Received: 22 September 2016 | Revised: 26 June 2017 | Accepted: 1 July 2017

DOI: 10.1002/cpe.4262

WILEY

SPECIAL ISSUE PAPER

Rule extraction from autoencoder-based connectionist computational models

Juan Yang[1] | Michael S. C. Thomas[2] | Hongtao Liu[3]

[1]College of Computer Science, Sichuan Normal University, Chengdu, Sichuan, China
[2]Developmental Neurocognition Lab, Department of Psychological Sciences, Birkbeck, University of London, London, UK
[3]College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China

Correspondence
Juan Yang, College of Computer Science, Sichuan Normal University, Chengdu, Sichuan, China.
Email: jkxy_yjuan@sicnu.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Number: (61402309); UK Economic and Social Research Council, Grant/Award Number: RES-062-23-2721

Summary

Mapping problems are typical research topics related to natural language learning, and they include not only classification mappings but also nonclassification mappings, such as verbs and their past tenses. Connectionist computational models are one of the most popular approaches for simulating those mapping problems; however, their lack of explanatory ability has prevented them from being further used to understand the language learning process. Therefore, the work of extracting rational rules from a connectionist model is as important as simulating the mapping behaviors. Unfortunately, there is no available technique that can be directly applied in those computational models to simulate nonclassification problems. In this paper, an autoencoder-based connectionist computational model is proposed to derive a rule extraction method that can construct "If-Then" rational relations with high fidelity for nonclassification mapping problems. To demonstrate its generalizability, this computational model is extended to a modified version to address a multi-label classification mapping problem related to cognitive style prediction. Experiments prove this computational model's simulation ability and its explanatory ability on nonclassification problems by comparing its fidelity performances with those of the classical connectionist computational model (multilayer perceptron artificial neural network), and its similar ability on a multi-label classification problem (Felder-Silverman learning style classification) by comparing its prediction accuracy with those of other rule induction techniques.

KEYWORDS

connectionist computational model, classification mapping problem, non-classification mapping problem, rule extraction

1 | INTRODUCTION

Pattern mapping generally refers to a mapping from the feature space into a category space, which defines objects.[1] In research related to natural language learning, pattern mapping is a major research topic that may reveal language developmental natures and include discovering the structural dependencies inherent in the aspects of linguistic units and labeling them with the correct linguistic abstract category.[2] Mapping problem reflects not only the general characteristics of language learning but also the common properties shared by most language developmental processes, eg, the process of summarizing and capturing the general mapping rules of building the past tenses for verbs or generating the plural forms for nouns are shared by most normal children on their language developmental trajectories.

To reveal and understand the causal relationships inherent in those mapping problems, computational models are employed as a basic presentation tool because they can not only simulate humans' mental abilities from experience but also provide plausible explanations for their learning processes [3], such as the mechanistic causal explanations about an intervention method for a language developmental disorder problem.[4] Although computational models can be used in a wide variety of different problem domains related to human cognition, most of them are applied on the language developmental-related problem domains.[5] In the classical generative approach,[6,7] language is characterized in terms of a domain-specific form of knowledge representation called grammar.[8] Generative grammar provides an effective way to understand how language is structured as a production system in a white-box way.[5] Another approach is to use connectionist computational models, such as artificial neural network (ANN) models.[3] In connectionist networks, grammars are taken as the characterizations of some aspects of the behavior,[9,10] and the correlations between input and output are learned based on experience (training data) in a black-box way.

Although connectionist computational models are superior to symbolic ones from a development perspective,[3] to understand and explain the mapping behaviors, one needs to go inside those black boxes (networks) and make them readable.[11] For example, connectionist computational models can be used to simulate the input(verb) and output(past tense) of the verb/past tense mapping problem, and it is necessary to further understand how the past tenses can be successfully mapping from the verbs by the chosen connectionist computational model.

Until now, this cognition-related problem has been described as making a connectionist computational model available to explain the causal relationships between its input and the output. This description can be further extended to a machine learning problem: to discover the hidden knowledge (rules) embedded in a connectionist computational model. If the connectionist computational model simulates a typical mapping problem, such as classifying data into categories based on some similarity of function,[12] there are a variety of rule extraction techniques and methods can be chosen. Unfortunately, there are two unsolved challenges laid before us: first, mapping problems related to language learning include not only classification mappings but also nonclassification mappings (such as building the past tenses for verbs or generating the plural forms for nouns). If we want to extract the rules for a nonclassification problem, we need to design an explanatory component that can reveal the correlations between general input and the nonclassification output of a connectionist computational model. Second, the classical computational model being used to simulate the nonclassification problems is the 3-layered ANN due to its optimal simulating ability on the correlations between input and output.[13] However, there are no available machine learning techniques that can be directly applied to those ANN computational models; even if there were, the performance of the explanation component based on those 3-layered ANNs is inadequate.[4] Therefore, the 3-layered ANN computational model needs to be revised so that the rule extraction derived from it achieves satisfactory explanation ability, while retaining its simulation fidelity to the nonclassification mapping problems.

In this paper, we propose a new connectionist computational model based on autoencoder[14] to simulate the language-learning-related mapping problems, and a rule extraction method with satisfactory explanatory ability for the nonclassification mapping problem is derived from this computational model. For nonclassification mapping problems, the proposed rule extraction approach can effectively reveal the correlations between prior knowledge and new knowledge by rebuilding the inherent rules in an autoencoder-based computational model. Not limited to nonclassification problems, this approach can be further expanded to handle multi-label classification mapping problems (for instance, to construct the correlations between humans' learning behaviors and their cognitive biases from the experience data).

The structure of the rest of this paper is as follows: Section 2 describes the available techniques for rule extraction; Section 3 introduces the paradigmatic descriptions of the linguistic nonclassification mapping problems; the proposed autoencoder-based computational model and the rule extraction approach derived from it are described in Section 4; Section 5 extends this computational model and the rule extraction approach to handle a multi-label classification problem; and Section 6 refers to the related experiments, and the final section concludes this work.

2 | AVAILABLE RULE EXTRACTION METHODS

Some real-life applications, such as medical diagnosis, credit risk evaluation, and commercial decision support systems, require understanding the internal causal relations between observations and the corresponding conclusions, instead of jumping directly to the conclusions.[15] Even if the prediction accuracy of the conclusions was improved, there would be skeptical voices unless the inner rationale relations could be understood.[16] From this perspective, the use of transparent and comprehensible classifiers, such as decision trees[17] and rule-based systems,[15] to extract rules seems to be the best choice if the trade-off between explanatory ability and prediction accuracy is not considered. However, to understand the inner correlations between limited observations and the conclusions, one cannot bypass this question because those two metrics are mutually contradictory.[18]

To improve the explanatory ability while keeping the predicting accuracy as high as possible, black-box classifier-based rule extraction methods are proposed to extract classification rules, such as ANN-based rule extraction approaches[19-22] and SVM-based rule extraction approaches.[23-26] Generally, the basic idea of the black-box classifier-based method is to try to provide an explanatory function based on a black-box classification kernel, and its purpose is to obtain a comprehensible system that approximates the final output performances of those black-box methods. As stated before, the rule extraction approaches that can be applied to discover the hidden knowledge from the ANN-structured computational models are limited to those that carry out classification tasks.[19-22,27-29] In addition, other available rule extraction techniques aim at classification problems,[20,21,30-33] but there are no reports of their application to nonclassification mapping problems.

