using the same value of a regularization parameter, because previously extracted rules are frozen and only the small number of connections are modified at each later stage in successive regularization.

Secondly, extracted rules in the form of dominant rules and their exceptions are expected to be understood better than a large number of rules on a level. This hierarchical structure of rules agrees with tendency of humans to interpret data as a combination of a small number of dominant rules and their exceptions. This characteristic can be observed in various cognitive domains. One example is that humans adopt two basic principles in the formation of categories: cognitive economy combined with vertical and horizontal structure in the perceived world[25]. Another example is that in language acquisition simpler rules and forms are applied first before more complex ones[22].

Thirdly, hierarchical structure of rules seems to agree with developmental processes of children. A first example is an overgeneralization of past tense of verbs at some stage. However, hearing a lot of irregular past tense forms at later stages, they drive out the overgeneralized forms[21]. A second example is an overextension in concept formation. For children at some stage a “dog” means an “animal,” but they constrain extension of a dog at later stages[4]. A third example is understanding of balance scale problems. Children of different ages differ in the rules they use. Younger children use only the amount of weight and ignore the amount of distance, and older children encode both weight and distance dimensions correctly[27].

Lastly, empirical results presented in the paper suggest superior performance of rule extraction in terms of the number and the size of rules for explaining data. This may be attributed to the first advantage.

It should be emphasized that rule extraction by successive regularization proposed here is unique in that it generates a set of rules in a *hierarchical* way. To my knowledge, there has been no neural networks study on rule extraction generating a hierarchical structure of rules. Another point which needs attention is that a major purpose of the present paper is to obtain a compact set of rules which best explains the entire samples. Because of this, conventional methods of model evaluation such as crossvalidation are not adopted here, which makes direct comparison of generalization performance with other methods difficult.

The following section presents a structural learning with forgetting[12] as an effective method for learning with regularization. It is followed by brief overviews of rule extraction methods. A detailed procedure for rule extraction by SLF and its extension to SLF with successive regularization are described. Its applications to the classification of mushrooms, the recognition of promoters in DNA sequences, and the classification of irises are also provided.

2 Structural learning with forgetting

It is well known that back propagation(BP) learning suffers from such difficulties as prior specification of network structure and interpretation of hidden units. To overcome mainly the former difficulty, various structural learning methods have been proposed. They are roughly classified into four categories: the addition of a regularizer to the criterion of mean square error(MSE)[12][24], the deletion of hidden units with small contribution to MSE, the deletion of connections with small contribution to MSE[17], and the incremental increase in the number of hidden units until MSE becomes sufficiently small.

Empirical comparative studies suggest the superiority of SLF[12] in the first category over other methods. The criterion function of SLF is,

$$J_f = J + \varepsilon' \sum_{i,j} |w_{ij}| = \sum_k (o_k - t_k)^2 + \lambda \sum_{i,j} |w_{ij}| \tag{1}$$

where w_{ij} is the connection weight from unit j to unit i, o_k is the output of output unit k, t_k is its target, and λ is a regularization parameter. The weight change, Δw_{ij}, is,

$$\Delta w_{ij} = -\eta \frac{\partial J_f}{\partial w_{ij}} = \Delta w'_{ij} - \varepsilon\, \mathrm{sgn}(w_{ij}) \tag{2}$$

where $\Delta w'_{ij}(= -\eta \frac{\partial J}{\partial w_{ij}})$ is the weight change due to BP learning, η is a learning rate, $\varepsilon(= \eta\lambda)$ is the amount of forgetting or decay, and $\mathrm{sgn}(w_{ij})$ is a sign function, i.e., 1 when w_{ij} is positive and –1 otherwise.

A key idea of SLF is constant decay of connection weights in contrast to frequently used exponential decay[24]. Due to this constant weight decay in Eq.(2), unnecessary connections fade away and a skeletal network reflecting regularities in data emerges. This is a big advantage of learning with forgetting.

The learning with forgetting, however, causes two difficulties. The first is the emergence of distributed representations on hidden layers, which hinders interpretation of hidden units and rule extraction. Empirically, a distributed representation is mostly a combination of a major salient representation and minor representations. The learning with hidden units clarification, which succeeds the learning with forgetting, frequently suppresses minor representations, hence prevents the emergence of distributed representations.

The learning with hidden units clarification using the following criterion dissipates distributed representations by forcing each hidden unit to be fully active or inactive.

$$J_h = J + c \sum_i min\{1 - h_i, h_i\} \tag{3}$$

where h_i is the output of hidden unit i satisfying $h_i \in [0, 1]$, and c is a relative weight of the penalty term. The minimization of the penalty term can easily be carried out by taking the derivative of $1 - h$(if $h \geq 0.5$) or h(if $h < 0.5$) with respect to the connection weight w_{ij}.

The second difficulty is that MSE by the learning with forgetting is still larger than that by BP learning. From the viewpoint of AIC[1], it deteriorates the goodness of fit of a model. The learning with *selective* forgetting, which succeeds the learning with forgetting and learning with hidden units clarification, solves this deficiency.

MSE by the learning with forgetting is larger than that by BP learning, because the former minimizes the total criterion J_f instead of the quadratic criterion J. The following criterion makes only the connection weights decay whose absolute values are below the threshold, θ.

$$J_s = J + \varepsilon' \sum_{|w_{ij}|<\theta} |w_{ij}| \tag{4}$$

The penalty term in Eq.(4) makes MSE much smaller than that by the learning with forgetting, because the summation is restricted only to weak connections. It also prevents the revival of deleted connections.

The above procedure of SLF is summarized in the following 3 steps.

1. Apply the learning with forgetting to obtain a rough skeletal network structure.

2. Apply both the learning with forgetting and that with hidden units clarification to dissipate distributed representations.

3. Apply both the learning with selective forgetting and that with hidden units clarification to get better learning performance in terms of MSE.

A question might arise when to go to the next step in the above procedure. This is not serious. When training is carried out in sufficient number of iterations, the connection weights stabilize. This is the time to go to the next step.

Another question is how to determine various parameters for training. The determination of the regularization parameter, λ, or the amount of forgetting, ε, is crucial. If λ or ε is too large, even the necessary connections fade away, causing severe degradation of MSE. On the other hand, if it is too small, unnecessary connections survive, resulting a network far from skeletal. Mean prediction error (MPE), i.e., mean square error for test data, and various information criteria such as AIC[1] and NIC[19] help determine the value of a regularization parameter.

On the other hand, the determination of parameters c in Eq.(3) and θ in Eq.(4) are not crucial. The value of c can be arbitrary, provided it is large enough to render outputs of hidden units binary. The value of θ is set such that there is no significant connection weight whose absolute value is smaller than θ.

3 Overview of rule extraction

Andrews et al. carried out an extensive survey of rule extraction using neural networks, and reported performance evaluation using the MONKs problems and the classification of mushrooms[2][3].

Sestito and Dillon proposed BRAINNE(Building Representations for AI using Neural NEtworks), which adopted an expanded 3-layer network architecture with additional inputs representing target outputs[26]. BRAINNE can also be extended to data with continuous valued inputs. In the classification of mushrooms and irises, BRAINNE generates more rules than SLF; 46 rules for mushrooms and 11 rules for irises.

Towell and Shavlik proposed the following framework[28]: inserting knowledge into a neural network using KBANN(Knowledge BAsed Neural Networks), training it with a set of data, and extracting rules from a resulting network. The extraction phase is the most difficult part. They proposed a Subset algorithm and a MofN algorithm for the extraction phase. These algorithms are applied to the recognition of promoters in DNA sequences and the MONKs problems. Detailed analyses of the performance of rule extraction are also presented[29].

Fu proposed KBCNN(Knowledge-Based Conceptual Neural Network)[7] similar to the framework by Towell and Shavlik[28]. During a training phase, it uses heuristic procedures such as ignoring weak connections. During a rule extraction phase, a KT algorithm, which is derived from *Knowledgetron* and similar to Subset, is proposed.

Fu proposed another method without prior rules[8]. During a rule extraction phase it uses the KT method. Heuristic search is carried out using three kinds of thresholds, which affect the rules extracted. The proposed method is applied to three kinds of tasks, i.e., the classification of irises, hepatitis prognosis prediction and hypothyroid diagnosis. The first task has continuous valued inputs, and the last two have both continuous and discrete valued inputs.

Kasabov proposed REFuNN for extracting fuzzy rules from fuzzy neural networks[15]. In contrast to SLF, extracted rules by C4.5, BRAINNE, KT and REFuNN using data with continuous valued inputs have only hyper-rectangle if-parts, i.e., each attribute has a lower bound or an upper bound or both. In other words, linear combination of attributes is not allowed in rule expression.

Historically, rule extraction and structural learning have developed almost independently. Therefore, effective structural learning methods have not been implemented in most methods for rule extraction. This has necessarily made rule extraction phase from trained neural networks the most difficult part.

In the author's opinion, structural learning methods which are capable of generating skeletal structured networks with easy interpretation of hidden units are effective in rule extraction. Resulting network structure by SLF is skeletal and hidden units are binary owing to learning with hidden units clarification. This makes the extraction phase by SLF quite simple. This simplicity is a big advantage over other methods, because ad hoc procedures such as ignoring weak connections are no longer necessary. SLF can also dispense with prior information such as pre-selection of attributes and initial theories.

SLF has been applied to rule extraction and has demonstrated its effectiveness in various tasks such as the classification of mushrooms [9][11][12][14], the MONKs problems [9][10][14], the recognition of promoters in DNA sequences [9][10][14], the classification of irises [10][13][14], and diagnosis of thyroid functioning [13][14]. The first 3 have discrete valued inputs and the last 2 have continuous valued inputs.

Recently, special sessions and panel sessions on rule extraction have taken place at various international conferences such as ICNN'96[10], ICONIP'97[13] and IJCNN'98 reflecting wide interests in this area.

4 Rule extraction by successive regularization

A typical task considered here is to classify samples into predetermined number of classes. Each sample has a set of attributes, and each attribute has an attribute value. Firstly, a procedure for rule extraction by SLF from data with discrete valued inputs and outputs is given[14]. A unit with a discrete value at an input or an output layer can be converted into a set of binary units. Therefore, it is assumed here, without loss of generality, that all input and output units are binary.

Given a regularization parameter, λ, or the amount of forgetting, ε, rule extraction is carried out in the following 7 steps. Steps 2 through 4 correspond to 3 training steps in section 2.

1. Consider a 3-layer or 4-layer network depending on the complexity of the task. Units in an input layer represent attribute values. Each unit in the first hidden layer correspond

to an attribute. Output units represent categories of classification. Connections between adjacent layers are fully connected initially except between the input layer and the first hidden layer; each unit in the first hidden layer is connected only to the input units representing its attribute values.

2. Train a neural network by the learning with forgetting to obtain a rough skeletal network structure.

3. Train it by both the learning with forgetting and that with hidden units clarification to dissipate distributed representations.

4. Train it by both the learning with selective forgetting and that with hidden units clarification to get smaller MSE. At this step, all the hidden units are binary.

5. Represent each hidden unit as a Boolean function of input units. Suppose some attribute has m attribute values. Among m input units corresponding to this attribute, only one of them is active at a time. It means that among 2^m cells in a Karnaugh map, only m cells have values of 1 or 0, and other cells are labeled as *don't care.* Introducing *don't care* is a basic standard procedure in mathematical logic for obtaining the simplest Boolean function from a given Karnaugh map.

6. Represent each output unit as a Boolean function of hidden units. Suppose there are n attributes. It can happen that among 2^n cells in a Karnaugh map, some don't have corresponding samples. This absence of samples happens more frequently as n becomes larger. Unless we treat those cells without samples as *don't care*, the resulting rules are not sufficiently general, i.e., resulting rules become unnecessarily complex. Although the use of *don't care* is a basic standard procedure in mathematical logic, their use has been neglected in rule extraction.

7. By combining the above two Boolean functions, each output unit can be represented as a Boolean function of input units. These are the rules we seek for.

The above procedure is not applicable, as it is, to rule extraction from data with continuous valued inputs because of inherent difficulty in dealing with continuous valued inputs. To overcome this difficulty, this paper proposes to train neural networks with various degrees of complexity. The degree of complexity, here, is defined by the maximum number of incoming connections into each hidden unit. From among them, the one with the smallest AIC is selected as the most appropriate. Since outputs of hidden units are binary owing to the learning with hidden units clarification, incoming connection weights into each hidden unit determine the corresponding discriminating hyperplane. A logical combination of these hyperplanes provides rules.

A newly proposed method of rule extraction by successive regularization is quite different from the previous ones. It is a succession of learning with a decreasing regularization parameter. At each stage the previous procedure composed of 7 steps including three stages of learning is carried out. At an earlier stage, learning with a large value of a regularization parameter is carried out to obtain dominant rules. At later stages, the value of a regularization parameter is decreased step by step to obtain rules with finer details by freezing connection weights corresponding to previously acquired rules. The total criterion function can generally be written as,

$$M = J + \lambda E_w \tag{5}$$

where E_w is a regularizer and is a function of connection weights, and λ is a regularization parameter. Rule extraction by successive regularization is carried out in the following 5 steps. In step 2 and step 4 below, the entire 7 steps for rule extraction described in the beginning of this section are carried out.

1. Set $i = 1$.
2. Acquire rules by learning with a large value of a regularization parameter, λ_i. They are dominant rules.
3. $i = i + 1$

4. Acquire rules with a smaller value of a regularization parameter, λ_i, by freezing the connection weights corresponding to previously acquired rules.

5. If all the training samples are learned satisfactorily, stop. Else go to step 3.

Although an idea of successive regularization is simple, it is quite effective and has various advantages over conventional methods as shown in Introduction. Learning with successive regularization seems to resemble the cascade-correlation learning[6] in that network grows step by step by freezing previously learned connection weights. The latter is carried out by simply adding hidden units using the same learning method. On the other hand, the learning with successive regularization is carried out by using the same network architecture with different regularization parameters at each step. Although the resulting network is simple at an early stage, the whole network architecture is retrained all the time with most connections having zero weights.

A question might arise on how to determine a regularization parameter at each step. At the first step it suffices to select a regularization parameter generating a rough skeletal structure, and its determination is not so difficult. At later steps it is necessary to select a regularization parameter generating a different structure from that in the previous stage. Its determination is not so difficult either, although it affects the resulting hierarchical structure of rules.

5 Classification of mushrooms

A mushroom database contains 8124 samples, each with 22 attributes in Table 1[20]. As shown in Table 1 each attribute has from 2 to 12 attribute values; in total there are 126 attribute values. In addition to these attribute values, each sample is given a categorical value, i.e., edible or poisonous. The task here is to classify mushrooms into these two categories. Table 2 presents major attributes and their attribute values. 812 out of 8124 samples are randomly chosen for training.

A 3-layer network is adopted here; 1 output unit indicates edible or poisonous, 22 hidden units correspond to 22 attributes, and 126 input units represent their attribute values. Each hidden unit is connected only to the relevant input units. This special architecture corresponds to an implicit assumption frequently adopted in machine learning, i.e., each sample has attributes and each attribute has its attribute value.