Understanding the processing of neural networks is a long-standing open problem due to the complicated relations between different layers. One solution is to transform the ANNs into fuzzy systems and produce fuzzy inference rules to simulate the ANNs' mapping behaviors.[19,20,22] Castro et al[19] proposed a fuzzy rule extraction method from the ANNs whose fuzzy propositions are obtained from the weight matrix. This method focuses on building the If-Then relations between the input and the output of an ANN classifier. In Mantas's work,[20] the antecedents of the fuzzy rules consider not only the weight matrix but also the similarity between input data, and the fuzzy rules achieved a more comprehensible description of the ANN's action. In addition to constructing fuzzy rules to simulate the mapping actions between the input and output of a typical ANN-structured classifier, some other solutions are proposed to extract the rules from neural networks by changing their inner architectures, such as changing an ANN into a fuzzy cognitive map,[22] pruning the ANN connections to reduce the number of available assumptions related to the rules,[28] and designing a new error term in the backpropagation learning process to modify the representations of the ANN's hidden layer.[29] In addition to the aforementioned proposals, integrating symbolic techniques to build components that are comprehensible to the ANN classifiers is a possible solution, eg,

Raunal et al[21] added a symbolic regression model to an ANN to imitate the ANN's behavior. Some recent studies have paid more attention to the general rule extraction structures without considering the assumptions of the underlying models.[34]

3 | PARADIGMATIC DESCRIPTIONS OF THE NONCLASSIFICATION LINGUISTIC PROBLEMS

Some linguistic mapping problems are of nonclassification character but can still be represented by "If-Then" rules. Take verbs and past tense, for instance: The mapping behavior is predominantly characterized by a general rule (add -ed to a verb stem to form its past tense). For example, a rule used to generate a past tense of a verb may look like this:

If a verb ends with a voiceless consonant phoneme and this phoneme is not "/d/,"

then add a "/t/" suffix to the end of this verb.

In addition to regular verbs and their past tenses, there exist a minority of exceptions that form their past tenses in different ways:

(1) arbitrary irregular verbs and their past tenses (go-went): there are no rules for converting verbs to their past-tense forms, and the changes are arbitrary;
(2) vowel-change irregular verbs and their past tenses (sing-sang, ring-rang): the vowel phoneme is randomly changed to a different vowel phoneme;
(3) identical irregular verbs and their past tenses (hit-hit): there is no change between verbs and their corresponding past tenses.

Generalization ability can be tested by determining whether the past tense rules extracted from the training data can be applied to novel verbs or can be applied to any of the irregular patterns that are similar to irregular verbs from the training set. Past tense's learning is a commonly used base model for computational models to provide hypotheses about the mechanistic bases of cognition and language[4] because the past tense has been taken to be a paradigmatic linguistic subsystem exhibiting fundamental properties of language.[8] Learning English noun plurals shares many of the same characteristics with learning verb past tenses.[35]

In this paper, past tenses and plurals were used as the representatives of the paradigmatic linguistic problems. The coding mechanism of Plunkett & Marchman,[36] with 19 binary phonological features, and that of Plunkett & Juola, with 16 binary phonological features, [35] are chosen for the verbs/past tenses and the nouns/plurals, respectively. The patterns encoded in Plunkett & Marchmans mechanism are composed of two parts: 3 phoneme verbs and their corresponding past tenses. Each phoneme is encoded in 19 binary bits, and the corresponding meaning of those 19 binary phonological features for each phoneme is illustrated in Figure 1, which correspond to the prior knowledge before learning verbs' past tenses. In fact, there are only 3 basic forms of the 3 phoneme verbs: CVC, CCV, and VCC, where "C" refers to consonant pronunciation and "V" refers to vowel pronunciation. The past-tense suffixes are each encoded in 5 bits for different kinds of past tenses. There are 4 different past tenses: regular, identical irregular, vowel change irregular, and arbitrary irregular. The data sets used in this paper include a training set composed of 508 artificial verbs/past tenses, 410 regular verbs/past tenses, 20 identical irregular verbs/past tenses, 10 arbitrary irregular verbs/past tenses, and 68 vowel change irregular verbs/past tenses, and two testing data sets, respectively, composed of 572 generalized verbs/past tenses and 5000 randomly blended artificial verbs/past tenses.

Plunkett & Marchman's verbs and past tense structures are fixed representations of the language, while Plunkett & Juola's nouns and plural structures are unfixable representations since the lengths of the nouns are unlimited, in the form of "####KAt," where # represents a vacant

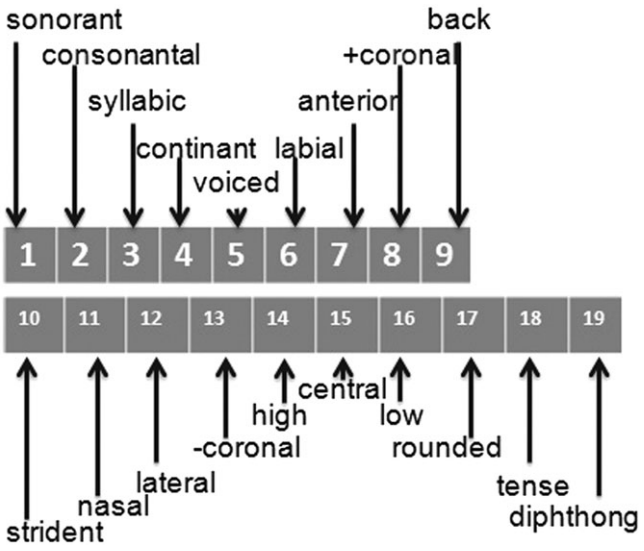

FIGURE 1 Phonological features encoded in 19 binary bits

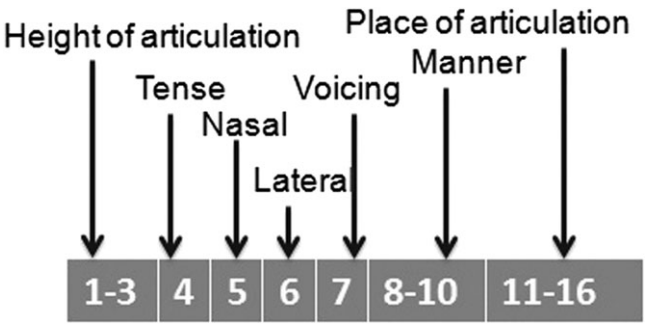

FIGURE 2 Phoneme features encoded in 16 binary bits

phoneme. The nouns in this paper are from Moby's[37] Part-of-Speech data set, and their corresponding pronunciation representations are from Moby's Pronunciator data set. All nouns and their plurals are converted into pronunciation form, and each phoneme is encoded into a 16-bit binary vector; the meanings of the phoneme features are illustrated in Figure 2. There are approximately 51 320 nouns, and their plurals are constructed in a regular way by following the regular plural building rules, such as adding the /iz/ suffix to sibilant-sound (/s,/,/z,/,/S,/,/Z,/,/C/ or /J/)[35] terminated nouns to form their plurals. One thousand randomly chosen items in this data set are used as the training data set, and the rest are treated as testing data.