Learning is carried out by SLF, which uses a regularizer of the sum of the absolute values of connection weights in addition to an ordinary criterion function. Since the value of a regularization parameter, λ, is directly related to the amount of forgetting, ε, i.e. $\varepsilon = \eta\lambda$, it suffices to show either one of them.

In the first stage, SLF with the regularization parameter, $\lambda = 3 \times 10^{-3}$, generates the network in Figure 1 with 2 attributes: odor and spore-print-color. From Figure 1 it is almost straightforward to represent these two attributes as a function of input variables by taking into account the fact that only one of the attribute values is active in each attribute. Since this is the first example, the Karnaugh map representing the relation between the hidden variable, y_4, and the corresponding 6 input variables is shown in Table 3 for better understanding. It is to be noted that all but six cells are *don't care*. The simplified Boolean functions for $\bar{y}_4$ and y_{19} are,

$$\begin{aligned} \bar{y}_4 &= \text{almond} \vee \text{anise} \vee \text{none} \\ y_{19} &= \text{green} \end{aligned} \tag{6}$$

where $\bar{y}_4$ stands for the negation of y_4.

Table 4 shows the Karnaugh map obtained from the network in Figure 1. A *don't care* in Table 4 indicates the absence of samples in this cell. The extracted rule for edible mushrooms, therefore, is,

$$\bar{y}_4 \wedge \bar{y}_{19} = (\text{odor} = \text{almond} \vee \text{anise} \vee \text{none}) \wedge (\text{spore-print-color} \neq \text{green}) \tag{7}$$

This rule misclassifies 5 poisonous mushrooms as edible out of 812 training samples and 48 poisonous mushrooms as edible out of 8124 samples.

In the second stage, SLF with the regularization parameter, $\lambda = 2 \times 10^{-3}$, is applied while freezing the connection weights in the first stage. It generates the network in Figure 2 with

4 attributes: odor, gill-size, stalk-surface-below-ring and spore-print-color. It is fairly easy to represent these attributes as a function of input variables.

$$\begin{aligned}\bar{y}_4 &= \text{almond} \vee \text{anise} \vee \text{none} \\ y_7 &= \text{broad} \\ y_{12} &\neq \text{scaly} \\ y_{19} &= \text{green} \qquad (8)\end{aligned}$$

Table 5 shows the Karnaugh map obtained from the network in Figure 2. The extracted rule for edible mushrooms is,

$$\begin{aligned}\bar{y}_4 \wedge \bar{y}_{19} \wedge (y_7 \vee y_{12}) &= (\textbf{odor} = \textbf{almond} \vee \textbf{anise} \vee \textbf{none}) \\ &\wedge(\text{spore-print-color} \neq \text{green}) \\ &\wedge\{(\text{gill-size} = \text{broad}) \vee (\text{stalk-surface-below-ring} \neq \text{scaly})\} \quad (9)\end{aligned}$$

The rule in Eq.(9) makes no classification error for 812 training samples and misclassifies 8 poisonous mushrooms as edible for the total samples. The bold face part corresponds to the rule in Eq.(7). The additional term, (gill-size = broad) $\vee$ (stalk-surface-below-ring $\neq$ scaly), makes the input space for edible mushrooms smaller to further decrease the number of misclassifications.

In the third stage, the number of training samples is increased to 820 by merging the misclassified 8 samples in the second stage. SLF with the regularization parameter, $\lambda = 0.2 \times 10^{-3}$, is applied while freezing the connection weights in the first and second stages. It generates the network in Figure 3 with 6 attributes: odor, gill-size, stalk-shape, stalk-surface-below-ring, spore-print-color and population. It is easy to represent these attributes as a function of input variables.

$$\begin{aligned}\bar{y}_4 &= \text{almond} \vee \text{anise} \vee \text{none} \\ y_7 &= \text{broad} \\ y_9 &= \text{enlarging} \\ y_{12} &\neq \text{scaly}\end{aligned}$$

$$\begin{aligned} y_{19} &= \text{green} \\ y_{20} &\neq \text{clustered} \end{aligned} \tag{10}$$

Table 6 shows the Karnaugh map obtained from the network in Figure 3. The extracted rule for edible mushrooms is,

$$\begin{aligned} \bar{y}_4 \wedge \bar{y}_{19} \wedge \{y_7 \vee (y_{12} \wedge y_{20})\} = & \quad (\textbf{odor} = \textbf{almond} \vee \textbf{anise} \vee \textbf{none}) \\ & \wedge \quad (\text{spore-print-color} \neq \text{green}) \\ & \wedge \quad \{(\text{gill-size} = \text{broad}) \vee \\ & \quad\quad (\text{stalk-surface-below-ring} \neq \textbf{scaly}) \wedge \\ & \quad\quad (\text{population} \neq \text{clustered})\} \end{aligned} \tag{11}$$

It is noteworthy that although 6 attributes are included in the resulting network in Figure 3, only 5 attributes appear in the final rule in Eq.(11). This is due to the existence of *don't care* cells in Table 6.

The resulting rule in Eq.(11) makes no classification error not only for 820 training samples but also for the entire samples. The bold face part corresponds to the rule in Eq.(9). The additional term, (population $\neq$ clustered), makes the input space for edible mushrooms smaller. In Eq.(9), 8 poisonous mushrooms are misclassified as edible. By changing (gill-size = broad) $\vee$ (stalk-surface-below-ring $\neq$ scaly) into (gill-size = broad) $\vee$ (stalk-surface-below-ring $\neq$ scaly) $\wedge$ (population $\neq$ clustered) these 8 misclassified mushrooms are classified correctly. To my knowledge, a set of rules with 6 attributes, which can explain all the mushroom samples, has been the best one[5], but Eq.(11) is even better because the number of attributes is 5.

In the Karnaugh maps there is 1 *don't care* cell in the first stage, and there are 8 *don't care* cells in the second stage. The resulting rules happen to be the same as when *don't care* cells are not properly treated. As the number of attributes increases, the number of *don't care* cells increases. In the third stage the number of *don't care* cells is 49 out of 64 cells. If *don't care* cells are not properly treated, the resulting rule is,

$$\bar{y}_4 \wedge \bar{y}_{19} \wedge \{y_7 \wedge y_9 \wedge (y_{12} \vee y_{20}) \vee y_{12} \wedge y_{20}\} \tag{12}$$

This is easily given by replacing all *don't care* cells by 0. The resulting rule is different from Eq.(11) and contains 6 instead of 5 attributes. This indicates the importance of treating *don't care* cells properly.

Table 7 summarizes the results by BP learning, SLF and ID3. It well indicates the superiority of SLF with successive regularization over other methods; when the number of attributes is 4, SLF is superior to ID3 in terms of the number of classification errors. It is also to be noted that BP learning, in contrast to SLF and ID3, cannot extract rules and requires all the attribute values for classifying test samples due to distributed representation of the resulting networks.

When only the final rule in Eq.(11) is given, we still can understand it, but there's no way to know which part is dominant. On the other hand, the succession of three rules, i.e., Eqs.(7)(9)(11), is much easier to understand, because information on which part is dominant and which part is less dominant is available. This is a major advantage of rule extraction by successive regularization.

Another advantage of successive regularization is that it is computationally robust compared to conventional regularization. In the first and second stages, the resulting networks are almost the same for different initial connection weights. In the third stage the resulting networks are slightly varying. On the other hand, when SLF with a constant regularization parameter is applied, results are much varying. Even the number of attributes differ from 5 to 8 for different initial connection weights as shown in Table 8.