4 | AN AUTOENCODER-BASED COMPUTATIONAL MODEL TO SIMULATE THE NON-CLASSIFICATION LINGUISTIC MAPPING PROBLEMS

4.1 | Structure of the autoencoder-based computational model

The classical method of simulating the verb/past-tense mapping problem is to construct an ANN with 57 neurons in the input layer and 62 neurons in the output layer. It is difficult to discover the rules embedded in the trained ANN because of the complicated activation states of the output neurons. Our hypothesis is that both verbs and past tenses can be transformed into a simple representation, which can make rule learning easier. The patterns in the training data set can be separated into two parts: premise patterns and conclusion patterns. Premise patterns correspond to the encoded verbs and nouns, and conclusion patterns correspond to their past-tense and plural forms. In this paper, an autoencoder [14] is employed as a black-box component of the computational model because of its high efficiency in reducing the dimensionality and its unsupervised learning ability. The structure of this autoencoder-based computational modal is illustrated in Figure 3. In this computational model, two different autoencoders are involved: the first is a k-$h1$-$h2$-k' autoencoder, which is used to learn the conclusion patterns' features, and the second is a t-$h1$-$h2$-t'-structured autoencoder, which is used to learn the premise patterns' features. In the verb/past-tense problem, $k = 62, k' = 8, h1 = 500, h2 = 200$, and $t = 57, t' = 4$, and in the nouns/plurals problem, $k = 450, h1 = 1000, h2 = 500, k' = 32, t = 418$, and $t' = 16$. Rule building is implemented on the extracted conclusion features and their corresponding premise features, as in Fig4.

4.2 | Training the computational model

As stated before, two autoencoders are constructed to separately learn the features of the conclusion patterns and the features of their corresponding premise patterns. The pretraining process needs to be executed $|class| + 1$ times, where $|class|$ is the number of the classes into which conclusion patterns are being categorized after they are learned in an unsupervised manner in the autoencoder. The k-$h1$-$h2$-k'-structured autoencoder executes once for learning the conclusion patterns, and the t-$h1$-$h2$-t'-structured autoencoder executes $|class|$ iterations for learning premise patterns in different classes. Both autoencoders use restricted Boltzmann machines[14] to pretrain the patterns for a new neuron layer of the autoencoder. After an autoencoder has been successfully trained, the network is stable and supported by the sample patterns. The output of the final hidden encoding layer represents the features for which we are looking.

4.3 | Extracting rules from the trained computational model

These language learning problems can be generally described as follows:

Already known: Necessary prior knowledge, namely, premise patterns and their corresponding conclusion patterns in a training data set.

Hidden knowledge that needs to be discovered: Specific rules can generate corresponding conclusion patterns for the premise patterns.

Result: Given new premise patterns, the system can generate corresponding conclusion patterns by using the rules (hidden knowledge).

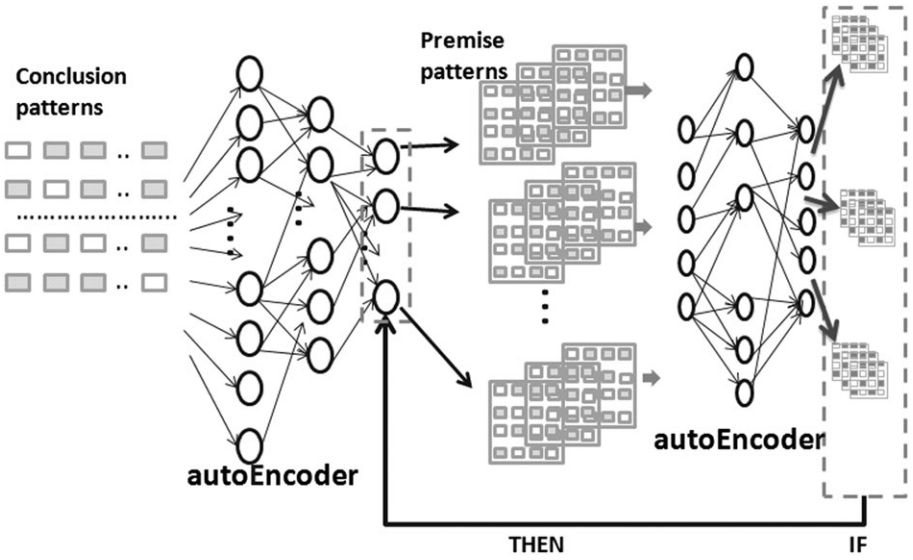

FIGURE 3 An autoencoder-based computational model to simulate the nonclassification linguistic mapping problems

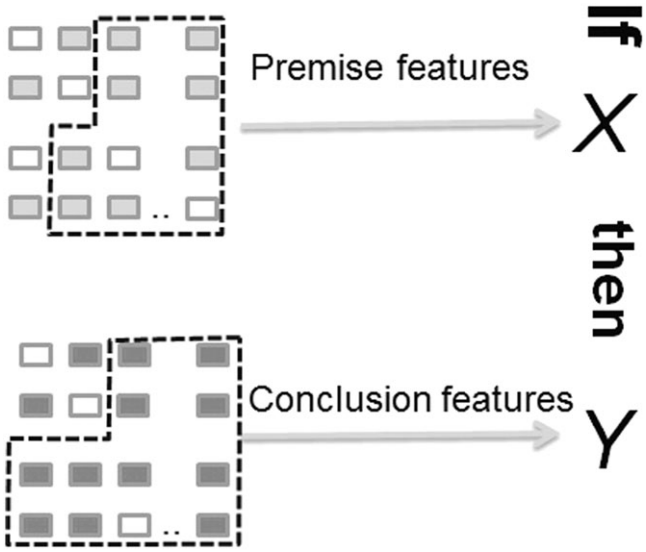

FIGURE 4 Feature representations of the sample patterns

To implement knowledge discovery on this kind of nonclassification learning problem, the process of rule extraction can be implemented using Algorithm 1.

Algorithm 1 Hidden knowledge finding Algorithm

1: Construct a *k-h1-h2-k*'-structured autoencoder, and set the number of neurons *k* of the visible layer equal to the number of binary bits of a conclusion pattern.
2: Pretrain the autoencoder with the conclusion patterns in the training data set, and obtain the weight matrix $weight_{conclusion}$.
3: Classify conclusion patterns and their corresponding premise patterns into $class_i$ according to the conclusion patterns' neuron-activation states in the final hidden layer. Conclusion patterns are stored in *con_class*{*i*}, and their corresponding premise patterns are stored in *pre_class*{*i*}.
4: Construct the second autoencoder in a *t-h1-h2-t*' structure, and set the number of neurons *t* of the visible layer equal to the number of binary bits of a premise pattern.
5: Pretrain the autoencoder separately with the patterns in each premise subclass *pre_class*{*i*}, *classifypre_class*{*i*} patterns into different *pre_class*{*i,j*} subclasses according to the neurons' activation states in the final hidden layer of the encoder, and separately obtain the weight matrices $weight_{premise}\{i,j\}$ for each class *i*.
6: Use function *Recog*(.) to identify the key feature space $filter_{conclusion}\{i\}$ of the conclusion patterns for each $class_i$.
7: Use function *Recog*(.) to identify the key feature space $filter_{premise}\{i,j\}$ of the premise patterns within the same premise subclass *pre_class*{i,j}.
8: Build "If-Then" rules of the form "If a pattern's feature space $filter_{premise}\{i,j\}$ satisfies feature state $Val_{Kp}\{i,j\}$; then, this pattern can be transformed into a new form by changing the value of its conclusion feature space $filter_{conclusion}\{i\}$ to $Val_{Kc}\{i\}$ while the values of other features are unchanged."
9: Prune rules using *Prune*(.)

There are two parameters that need to be further explained: feature space *filter*{} and feature state *Val*{}. Both parameters appear in each rules proposition part and its conclusion part. The feature space *filter*{} is the set of the names of the attributes being used to encode a pattern, and the feature state *Val*{} is their activation states (0 or 1). Combined together, they work like a fuzzy set in an ANN-based fuzzy rule inference system. For example, suppose that the feature space is [x1×4 x6×8], which refers to first, 4th, 6th, and 8th bits of a pattern; its feature state would be [1 0 0 0].

4.4 | Clustering patterns without classification labels

Unlike classification problems, most verbs' past tenses and nouns' plurals share the same feature space with their original forms except for some modifications to the specific features, which are indicated in the construction rules. Modifications in the conclusion part are usually regular and can thus be used as symbols to cluster the patterns.

The basic idea of using an autoencoder is to try to understand whether there are differences between the patterns by using its key-level feature extraction ability and its clustering ability. After successfully learning the conclusion patterns in a semisupervised manner, the weight matrix is stored in $weight_{conclusion}$. Then, the conclusion patterns and their corresponding premise patterns are classified into $class_i$ according to the conclusion patterns' neuron activation states in the final hidden layer, which is also the feature representation layer of the encoder. Conclusion patterns are stored in *con_class*{*i*}, and their corresponding premise patterns are stored in *pre_class*{*i*}. Then, the premises patterns in different classes will be learned in a semisupervised manner with another autoencoder in a similar process.

4.5 | Finding feature spaces for premises and conclusions

At this point, the high-level features of conclusions and their corresponding premises have been obtained; however, we do not know what they represent. To understand what those features stand for, we need to interpret them. Algorithm 2 is designed to reinterpret the meanings of the features on the highest abstract feature layer. The basic idea of this rule extraction model is borrowed from image recognition, and the activation states in the highest hidden layer always represent the most representative features of the original input data. In this section, those features can be reinterpreted in the form of prior knowledge because of their structured presentations.

The pseudocode of the feature-space identification process *Recog*(.) is illustrated in Algorithm 2. To find the optimal mapping combinations of the key features, we use a modified genetic algorithm to solve the optimal problem. The key feature spaces' recognition process will be repeatedly implemented on both conclusion patterns and their premises patterns.

Algorithm 2 Recog( )

1: compute Net_j

$$Net_j = \begin{cases} sigm(TrainData, w_{12}), & j = 1, \\ sigm(Net_{j-1}, w_{j-1j}), & j > 1. \end{cases} \quad (1)$$

2: minimize

$$\| TrainData \times repmat(X, n, 1) - Net_{t'} \|^2; \quad (2)$$

3: **return** X

In Algorithm 2, Net_j in Equation 1 is the sigmoid output of layer j, and $Net_{t'}$ represents the output of the last layer of the autoencoder; it is a recursive function. n in Equation 2 is the size of the class *TrainData*, repmat is a function to replicate n rows of matrix X, and operator × stands for the Hadamard product. The modified GA is used to solve the minimization problem. In this GA, the goal function is defined as the "evaluating function," and two modifications are made to the classical GA process:

(1) Symmetrically exchanging the genes being selected out. If we exchange a gene code 0 of parent A with the gene code 1 of parent B, we have to exchange another gene pair with the opposite values, and vice versa. For example, suppose that parent A has the gene code < 000111 >, and parent B has the gene code < 111000 >. If we decide to exchange the third gene codes of A and B, A will change from < 000111 > to < 001111 >, and B will change from < 111000 > to < 110000 >. According to the symmetric exchange policy, another gene pair with the opposite values should be exchanged at the same time. In this case, if we choose to exchange the sixth genes, A will finally change from < 001111 > to < 001110 >, and B will change from < 110000 > to < 110001 >. After completing the exchange process, both A and B should have the same number of 1s as their original codes.

(2) Symmetrically mutating the selected gene. When mutating a parent with probability α, if we choose to mutate one of that parent's gene codes from 0 to 1, another of that parent's gene codes should be mutated from 1 to 0, and vice versa. For example, suppose that parent A has the gene code < 000111 >, and A has been chosen for mutation with probability α. If we choose to mutate the first 0, which means that A will change from < 000111 > to < 100111 >, another 1 should be chosen for mutation at the same time to maintain the fixed number of 1s. In this case, we choose to mutate the last 1 to 0. Finally, A will change from < 000111 > to < 100110 >.

4.6 | Pruning the rules

The rule generation strategy proposed in this paper is a black-box-based technique and the problem domain is nonclassification. Therefore, we can borrow the modified fidelity metric (instead of calculating the correctly classified testing patterns, we calculate the RMSE values for each testing pattern) to evaluate the effectiveness of the rules and then prune them. As mentioned by Fortuny and Martens,[34] "*if the fidelity metric is high enough, one can decide that enough insight into the black-box model is obtained*," which implies that if the accuracies of the rules are approximately equal to the RMSE values produced by the best-performing black-box technique, the rules are meaningful and practical. The pseudocode for pruning the rules according to their accuracy and covering ratio is given in Algorithm 3.