In a structural learning with forgetting(SLF) using a specified regularization parameter, connections which do not contribute to mean square output error fade away. Therefore, from an engineering point of view, the smaller a regularization parameter is, the more precise the mapping from input to output of a network becomes. From the viewpoint of cognitive science, it could be suggested that the smaller the value of a regularization parameter is, the deeper the understanding of data becomes.

When the values of connection weights reaches a steady state, we terminate learning of that stage. A decrease of the value of a regularization parameter is done by hand. Experimental results indicate that, empirically, the network structure does not change for some range of a regularization parameter and at some value of a regularization parameter the network structure suddenly changes. A new value of a regularization parameter, therefore, is determined so as to generate a *different* network structure.

6 Recognition of promoters

The task here is the recognition of promoters in DNA sequences. A promoter is a site where the protein RNA polymerase binds to DNA. The database has 53 promoters and 53 non-promoters[20]. The input is 57 sequential DNA nucleotides. The location of each nucleotide is numbered between –50 and +7 with respect to a fixed reference point. As is well known, each location in a DNA sequence is represented by 4 input units, corresponding to 4 types of nucleotides, {A, G, T, C}. In total there are 228(57 × 4) input units. Figure 4 indicates the network architecture adopted here. The second hidden layer with two units corresponds to two intermediate concepts, i.e., *contact* and *conformation*, found in the existing domain theory[28]. Out of 106 samples, 95 samples are randomly chosen for training and 11 samples are used for test.

In the first stage, SLF with the regularization parameter, $\lambda = 1.8 \times 10^{-2}$, generates the network in Figure 5. There are 7 misclassifications for training samples. Three promoters are misclassified as non-promoters and 4 non-promoters are misclassified as promoters. The following rule for promoters is extracted from Figure 5.

$$(T_{-36} \wedge T_{-35}) \vee (T_{-36} \wedge G_{-34}) \vee (T_{-35} \wedge G_{-34}) \tag{13}$$

where T_{-36} is a simplified expression that the nucleotide T is located at –36 in a DNA sequence. Table 9 gives the alternative expression of this rule.

In the second stage, SLF with the regularization parameter, $\lambda = 10^{-2}$, is carried out while freezing the connection weights in the first stage. The network in Figure 6 is generated. The following rule and Table 10 are extracted from this network.

$$\{(\boldsymbol{T}_{-36} \wedge \boldsymbol{T}_{-35}) \vee (\boldsymbol{T}_{-36} \wedge \boldsymbol{G}_{-34}) \vee (\boldsymbol{T}_{-35} \wedge \boldsymbol{G}_{-34})\} \wedge \neg G_{-12} \tag{14}$$

where $\neg G_{-12}$ stands for the negation of G_{-12}, i.e., the nucleotide G is not located at –12 in a DNA sequence. The bold face part corresponds to the rule in Eq.(13). The addition of the term, $\neg G_{-12}$, makes the input space for promoters smaller, which decreases the number of misclassifications of promoters as non-promoters. In the modified rule, 4 promoters are misclassified as non-promoters, and 1 non-promoter is misclassified as a promoter.

In the third stage, SLF with the regularization parameter, $\lambda = 0.6 \times 10^{-2}$, is applied while freezing the connection weights in the first and second stages. The network in Figure 7 is generated. The following rule and Table 11 are extracted from this network.

$$\begin{aligned} &\{(K_{-36} \wedge \boldsymbol{T}_{-35}) \vee (K_{-36} \wedge \boldsymbol{G}_{-34}) \vee (\boldsymbol{T}_{-35} \wedge \boldsymbol{G}_{-34})\} \wedge \neg \boldsymbol{G}_{-12} \\ &\vee (T_{-38} \wedge K_{-36} \wedge \neg C_{-31}) \end{aligned} \tag{15}$$

The addition of the term, $(T_{-38} \wedge K_{-36} \wedge \neg C_{-31})$, makes the input space for promoters larger. Among the training samples, only 1 non-promoter is misclassified as a promoter.

In the 4th stage, SLF with the regularization parameter, $\lambda = 0.2 \times 10^{-2}$, is carried out while freezing the connection weights in the previous stages. The network in Figure 8 is generated. The following rule and Table 12 are extracted from this network.

$$\begin{aligned} &\{(\boldsymbol{K}_{-36} \wedge \boldsymbol{T}_{-35} \wedge \neg C_{-31}) \vee (\boldsymbol{K}_{-36} \wedge \boldsymbol{G}_{-34})\} \wedge \neg \boldsymbol{G}_{-12} \\ &\vee (\boldsymbol{T}_{-38} \wedge \boldsymbol{K}_{-36} \wedge \neg \boldsymbol{C}_{-31}) \vee (T_{-39} \wedge T_{-35}) \end{aligned} \tag{16}$$

The addition of the term, $\neg C_{-31}$, the deletion of the term, $(T_{-35} \wedge G_{-34})$, and the addition of the term, $(T_{-39} \wedge T_{-35})$, in Eq.(16) realize perfect classification.

Towell and Shavlik reported the results of comparative study on this task[28]. Initial domain knowledge used is represented by 12 rules and the number of antecedents is 77. They used a tenfold cross-validation method for evaluation. Average error rate is 2.4% for training samples and 3.8% for test samples by MofN and is superior to other methods such as Subset and C4.5. Fidelity, which shows the ability of the extracted rules to mimic the behavior of the network, is 99% and is also superior to Subset. The average number of rules and the number of antecedents are about 12 and 102, and is much superior to Subset, although symbolic methods such as C4.5 is superior.

Fu[7] applied KBCNN to the whole samples and extracted 14 rules with 3 recognition errors, although the resulting network makes no error. The lack of the fidelity is often observed when inappropriate learning methods are used.

Since the numbers of training samples used above are different from that in the proposed method, classification rates of the above methods cannot directly be compared with ours. However, the number of rules and rule complexity can still be compared with ours. They show the superiority of successive regularization in terms of the number and the simplicity of generated rules over other methods.

Table 13 summarizes the results of the recognition of promoters in DNA sequences. To be stressed here is that the rules obtained by SLF with successive regularization can completely recognize promoters by a simple set of rules. Its accuracy, however, cannot be directly compared with other methods, because we don't use tenfold cross-validation. Since all the hidden units are binary in the learning with successive regularization, the fidelity is perfect. The number of rules and the number antecedents are 4 and 15, respectively. They are much smaller than those by MofN and other methods.

7 Classification of irises

The task here is to classify 150 irises into three categories each with 50 samples: setosa, versicolor and virginica[20]. Each sample has 4 attributes with continuous attribute values i.e., sepal length(x_1), sepal width(x_2), petal length(x_3) and petal width(x_4). Networks of various degrees of complexity are trained using all the samples; the maximum number of incoming connections to each hidden unit is 1, 2, 3 or 4. Table 14 indicates that the network with at most 3 incoming connections to each hidden unit and with the regularization parameter λ=0.0001 is the best from the viewpoint of AIC. Figure 9 illustrates the corresponding network, which makes 2 classification errors. From this network, the following rules can be extracted.

$$\begin{aligned}(x_3 < 2.60) \wedge (x_2 - 3.57x_4 > 0.024) &\Rightarrow \text{setosa}\\ (x_2 - 3.57x_4 < 0.024) \wedge (x_2 - 0.28x_3 - 3.32x_4 > -4.34) &\Rightarrow \text{versicolor}\\ (x_1 - 3.97x_3 - 0.33x_4 < 16.26) &\Rightarrow \text{virginica} \qquad (17)\end{aligned}$$

Table 15 summarizes the performance of classification. Since the maximum number of incoming connections to each hidden unit is three, the resulting rules in Eq.(17) are not easy to understand. Furthermore, they make 2 classification errors out of 150 total samples. To overcome this difficulty, SLF with successive regularization is applied.