Algorithm 3 Prune( )

for all $(filter_{premise} = Val_{Kp}) \rightarrow (filter_{conclusion} = Val_{Kc})$ rules **do**
  for all patterns $p \in Pretest$ **do**
    if p satisfies the premise $(filter_{premise}\{i,j\} = Val_{Kp}\{i,j\})$ **then**
      Construct p's conclusion form p^* by setting p's specific feature bits to $(filter_{conclusion}\{i\} = Val_{Kc}\{i\})$ while keeping the remaining bits unchanged
      Select p's conclusion pattern p^c from the conclusion testing data set
      $err_p = \frac{1}{N}\sqrt{\sum_{t=1}^{N}(p_t^* - p_t^c)^2}$
      $P_i^j \leftarrow p$
    end if
  end for
  $RMSE\{i,j\} = \frac{1}{|P_i^j|} \cdot (\sum_{p \in P_i^j} err_p)$
  $CoverRatio\{i,j\} = \frac{|P_i^j|}{|ConTest|}$
end for
Ordering rules in descending order of their *CoverRatio*s
Eliminating rules whose $RMSE \geq \alpha$;
for all rules within the same i **do**
  Merging rules' premises with 'or' operator;
end for

This algorithm is used to reduce the number of available rules, which are in the form of

IF $(filter_{premise}, Val_{Kp})$ satisfied,

THEN $(filter_{conclusion}, Val_{Kc})$,

based on their prediction accuracy (α) and covering ratio (δ), where α is the MRSE performance of the multilayer perceptron–based ANN. MSE is computed between a generated conclusion pattern, which is predicted by a specific inference rule of a premise pattern, and its real conclusion pattern in the testing data set. If a rule's MRSE performance is above the threshold, it will be deleted; otherwise, it will be kept and merged with others into a single new rule.

5 | EXTENDING THE COMPUTATIONAL MODEL TO ADDRESS MULTI-LABEL CLASSIFICATION PROBLEMS

The process described in the previous section is a standard rule extraction procedure for nonclassification problems. In this section, the generalization of this hidden knowledge discovery method can also be extended to address classification mapping problems, especially those for which classification can be performed from different perspectives (multi-label classification). For example, classifying a learner's cognitive style based on his online learning behaviors may be more complicated than other classification problems. The classification labels in the training data set usually come from questionnaires, of which there can be more than 4 types, while the learning behaviors are the structured records collected from the online learning processes. In those multi-label classification problems, even the same learning behavior records would result in different learning style (LS) labels because of the different content that learners are learning. In this section, we take the Felder-Silverman LS[38] as an example, and the patterns can be classified into 8 polarized labels from 4 different perspectives: active-reflective, sensitive-intuitive, verbal-visual, and sequential-global.

Learning behaviors are collected from one of our learning systems, named PIJ,[39] and the content of the lesson is "Syntax of Java," which can reflect programming characteristics of students majoring in computer science. One hundred computer science–majored learners from Sichuan Normal University, Chengdu University, and Chongqing University of Posts and Telecommunications participated. Before they began the learning process, they were required to carefully complete the Index of Learning Style instrument for Felder& Silverman LS evaluation. Learning behaviors are versatile and include the use of tools for interpersonal communication, practice quizzes/exercises/questions, users' learning sequences, and extra hyperlinks visited by users. Then, the hidden knowledge discovery process is modified as shown in Algorithm 4.

Algorithm 4 Rule extraction algorithm for LS problems

1: Construct a *49-100-30-k*-structured autoencoder
2: Pretrain the autoencoder with the training data
3: Categorize the training data into class *Class_i* according to the neuron-activation states in the *k*th layer
4: Use function *Recog*(.) to recognize the key feature space *X_i* of the sample patterns within the same class Class_*i*
5: Compute the effective distribution *accuracy* of the feature behavior set (*X_i*, *Class_i*) on each LS bias state e_j using function *Effec*(.)
6: Build "If-Then" rules of the form "If a learner's learning behavior satisfies (*X_i*, *Class_i*), then he/she is predicted as e_j biased with probability *accuracy(i)*$\{e_j\}$."

The recognition process *Recog*(.) mentioned in Algorithm 4 is the same as that used in Algorithm 1. The rule extraction process being described here is similar to the one being used to address the linguistic nonclassification mapping problems; the only difference is that there are no conclusion patterns and their feature spaces are replaced by the different classification labels.

5.1 | Computing the effective distribution of the features

Different from nonclassification problems, multi-label cognitive models need to compute the effective distribution on labels with different dimensions and compute the prediction accuracy on classification labels. In this section, we construct a "cause and effect" relationship between key features and the LS preference labels with a probability. Function *Effec*(.) is used to compute the effective probability distribution *accuracy* of the key features on different LS preference states. Given an effective probability distribution, the rule "If a learner satisfies (*X*_i,*Class_i*), then he/she is predicted to be an e_j biased learner with probability *accuracy(i)*$\{e_j\}$" can be generated. The distribution *accuracy* is a factorized mean 2-norm value between e_j and the learners' labels. The pseudocode of the function *Effec*(.) is given in Algorithm 5:

Algorithm 5 *Effec*()

1: **for** $1 \le j \le |E|$ **do**
2: **for** $1 \le i \le |Class|$ **do**
3: **if** (*X_i*,*Class_i*) is satisfied a pattern **then**
4: $pre = e_j$
5: $accuracy(i)\{e_j\} = 1 - \frac{1}{N}\sum(\frac{\|e_i - target\|^2}{\sum(\|e_i - target\|^2)})$
6: **end if**
7: **end for**
8: **end for**
9: **return** *accuracy*

This is a procedure for computing the effective distribution of the feature space *Class_i* and its corresponding explanation *X_i* on each LS bias state e_i, where $e_i \in E$. *Class* is a set containing the classes revealed by the autoencoder after an unsupervised learning process, and *E* is a set containing all possible combinations of the LS bias values within an LS model. Suppose that there are *m* dimensions of an LS model and *n* types of values on each dimension, $|E| = \sum_{0 \le i \le m} C_m^i (C_n^1)^i$. $e_i \in E$ is a kind of LS bias state. Take Felder & Silvermans LS model as an example. There are 4 (*m*) dimensions, and on each dimension, a learner can be evaluated as one kind out of two (*n*), eg, visual or verbal; in this case, |*E*|=80, and e_i can be "0," "01," or "1111." *Target* is a set of labels collected from the LS instruments.

5.2 | Merging the rules

Different features have different effects on a specific LS bias state e_j. We can merge those features into a specific LS bias prediction rule by specifying an accuracy threshold parameter δ, and the pruning is controlled by the threshold. The pseudocode of the function *Merge*() describes a procedure of generating a single rule for predicting the same LS bias state e_j if and only if the feature's prediction accuracy on e_j is greater than the given threshold δ; the details are given in Algorithm 6.

Algorithm 6 *Merge*()

1: **for all** e_j **do**
2: **if** *accuracy*(t)$\{e_j\} \ge \delta$ and *accuracy*(k)$\{e_j\} \ge \delta$ and $t \ne k$ **then**
3: premier=((X_t,Class_t) or (X_k,Class_k))
4: **end if**
5: **end for**

6 | EXPERIMENTS

To evaluate the efficiency of the rules discovered for nonclassification cognitive models, we examined the fidelity (RMSE) of the rules of generating past tenses and plurals. The experiment is based on three mixed data sets: 572 generalized verbs/past tenses, 5000 artificially generated verbs/past tenses, and 50 320 nouns and their plurals from Moby's Pronunciator. Four rules are found for the verbs/past tense data sets and 14 rules are found for the nouns/plurals data set. The performance of the hidden knowledge (rules) is compared with that of the multilayer perceptron based 3-layered ANNs because this has been proven to be the most effective technique for learning verbs and their past tenses.[13]

Conversely, to prove the general advantage of the proposed method on multi-label classification cognitive models, we use 100 students' online learning records on "Syntax of Java" along with their multidimensioned LS classification labels as the data set.

6.1 | Hidden knowledge embedded in verbs and their past tenses

In this part, 4 rules of the following form are learned from a training data set composed of 508 blended patterns:

- $Rule_1$:
  $If(x_{15} = 0$ and $x_{24} = 1$ and $x_{40} = 1)$
  $then(y_2 = 1$ and $y_3 = 0$ and $y_7 = 1$ and $y_{20} = 1$ and $y_{22} = 1$ and $y_{23} = 1$ and $y_{24} = 1$ and $y_{40} = 1)$
- $Rule_2$: $If(x_4 = 1$ and $x_{15} = 1$ and $x_{40} = 1$ and $x_{43} = 0)$
  or $(x_4 = 1$ and $x_{15} = 0$ and $x_{40} = 1$ and $x_43 = 1)$
  or $(x_4 = 1$ and $x_{15} = 0$ and $x_{40} = 0$ and $x_{43} = 1)$
  or $(x_4 = 0$ and $x_{15} = 1$ and $x_{40} = 0$ and $x_{43} = 1)$
  or $(x_4 = 0$ and $x_{15} = 1$ and $x_{40} = 1$ and $x_{43} = 1)$
  $then(y_4 = 1$ and $y_5 = 1$ and $y_{21} = 1$ and $y_{39} = 1$ and
  $y_{42} = 1$ and $y_{43} = 1$ and $y_{58} = 1$ and $y_{59} = 1)$
- $Rule_3$:
  $If(x_4 = 1$ and $x_{21} = 1$ and $x_{40} = 1$ and $x_{56} = 0)$
  it then $(y_1 = 1$ and $y_5 = 1$ and $y_6 = 0$ and $y_{21} = 1$ and
  $y_{40} = 1$ and $y_{43} = 1$ and $y_{58} = 1$ and $y_{60} = 1)$
- $Rule_4$:
  $If(x_4 = 1$ and $x_5 = 1$ and $x_{17} = 1$ and $x_{36} = 0)$
  it then $(y_1 = 1$ and $y_5 = 1$ and $y_7 = 0$ and $y_9 = 1$ and
  $y_{18} = 1$ and $y_{26} = 1$ and $y_{28} = 1$ and $y_{60} = 1)$

The performance results of the two testing data sets are listed in Table 1. Two indices are used to evaluate their efficiency: RMSE and covering ratio. The two testing data sets are 5000 blended artificial verbs and their past tenses (A) and 572 generalized verbs and their past tenses (B).

The rules discovered by this computational model for verbs and their past tenses work effectively in terms of both accuracy and covering ratio. Index "Cross covering ratio' refers to the rules" effect area among data sets, except for arbitrary irregular verbs, almost regular verbs, identical irregular verbs, and vowel change irregular verbs, which are covered by the premise feature spaces. Since data set A shares the most characteristics with the training data set, the fidelity performance on test data set A is approximately the same as the performance on training data set. In contrast, data set B is composed of verbs that were randomly selected from the database; therefore, the result on B is much better than that on A.

TABLE 1 Performance of the discovered hidden knowledge in verbs/past tenses

Rules	B(RMSE/ Covering Ratio)	A(RMSE/ Covering Ratio)
$Rule_1$	2.1747/0.7622	2.0244/0.4679
$Rule_2$	2.3547/0.2342	2.4926/0.3596
$Rule_3$	2/0.1905	2.0976/0.3019
$Rule_4$	2.0510/0.0506	2.0912/0.1196
Cross covering ratio	0.9775	0.9746
RMSE of the MLP-based ANN	1.5678	2.3756

Abbreviations: ANN, artificial neural network; MLP, multilayer perceptron.

TABLE 2 Performance of the discovered hidden knowledge in nouns/plurals

Rules	RMSE	Covering ratio	Rules	RMSE	Covering ratio
$Rule_1$	9.7610	0.4724	$Rule_8$	9.9438	0.1834
$Rule_2$	8.3694	0.0998	$Rule_9$	8.4921	0.1667
$Rule_3$	9.3921	0.096	$Rule_{10}$	9.5886	0.1243
$Rule_4$	9.4430	0.1244	$Rule_{11}$	9.3706	0.2861
$Rule_5$	9.7105	0.4449	$Rule_{12}$	9.9025	0.2777
$Rule_6$	9.0260	0.1912	$Rule_{13}$	7.9994	0.0911
$Rule_7$	9.9992	0.1992	$Rule_{14}$	9.9124	0.1372
Cross covering ratio	0.7618				
RMSE of the MLP based ANN	9.985				

Abbreviations: ANN, artificial neural network; MLP, multilayer perceptron.

6.2 | Hidden knowledge embedded in nouns and their plurals

For the problem with nouns and plurals, only rules with the highest covering ratio (more than 0.09) and with the highest accuracy (remove the rules whose $RMSE \geq 10$) based on the training data set will be selected out to produce the plurals of the nouns. Fourteen rules emerged, and their performances are listed in Table 2. Although the rules' fidelity is also approaching ANN's performance, the RMSE results are much higher compared with the problem with verbs and past tenses. That is because the nouns are unfixable while the verbs are fixed. This result exposes the deficit of general processing abilities of the ANN-based computational models, especially their ability to address heterogeneous data.

6.3 | Hidden knowledge embedded in the Felder-Silverman LS cognitive model

For the Felder-Silverman LS cognitive model, the autoencoder is structured as 49-100-30-*t*', where 49 is the number of input units and 100 and 30, respectively, correspond to the numbers of units in the first and second hidden layers. To verify the number of feature units, we separately set the value of *t* as 2, 3, 4, and 5, which correspond to 4, 8, 16, and 25 classes, respectively. The possible values of *t* imply that there may exist 4, 8, 16, or 25 kinds of different classes. By separately setting *t* to 2,3,4, and 5, and computing the reconstruction MSEs, *t* is finally set to 2, which has a smallest MSE reconstruction error of 4.15.

Four (*X_i*,*Class_i*) pairs are found. The probability threshold δ is set to 0.6, and the effective distribution above δ of each rule is treated as the probability of predicting LS bias state e_j, which is listed in Table 3. Actually, if we set δ to 0.8, the 4 features can generate one highly effective rule. We predict a learner to be a visually biased learner if his learning behavior feature satisfies the following premise:

($x_{38} = 0$ and $x_{48} = 0$) or ($x_{17} = 1$ and $x_{30} = 1$) or ($x_{28} = 1$ and $x_{33} = 0$) or ($x_{14} = 0$ and $x_{30} = 1$).