In the first stage, a simple network with only one incoming connection to each hidden unit is trained by SLF with the regularization parameter, λ=0.005. Figure 10 illustrates the resulting network. The extracted rules are,

$$\begin{aligned}(x_4 < 0.68) \wedge (x_3 < 4.78) &\Rightarrow \text{setosa}\\ (x_4 > 0.68) \wedge (x_3 < 4.78) &\Rightarrow \text{versicolor}\\ (x_4 > 0.68) \wedge (x_3 > 4.78) &\Rightarrow \text{virginica} \qquad (18)\end{aligned}$$

The rules in Eq.(18) make 7 classification errors out of 150 samples.

In the second stage, hidden units are added to the above network with each new hidden unit having at most two incoming connections. SLF with the regularization parameter, λ=0.0007 is carried out. Figure 11 delineates the resulting network. The extracted rules are,

$$\begin{aligned}(\boldsymbol{x_4} < \mathbf{0.68}) \wedge (\boldsymbol{x_3} < \mathbf{4.78}) &\Rightarrow \text{setosa}\\ (\boldsymbol{x_4} > \mathbf{0.68}) \wedge (x_2 - 3.30x_4 > -2.82) &\Rightarrow \text{versicolor}\\ (\boldsymbol{x_4} > \mathbf{0.68}) \wedge (\boldsymbol{x_3} > \mathbf{4.78}) &\Rightarrow \text{virginica} \qquad (19)\end{aligned}$$

where the bold face part corresponds to the rules in the first stage. The rules in Eq.(19) make only one classification error out of 150 samples. Compared to those in Eq.(17), they are simpler in term of the number of variables included, and superior in terms of classification errors.

Fu[8] applied KT algorithm using twofold cross-validation. It generates 5 hyper-rectangle if-part rules with 3.3% error rate. The number of conjunctive terms is 10. It is equivalent to 2 or 3 classification errors for test samples. Kasabov[15] applied REFuNN using 120 samples, which generates a large number of rules using various thresholds, on which extracted rules heavily depend.

8 Conclusions and discussions

In the present paper, a novel approach to rule extraction is proposed using structural learning with successive regularization. It is a succession of learning with a decreasing regularization parameter. Rules are obtained hierarchically, i.e., a combination of dominant rules and less dominant rules or exceptions.

It has various advantages such as the robustness of computation, good understandability, and similarity to human development. It is applied to the classification of mushrooms, the recognition of promoters in DNA sequences and the classification of irises. Empirical results indicates superior performance in rule extraction in terms of the number and the size of rules for explaining data.

The first advantage, the robustness of computation, however, is substantiated by experiments using only mushroom data, hence needs further verification. Concerning the second and third advantages, i.e., good understandability and similarity to human development, we have various observations supporting these, but detailed studies clarifying the relationship between the process of rule extraction and these advantages are left for further study.

References

[1] Akaike, H. (1974). A new look at the statistical model identification, *IEEE Trans. AC*, **19-6**, 716-723.

[2] Andrews, R., Diederich, J. and Tickle, A.B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems*, **8(6)**, 373-389.

[3] Tickle, A.B., Andrews, R., Golea, M. and Diederich, J. (1998). The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks, *IEEE Transactions on NN,* **9(6)**, 1057-1068.

[4] Clark, E.V. (1986). The principle of contrast: a constraint on language acquisition, In B. MacWhinney, (Ed.), *Mechanisms of Language Acquisition,* (pp.Erlbaum.

[5] Duch, W., Adamczak, R., Grabczewski, K., Ishikawa, M. and Ueda, H. (1997). Extraction of crisp logical rules using constrained backpropagation networks — comparison of two new approaches, *Proc. of the European Symposium on Artificial Neural Networks (ESANN'97)*, Bruge, 109-114.

[6] Fahlman, S.E. and Lebiere, C. (1990). The Cascade-correlation learning architecture, *NIPS*, **2**, 524-532.

[7] Fu, L.M.. (1993). Knowledge-based connectionism for revising domain theories, *IEEE Trans. SMC*, **23(1)**, 173-182.

[8] Fu, L.M. (1994). Rule Generation from Neural Networks, *IEEE Trans. SMC*, **24(8)**, 1114-1124.

[9] Ishikawa, M. (1995). Neural networks approach to rule extraction, *ANNES'95*, Dunedin, 6-9.

[10] Ishikawa, M. (1996). Structural learning and knowledge acquisition, *IEEE ICNN'96*, Washington D.C., 100-105.

Table 1: Attributes of mushrooms. *No* stands for an attribute number and *Na* stands for the number of attribute values.

No	*Na*	attributes	*No*	*Na*	attributes
0	6	cap-shape	11	4	stalk-surface-above-ring
1	4	cap-surface	12	4	stalk-surface-below-ring
2	10	cap-color	13	9	stalk-color-above-ring
3	2	burises	14	9	stalk-color-below-ring
4	9	odor	15	2	veil-type
5	4	gill-attachment	16	4	veil-color
6	3	gill-spacing	17	3	ring-number
7	2	gill-size	18	8	ring-type
8	12	gill-color	19	9	spore-print-color
9	2	stalk-shape	20	6	population
10	7	stalk-root	21	7	habitat

Table 2: Major attributes of mushrooms and their attribute values. Acronyms of attribute values are also shown in parentheses.

No	attributes	attribute values
4	odor	almond(a), anise(l), creosote(c), fishy(y), foul(f), musty(m), none(n), pungent(p), spicy(s)
7	gill-size	broad(b), narrow(n)
9	stalk-shape	enlarging(e), tapering(t)
12	stalk-surface-below-ring	ibrous(f), scaly(y), silky(k), smooth(s)
19	spore-print-color	black(k), brown(n), buff(b), chocolate(it h), green(r), orange(o), purple(u), white(w), yellow(y)
20	population	abundant(a), clustered(c), numerous(n), scattered(s), several(v), solitary(y)

Table 3: The Karnaugh map representing the relation between the hidden variable, y_4, representing odor and the corresponding input variables, $\{a,\ l,\ y,\ f,\ n,\ s\}$, in the first stage of the mushroom classification. Acronyms for input variables are given in Table 2. $\times$ represents *don't care.*

n=0, s=0

$y\ f$ / $a\ l$	00	01	11	10
00	$\times$	1	$\times$	1
01	0	$\times$	$\times$	$\times$
11	$\times$	$\times$	$\times$	$\times$
10	0	$\times$	$\times$	$\times$

n=0, s=1

$y\ f$ / $a\ l$	00	01	11	10
00	1	$\times$	$\times$	$\times$
01	$\times$	$\times$	$\times$	$\times$
11	$\times$	$\times$	$\times$	$\times$
10	$\times$	$\times$	$\times$	$\times$

n=1, s=0

$y\ f$ / $a\ l$	00	01	11	10
00	0	$\times$	$\times$	$\times$
01	$\times$	$\times$	$\times$	$\times$
11	$\times$	$\times$	$\times$	$\times$
10	$\times$	$\times$	$\times$	$\times$

n=1, s=1

$y\ f$ / $a\ l$	00	01	11	10
00	$\times$	$\times$	$\times$	$\times$
01	$\times$	$\times$	$\times$	$\times$
11	$\times$	$\times$	$\times$	$\times$
10	$\times$	$\times$	$\times$	$\times$