This rule can be expressed in a more readable form by mapping those features into their meaningful representations:

Under topic "Syntax of Java,"

If ("a learner's maximized staying time on a uni-learning object is not in the range [3 minutes, 5 minutes]" and "do synthesized hands-on experiments")

or ("use the search engine with a high frequency in a late time zone of the current learning period" and "use the searching engine with a high frequency in an early time zone of the current learning period")

or ("use the linkage with the prior knowledge" and "not use mail tools when solving problems")

or ("not use flow charts in the current topic" and "use the search engine with a high frequency in an early time zone of the current learning period")

Then this learner has probability 0.875 of being a visually biased learner.

The prediction accuracy of this merged rule is 0.875, and it approximates the mean prediction result of the network on the visual/verbal dimension after supervised learning, which is listed in Table 4. In Table 4, column "A/R" stands for the "Active/Reflective" dimension, "S/G" stands for the "Sequential/reflective" dimension, "S/I" stands for the "Sensitive/Intuitive" dimension, and "V/V" stands for "Visual/Verbal" dimension. Table 5 gives the performance comparisons between our rule extraction approach and other rule induction techniques with comparable explanation ability, such as DecisionTable,[40] M5Rules,[41] SQRex-SVM,[21] and ANN rule extraction.[21] To guarantee the efficiency of the rules, the accuracy threshold is set to 0.65. Two final rules are generated according to function *Generate*():

(1) ***If*** ($x_{17} = 1$ and $x_{30} = 1$) or ($x_{28} = 1$ and $x_{33} = 0$) or ($x_{14} = 0$ and $x_{30} = 1$)
Then this learner has probability **0.6667** of being a **sensitive** -biased learner.

(2) ***If*** ($x_{38} = 0$ and $x_{48} = 0$) or ($x_{17} = 1$ and $x_{30} = 1$) or ($x_{28} = 1$ and $x_{33} = 0$) or ($x_{14} = 0$ and $x_{30} = 1$)

TABLE 3 Effective distribution of the features on e_i

Feature1: $x_{38} = 0$ and $x_{48} = 0$	
e_j	$accuracy(e_j)$
Sequential	0.62
Visual	0.90
Feature2: $x_{17} = 1$ and $x_{30} = 1$	
e_j	$accuracy(e_j)$
Global	0.68
Sensitive	0.64
Visual	0.82
Intuitive & Visual	0.61
Feature3: $x_{28} = 1$ and $x_{33} = 0$	
e_j	$accuracy(e_j)$
Global	0.61
Sensitive	0.69
Visual	0.89
Sensitive & Visual	0.61
Feature4: $x_{14} = 0$ and $x_{30} = 1$	
e_j	$accuracy(e_j)$
Sensitive	0.67
Visual	0.89

TABLE 4 Correctly classified accuracy of the network after a supervised learning process

Correct Classified	4 LS dimensions			
	A/R	S/G	S/I	V/V
Train data	0.55	0.6833	0.7167	0.8667
Test data	0.36	0.4722	0.6667	0.8889

Abbreviations: A/R, Active/Reflective; LS, learning style; S/G, Sequential/reflective; S/I, Sensitive/Intuitive; V/V, Visual/Verbal.

TABLE 5 Comparing the performance with those of other rule induction techniques

		A/R		S/G		S/I		V/V	
Classifier	Data Sets	Correctly classified	RMSE	Correctly classified	RMSE	Correctly classified	RMSE	Correctly classified	RMSE
DecisionTable	Train	0.9214	0.1982	0.8259	0.2951	0.7921	0.3224	0.8549	0.2693
	Test	0.5377	0.5321	0.539	0.5674	0.5613	0.5867	0.7619	0.3675
M5Rules	Train	0.6012	0.446	0.6564	0.4023	0.7376	0.3297	0.7944	0.3206
	Test	0.4493	0.6002	0.5773	0.5195	0.6075	0.4962	0.7345	0.3085
SQRex-SVM	Train	0.9327	0.2158	0.8672	0.3015	0.7861	0.3177	0.9153	0.1577
	Test	0.5562	0.6011	0.6013	0.4271	0.6055	0.5571	0.8127	0.2987
ANN Rule-Extraction	Train	0.8652	0.2257	0.7651	0.3348	0.7621	0.3285	0.8779	0.1923
	Test	0.3765	0.7153	0.5867	0.6273	0.5845	0.6113	0.7953	0.2558
Our method	Train	–	–	–	–	(Sensitive) 0.7273	0.5222	(Visual) 0.8868	0.365
	Test	–	–	–	–	(Sensitive) 0.6471	0.5941	(Visual) 0.9032	0.3111

Abbreviations: A/R, Active/Reflective; S/G, Sequential/reflective; S/I, Sensitive/Intuitive; V/V, Visual/Verbal.

Then this learner has probability **0.875** of being a **visual** -biased learner.

Two indices are used to evaluate the performance of each technique: the first is the number "Correctly classified," and the other is "RMSE (root mean squared error)." The rules generated based on the current topic have a significant effect on predicting a learner's visual and sensitive biases; as a consequence, the learners' LS biases on the A/R and S/G dimensions cannot be predicted. However, it is obvious that other rule induction techniques cannot work effectively on those dimensions either. This result is consistent with our prior assumptions about LS preference prediction;

that is, learners' LS preferences would only be revealed if the learning contents have no influence factors on such dimensions, and our method can effectively find the hidden knowledge embedded in the premises that is vital to a specific classification label.

In addition, our method's advantage is revealed by its high prediction accuracy on the testing data set, especially compared with SQRex-SVM. Although the prediction accuracy of V/V LSs on the training data set is not as good as that of SQRex-SVM, its performance on the testing data set is superior to that of SQRex-SVM. Moreover, our autoencoder-based rule extraction approach is superior to all other techniques listed in Table 5 on the testing data set.

7 | CONCLUSIONS

In this paper, we proposed an autoencoder-based computational model that can simulate the nonclassification mapping problems in the language developmental domain and a derived rule extraction approach that can reveal the rational relations between the premise patterns and the conclusion patterns of this connectionist computational model. To demonstrate its generalizability, this computational model and the derived rule extraction method are extended to address a multi-label classification problem. Based on this computational model, the cognition-related mapping problems are transformed into machine learning problems, and the experimental result shows that this computational model not only has the ability to approximate the simulation performance of a classical ANN but also has satisfactory explanatory ability on the causal relations between the input and the output by constructing "If-Then" rules. The experimental results prove this conclusion by comparing this computational model's fidelity performance with that of the classical ANN model and by comparing its extended version's prediction accuracy on a multi-label classification problem with those of other techniques.

The work implemented in this paper also demonstrates the view presented in Zhuge[1] that the process of summarizing and capturing the mapping rules of the intelligence is also the process of emerging the causal semantic relations between the observing and the understanding. From this point of view, all intelligent behaviors can be simulated and understood through semantic representations and linkages between them,[42] and the method proposed in this paper can provide a new solution to generate the emerging causal relations in a semantic linked network.

ACKNOWLEDGMENTS

This research is supported by the National Natural Science Foundation of China (61402309) and the UK Economic and Social Research Council grant RES-062-23-2721.

ORCID

Juan Yang http://orcid.org/0000-0002-4868-3168

REFERENCES

1. Zhuge H. Multi-dimensional summarization in cyber-physical society: Morgan Kaufmann; 2016.
2. Onnis L, Monaghan P, Christiansen MH, Chater N. Variability is the spice of learning, and a crucial ingredient for detecting and generalizing in nonadjacent dependencies. In: Inproceedings of the Cognitive Science Society, Vol. 26; 2004:26.
3. Mareschal D, Thomas MS. Computational modeling in developmental psychology. *IEEE Trans Evol Comput*. 2007;11(2):137-150.
4. Yang J, Thomas MSC. Simulating intervention to support compensatory strategies in an artificial neural network model of atypical language development,; 2015:123-128.
5. Christiansen MH, Chater N. Connectionist psycholinguistics: capturing the empirical data. *Trends Cogn Sci*. 2001;5(2):82-88.
6. Chomsky N. Aspects of the Theory of Syntax, 11: MIT press; 2014.
7. Chomsky N. Knowledge of language: Its nature, origin and use: Greenwood Publishing Group; 1986.
8. Joanisse MF, Seidenberg MS. Impairments in verb morphology after brain injury: a connectionist model. *Proc Nat Acad Sci*. 1999;96(13):7592-7597.
9. rumelhart DE, McClelland JL, Group PR, et al. Parallel distributed processing: explorations in the microstructure of cognition; 1986.
10. Seidenberg MS, McClelland JL. A distributed, developmental model of word recognition and naming.*Psychol Rev*. 1989;96(4):523.
11. Mcclelland JL, Jenkins E. Nature, nurture, and connections: implications of connectionist models for cognitive development. *in Architectures for intelligence: The twenty-second Carnegie Mellon symposium on cognition*: Lawrence Erlbaum Associates; 1991:41-73.
12. Borovsky A, Elman J. Language input and semantic categories: a relation between cognition and early word learning. *J Child Lang*. 2006;33(04):759-790.
13. Plunkett K, Marchman V. U-shaped learning and frequency effects in a multi-layered perception: implications for child language acquisition. *Cognition*. 1991;38(1):43-102.
14. Hinton GE, Salakhutdinov R. Reducing the dimensionality of the data with neural networks. 2006;313(5786):504-507.
15. Cano A, Zafra A, Ventura S. An interpretable classification rule mining algorithm. *Inf Sci*. 2013;240:1-20.
16. Sutton S, Arnold V, Leech S, Collier P. *The differential use and effect of knowledge-based system explanations in novice and expert judgment decisions*; 2006.
17. Huysmans J, Dejaeger K, Mues C, Vanthienen J, Baesens B. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decis Support Syst*. 2011;51(1):141-154.

18. Breiman L et al. Statistical modeling: the two cultures (with comments and a rejoinder by the author). *Stat Sci*. 2001;16(3):199-231.
19. Castro JL, Mantas CJ, benítez JM. Interpretation of artificial neural networks by means of fuzzy rules. *IEEE Trans Neural Networks*. 2002;13(1):101-116.
20. Mantas CJ, Puche JM, Mantas J. Extraction of similarity based fuzzy rules from artificial neural networks. *Int J Approximate Reasoning*. 2006;43(2):202-221.
21. Rabuñal JR, Dorado J, Pazos A, Pereira J, Rivero D. A new approach to the extraction of ann rules and to their generalization capacity through gp. *Neural Compu*. 2004;16(7):1483-1523.
22. Sweta S, Lal K. Personalized adaptive learner model in e-learning system using FCM and fuzzy inference system. *Int J Fuzzy Syst*. 2017:1-12.
23. Barakat NH, Bradley AP. Rule extraction from support vector machines: a sequential covering approach. *IEEE Trans Knowl Data Eng*. 2007;19(6):729-741.
24. Núñez H, Angulo C, Català A. Rule extraction from support vector machines. *in ESANN*. 2002:107-112.
25. Maji P, Garai P. Fuzzy–rough simultaneous attribute selection and feature extraction algorithm. *IEEE Trans Cybern*. 2013;43(4):1166-1177.
26. Zhu P, Hu Q. Rule extraction from support vector machines based on consistent region covering reduction. *Knowledge Based Syst*. 2013;42:1-8.
27. Chorowski J, Zurada JM. Extracting rules from neural networks as decision diagrams. *IEEE Trans Neural Networks*. 2011;22(12):2435-2446.
28. Chorowski J, Zurada JM. Learning understandable neural networks with nonnegative weight constraints. *IEEE Trans Neural Netw Learn Syst*. 2015;26(1):62-69.
29. Huynh TQ, Reggia JA. Guiding hidden layer representations for improved rule extraction from neural networks. *IEEE Tran Neural Netw*. 2011;22(2):264-275.
30. Andrews R, Diederich J, Tickle AB. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge Based Syst*. 1995;8(6):373-389.
31. Craven MW., Shavlik JW. Extracting tree-structured representations of trained networks. *Adv Neural Inf Process Syst*. 1996:24-30.
32. Kolman E, Margaliot M. Are artificial neural networks white boxes?. *IEEE Trans Neural Networks/a Publ IEEE Neural Netw Counc*. 2005;16(4):844-852.
33. Kolman E, Margaliot M. Extracting symbolic knowledge from recurrent neural networks—a fuzzy logic approach. *Fuzzy Sets Syst*. 2009;160(2):145-161.
34. Fortuny EJ, Martens D. Active learning-based pedagogiacal rule extraction. *IEEE Trans Neural Netw Learn Syst*. 2015;26(11):2664-2677.
35. Plunkett K, Juola P. A connectionist model of english past tense and plural morphology. *Cognitive Sci*. 1999;23(4):463-490.
36. Plunkett K, Marchman V. From rote learning to system building: acquiring verb morphology in children and connectionist nets. *Cognition*. 1993;48(1):21-69.
37. Ward G. Moby pronunciator. 3449 martha ct. Arcata, CA, USA. (Also available at http://icon.shef.ac.uk/Moby/); 1997.
38. Felder R. M. Silverman L. K. Learning and teaching styles in engineering education. *J Eng Educ*. 1988;78(7):674-681.
39. Yang J, Huang Z, Gao Y, Liu HT. Dynamic learning style prediction method based on a pattern recognition technique. *IEEE Tran Learn Technol*. 2014;7(2):166-17.
40. Kohavi R. The power of decision tables. In: in Proc. 8th European Conference on Machine learning ML'95); 1995; Tahoe City, CA, USA:174-189.
41. Holmes G, Hall M, Frank E. Generating rule sets from model trees. In: Proc. Twelfth Australian Joint Conference on Artificial Intelligence; 1999; Tahoe City, CA:1-12.
42. Zhuge H, Xu B. Basic operations, completeness and dynamicity of cyber physical socio semantic link network CPSocio-SLN. *Concurr Comp Pract E*. 2011;23(9):924-939.

How to cite this article: Yang J, Thomas MSC, Liu H. Rule extraction from autoencoder-based connectionist computational models. *Concurrency Computat: Pract Exper*. 2017;e4262. https://doi.org/10.1002/cpe.4262