Table 4: The Karnaugh map representing the relation between the output variable, z, and the hidden variables, $\{y_4,\ y_{19}\}$, in the first stage of the mushroom classification. $\times$ represents *don't care.*

y_{19} / y_4	0	1
0	1	0
1	0	$\times$

Table 5: The Karnaugh map representing the relation between the output variable, z, and the hidden variables, $\{y_4, y_7, y_{12}, y_{19}\}$, in the second stage of the mushroom classification. × represents *don't care.*

$y_{12}\ y_{19}$ / $y_4\ y_9$	00	01	11	10
00	0	×	×	1
01	1	×	0	1
11	0	×	×	0
10	×	×	×	0

Table 6: The Karnaugh map representing the relation between the output variable, z, and the hidden variables, $\{y_4, y_7, y_9, y_{12}, y_{19}, y_{20}\}$, in the third stage of the mushroom classification. × represents *don't care.*

y_4=0, y_7=0

$y_{19}\ y_{20}$ / $y_9\ y_{12}$	00	01	11	10
00	×	×	×	×
01	×	1	×	×
11	0	1	×	×
10	0	0	×	×

y_4=0, y_7=1

$y_{19}\ y_{20}$ / $y_9\ y_{12}$	00	01	11	10
00	×	×	×	×
01	×	1	×	×
11	1	1	0	×
10	×	1	×	×

y_4=1, y_7=0

$y_{19}\ y_{20}$ / $y_9\ y_{12}$	00	01	11	10
00	×	×	×	×
01	×	0	×	×
11	×	0	×	×
10	×	×	×	×

y_4=1, y_7=1

$y_{19}\ y_{20}$ / $y_9\ y_{12}$	00	01	11	10
00	×	×	×	×
01	×	0	×	×
11	×	0	×	×
10	0	×	×	×

Table 7: Summary of rule extraction from a mushroom database. The number of training samples is 812 and that of total samples is 8124, † indicates that the number of training samples is 820 including 8 misclassified samples in test data. MSE_{tr} and MSE_{to} stand for the mean square error for training data and that for the total data, respectively. $\#\text{E}_{tr}$ and $\#\text{E}_{to}$ stand for the number of classification errors for training data and that for total data, respectively. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$, and $\theta = 0.1$. Attributes used in ID3 are: {odor(4), spore-print-color(19)} in case of 2 attributes, {odor(4), spore-print-color(19), cap-color(2)} in case of 3 attributes, and {odor(4), spore-print-color(19), cap-color(2), stalk-root(10)} in case of 4 attributes.

learning method	regularization parameter λ	no. of attri-butes	MSE_{tr}	MSE_{to}	$\#\text{E}_{tr}$	$\#\text{E}_{to}$
BP	–	22	0.00006	0.00606	0	56
SLF	3×10^{-3}	2	0.00611	0.00587	5	48
	2×10^{-3}	4	0.00007	0.00101	0	8
	0.2×10^{-3}†	6	0.00008	0.00008	0	0
ID3	–	2	–	–	5	48
	–	3	–	–	2	24
	–	4	–	–	0	24

Table 8: Resulting attributes by SLF with a constant regularization parameter, $\lambda = 0.2 \times 10^{-3}$, starting from 5 kinds of initial connection weights. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$ and $\theta = 0.1$.

No.	attributes
1	4, 7, 11, 19, 20
2	2, 4, 6, 7, 11, 19, 20, 21
3	4, 7, 10, 11, 19, 20
4	4, 7, 10, 11, 19, 20
5	2, 4, 7, 12, 19, 20

Table 9: The extracted rule for the recognition of promoters in the first stage. Each row corresponds to a conjunctive term and rows are interpreted as disjunctive.

–36	–35	–34
T	T	
T		G
	T	G

Table 10: The extracted rule for the recognition of promoters in the second stage.

–36	–35	–34	–12
T	T		$\neg G$
T		G	$\neg G$
	T	G	$\neg G$

Table 11: The extracted rules for the recognition of promoters in the third stage.

–38	–36	–35	–34	–31	–12
	K	T			$\neg G$
	K		G		$\neg G$
		T	G		$\neg G$
T	K			$\neg C$	

Table 12: The extracted rule for the recognition of promoters in the 4th stage.

–39	–38	–36	–35	–34	–31	–12
		K	T		$\neg C$	$\neg G$
		K		G		$\neg G$
	T	K			$\neg C$	
T			T			

Table 13: Summary of performance of the recognition of promoters in DNA sequences. $\#E_{tr}$ and $\#E_{te}$ stand for the number of classification errors for training data and that for test data, respectively.

learning method	λ	MSE_{tr}	MSE_{te}	$\#E_{tr}$	$\#E_{te}$
BP	0.0	0.000046	0.018660	0	2
SLF	1.8×10^{-2}	0.069080	0.085144	7	1
	10^{-2}	0.043932	0.080684	5	1
	0.6×10^{-2}	0.009040	0.000158	1	0
	0.2×10^{-2}	0.000246	0.000080	0	0

[11] Ishikawa, M. (1996). Rule extraction by successive regularization, *IEEE ICNN*, Washington D.C., 1139-1143.

[12] Ishikawa, M. (1996). Structural learning with forgetting, *Neural Networks*, **9(3)**, 509-521.

[13] Ishikawa, M. and Ueda, H. (1997). Structural learning approach to rule discovery from data with continuous valued inputs, *ICONIP'97*, 898-901, Springer.

[14] Ishikawa, M. (1998). Structural learning and rule discovery from data, In S. Amari and N. Kasabov (Eds.), *Brain-Like Computing and Intelligent Information Systems*, 396-415, Springer.

[15] Kasabov, N.K. (1996). Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems, *Fuzzy Sets and Systems,* **82**, 135-149.

[16] Kowalczyk, A., Ferrá, H. L. and Gardiner, K. (1991). Discovering production rules with higher order neural networks: a case study, *Proceedings of Machine Learning*, 158-162.

[17] Le Cun, Y., Denker, J.S. and Solla, S.A. (1990). Optimal brain damage, In D. S. Touretzky (Ed.) *Advances in Neural Information Processing Systems*, **2**, 598-605, Morgan Kaufmann.

Table 14: Performance of the classification of irises by SLF with various regularization parameters, λ. FI stands for the maximum number of incoming connections to each hidden unit. K is the number of connections and biases of a network. * signifies the minimum AIC. The parameters of learning are: η=0.1, α=0.2 and θ=0.1.

λ	FI	MSE	K	AIC
0.0005	1	0.023300	13	–1665
0.00002	2	0.004206	26	–2410
0.00005	3	0.004606	25	–2371
0.0001	3	0.004080	22	–2432*
0.0002	3	0.008480	20	–2107
0.00005	4	0.004096	30	–2413

Table 15: Summary of performance of the classification of irises. SLFsr stands for SLF with successive regularization. C4.5p means C4.5 with pruning. FI stands for the maximum number of incoming connections to each hidden unit. #p is the number of independent parameters. #r and #a are the number of conjunctive rules and the number of terms in antecedents, respectively. #e stands for the number of classification errors.

learning method	FI	#p	#r	#a	#e	error rate
SLF	1	2	3	6	7	4.7%
	2	7	6	16	1	0.7%
	3	9	4	5	2	1.3%
	4	13	3	7	0	0.0%
SLFsr	2	7	3	6	1	0.7%
C4.5	–	5	–	–	2	1.3%
C4.5p	–	3	–	–	4	2.7%
KT	–	5	5	10	5	3.3%

[18] Jeffrey, J. and Mooney, R.J. (1993). Combining connectionist and symbolic learning to refine certainty-factor rule bases, *Connection Science*, **5**, 339-364.

[19] Murata, N., Yoshizawa, S. and Amari, S. (1994). Network information criterion — determining the number of hidden units for an artificial neural network model, *IEEE Transactions of NN,* **5(6)**, 865-872.

[20] Murphy, P.M. and Aha, D.W. (1992). *UCI Repository of machine learning databases*, Department of Information and Computer Science, University of California, Irvine, CA.

[21] Pinker, S. and Prince, A., (1988). On language and connectionism: analysis of a parallel distributed processing model of language acquisition, In S. Pinker and J. Mehler (Eds.), *Connections and Symbols,* (pp.73-193): The MIT Press.

[22] Pinker, S., (1989). Language acquisition, In M.I. Posner (Ed.), *Foundations of Cognitive Science,* (pp.359-399), The MIT Press.

[23] Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning,* Morgan Kaufmann.

[24] Reed, R. (1993). Pruning algorithms — a survey, *IEEE Trans. Neural Networks*, **4(5)**, 740-747.

[25] Rosch, E. (1978). Principles of categorization, In E. Rosch and B.B. Lloyd(Eds.) *Cognition and categorization*(pp.27-48). Hillsdale, NJ.: Erlbaum.

[26] Sestito, S. and Dillon, T.S. (1994). *Automated Knowledge Acquisition,* Prentice Hall.

[27] Siegler, R. S. (1976). Three aspects of cognitive development, *Cognitive Psychology,* **8**, pp.481-520.

[28] Towell, G.G. and Shavlik, J.W. (1993). Extracting refined rules from knowledge-based neural networks, *Machine Learning*, **13**, 71-101.

[29] Towell, G.G. and Shavlik, J.W. (1994). Knowledge-based artificial neural networks, *Artificial Intelligence,* **70**, 119-165.

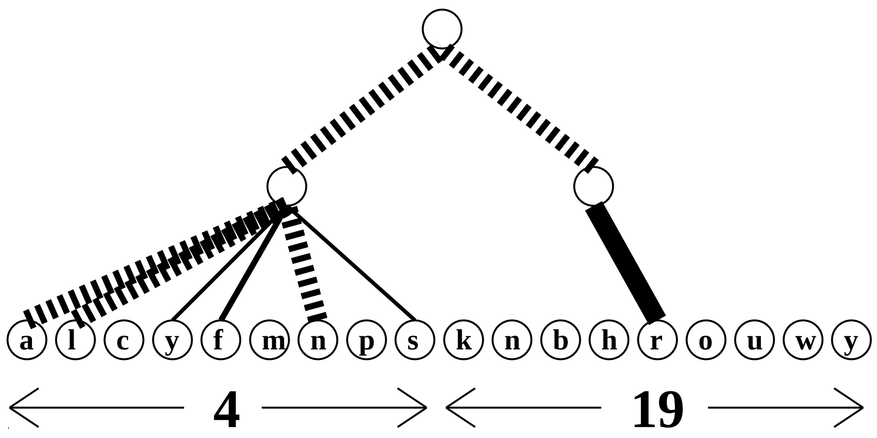

Figure 1: The resulting network for classifying mushrooms by SLF in the first stage. Among 22 attributes only two of them are shown here, because all the connections corresponding to other attributes fade away after learning. The remaining attributes are odor(attribute number 4) and spore-print-color(attribute number 19). An alphabet within an input unit circle is an acronym for the name of an attribute value. Width of each connection is approximately proportional to the absolute value of its weight. A solid connection and a dashed connection stand for positive and negative weights, respectively. The parameters of learning are: the learning rate $\eta = 0.05$, the momentum $\alpha = 0.9$, the amount of forgetting $\varepsilon = 1.5 \times 10^{-4}$, the threshold of selective forgetting $\theta = 0.1$, and the regularization parameter $\lambda = 3 \times 10^{-3}$.

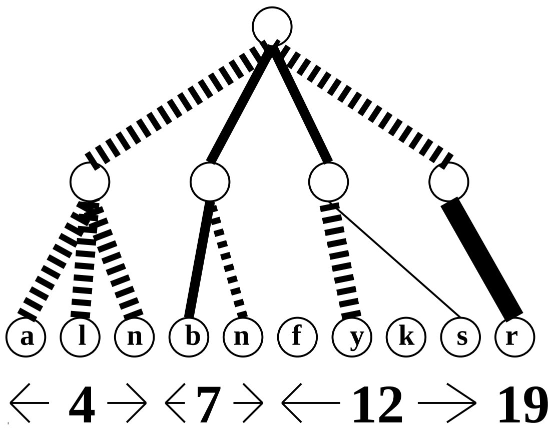

Figure 2: The resulting network for classifying mushrooms by SLF in the second stage. In addition to odor(4) and spore-print-color(19), gill-size(7) and stalk-surface-below-ring(12) are added in the input layer. Among the input units corresponding to attributes odor(4) and spore-print-color(19), only those actively connected to hidden units are shown here for clarity. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$, $\varepsilon = 10^{-4}$, $\theta = 0.1$, and $\lambda = 2 \times 10^{-3}$.

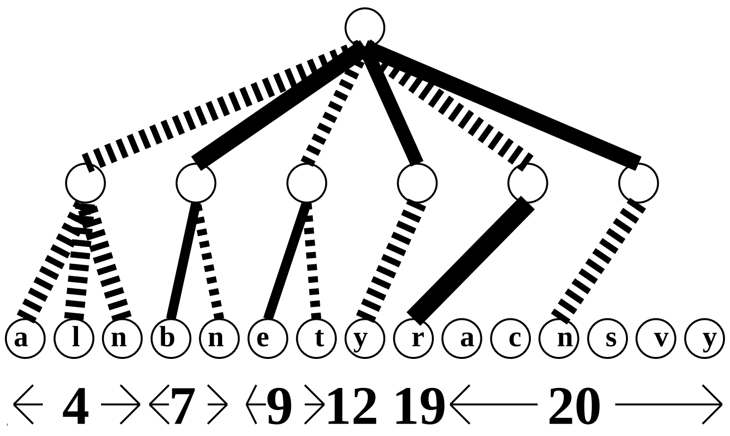

Figure 3: The resulting network for classifying mushrooms by SLF in the third stage. In addition to odor(4), gill-size(7), stalk-surface-below-ring(12) and spore-print-color(19), stalk-shape(9) and population(20) are added in the input layer. Among the input units corresponding to attributes {4, 7, 12 , 19}, only those actively connected to the hidden units are illustrated here for clarity. Otherwise we have to draw 32 input units corresponding to the above 6 attributes. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$, $\varepsilon = 10^{-5}$, $\theta = 0.1$, and $\lambda = 0.2 \times 10^{-3}$.

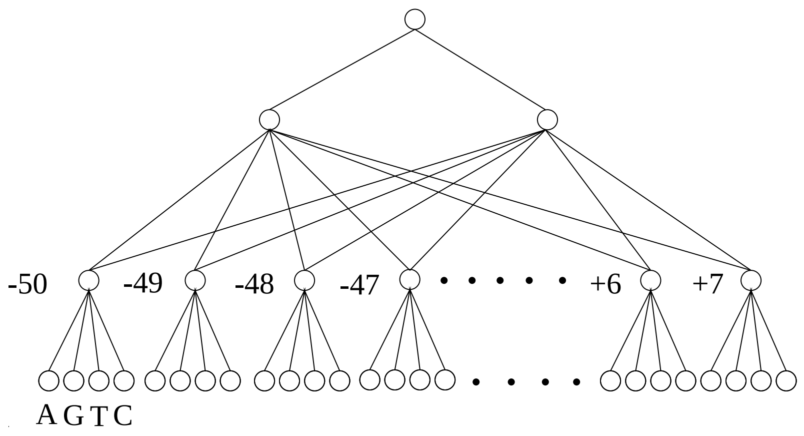

Figure 4: The network architecture for the recognition of promoters in DNA sequences. The input layer has 228(57 $\times$ 4) units and the first hidden layer has 57 units each corresponding to a nucleotide. The second hidden layer has 2 units and the output layer has one unit.

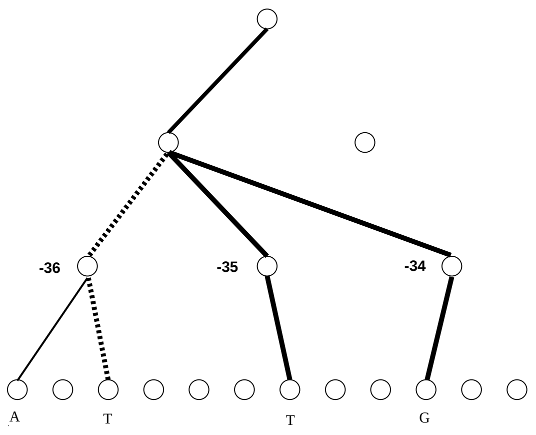

Figure 5: The resulting network for the recognition of promoters in the first stage. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$, $\varepsilon = 9 \times 10^{-4}$, $\theta = 0.1$, and $\lambda = 1.8 \times 10^{-2}$.

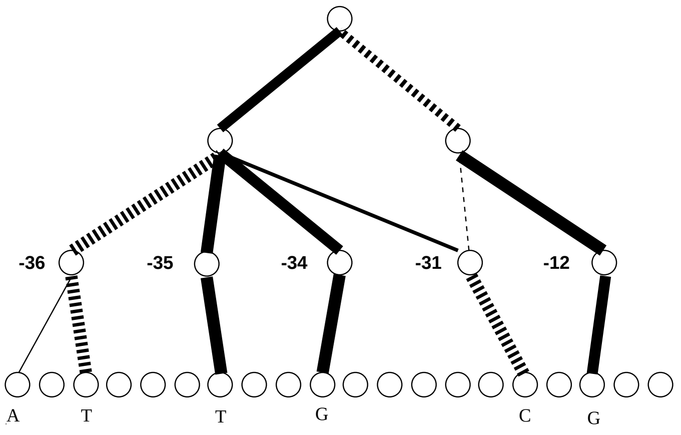

Figure 6: The resulting network for the recognition of promoters in the second stage. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$, $\varepsilon = 5 \times 10^{-4}$, $\theta = 0.1$, and $\lambda = 10^{-2}$.

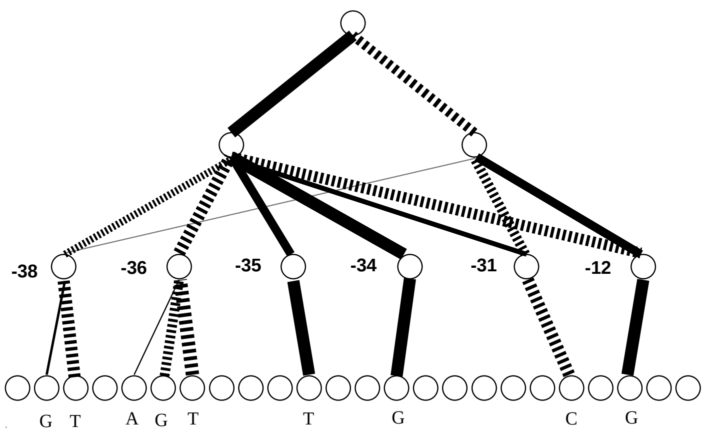

Figure 7: The resulting network for the recognition of promoters in the third stage. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$, $\varepsilon = 3 \times 10^{-4}$, $\theta = 0.1$, and $\lambda = 0.6 \times 10^{-2}$.

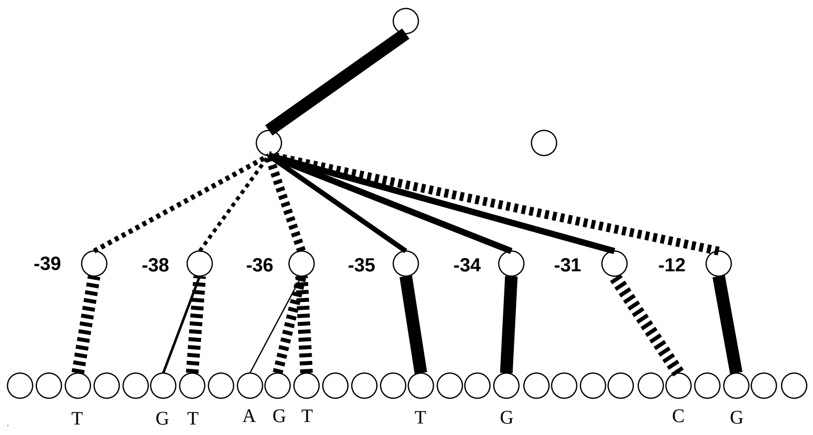

Figure 8: The resulting network for the recognition of promoters in the 4th stage. The parameters of learning are: $\eta = 0.05$, $\alpha = 0.9$, $\lambda = 10^{-4}$, $\theta = 0.1$, and $\lambda = 0.2 \times 10^{-2}$.

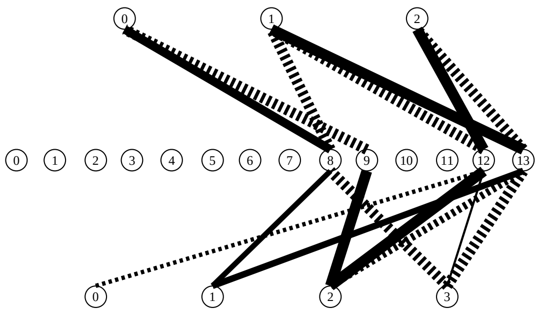

Figure 9: The resulting network for the classification of irises by SLF with at most 3 incoming connections to each hidden unit. The parameters of learning are: $\eta = 0.1$, $\alpha = 0.2$, $\theta = 0.1$, and $\lambda = 10^{-4}$.

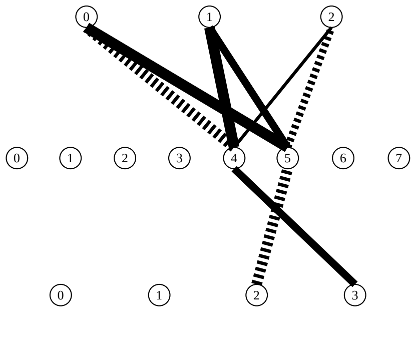

Figure 10: The resulting network for the classification of irises in the first stage by SLF with successive regularization. The parameters of learning are: $\eta = 0.1$, $\alpha = 0.2$, $\theta = 0.1$, and $\lambda = 5 \times 10^{-3}$.

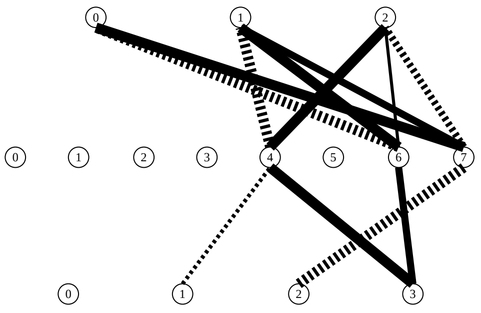

Figure 11: The resulting network for the classification of irises in the second stage by SLF with successive regularization. The parameters of learning are: $\eta = 0.1$, $\alpha = 0.2$, $\theta = 0.1$, and $\lambda = 0.7 \times 10^{-3}$.