J. Parallel Distrib. Comput. 117 (2018) 205–211

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A deep stochastic weight assignment network and its application to chess playing

Zhi Wang [a,b], Xizhao Wang [a,*]

[a] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, 518060, China
[b] College of Computer Science and Technology, Hebei University, Baoding, 071002, China

HIGHLIGHTS

- A feature extraction method of Chinese chess middle game situation is proposed.
- The deep stochastic weight assignment network for feature extraction and classification is proposed.
- The auto encoder in DSWAN is constrained by $L_{1/2}$ regularization to make data set more sparse.
- The experiments on the middle game data set of Chinese chess situation applying DSWAN are conducted. The performance is remarkable while the work is very challenging.

ARTICLE INFO

Article history:
Received 22 February 2017
Received in revised form 3 July 2017
Accepted 21 August 2017
Available online 7 September 2017

Keywords:
Chinese chess game
Chess situation analysis
Deep stochastic weight assignment network (DSWAN)
Auto encoder
$L_{1/2}$ regularization
Stochastic weight assignment network (SWAN)

ABSTRACT

Chinese chess is an ancient game in which the chess situation is the information ensemble of chess pieces' spatial locations and interrelations. The situation evaluation plays an extremely important role in the policy decisions of Chinese chess game. However, the situation evaluation is too complex for human to cover every detail with naked eyes. The deep stochastic weight assignment network (DSWAN) proposed in this paper to classify the situations in advantages and disadvantages can solve the above problem. DSWAN is a type of multi-layer perception (MLP) which is divided into two main components: unsupervised feature extractor formed by multi-layer auto encoder and supervised classifier trained by stochastic weight assignment network (SWAN). We summarize a series of chess situation features by gathering specialized knowledges of Chinese chess, and these features are proved valid to estimate the situation. Another highlight of this paper is that the auto encoder is constrained by $L_{1/2}$ regularization. By doing so, it can bring more sparsity to data set, make situation features more representative and ease the overfitting trouble of SWAN.

1. Introduction

Chinese chess is an old game which can be traced back to the Warring States Period. Red side chess player and black side chess player control their own chess pieces, follow common game rules, take flexible strategies, and finally checkmate the opponent's king by capturing chess piece step by step. In the field of artificial intelligence (AI), machine game is a typical research direction. In the imagination of AI researchers, machine should accumulate its own chess experience gradually, master similar strategies, and have capabilities to defeat human players. AI researches in other board games have obtained some gorgeous achievements. Deep Blue, a super computer chess player, which is designed by Hsu [7] defeated temporal world chess champion Kasparov by 3.5 to 2.5 in 1997. A computer program named AlphaGo developed by Silver and Huang [26] beats Go master Lee who is the 9-dan go professional player of South Korea by 4 to 1 in March 2016. The research of Chinese chess machine game began early in the 1980s [31]. But AI research referred to Chinese chess has not obtained such breakthrough like chess [27] and go. It remains on some traditional methods, e.g., alpha–beta pruning [6,21,32,35] to optimize the search algorithm, Chinese chess opening base [30] to store the beginning strategies of the game and transposition table [11,17] to record the node which has been searched to reduce the searching time. So, we import some advanced machine learning methods to Chinese chess game.

In the play of Chinese chess, the evaluation of chess situation is very critical. The chess situation is the information ensemble o

defeated temporal world chess champion Kasparov by 3.5 to 2.5

* Corresponding author.
E-mail address: xizhaowang@ieee.org (X. Wang).

is very critical. The chess situation is the information ensemble o chess pieces' spatial locations and interrelation on chess board ir a step of the Chinese chess game. At the beginning of a game, al the chess pieces are placed on the Chinese chess board by rule

and the initial chess situation arises. In the process of the game, some of the pieces are moved in some position that is decided by the players. With the piece's translocation, a new chess situation generates. Both machine and human players decide to adopt to offensive strategies or defensive strategies by analyzing the chess situation comprehensively. Because of the complexity and fuzziness of the chess situation, a machine can handle this task better than an emotional human player with the help of machine learning methods. Research results already existed are always focused on founding a function [24] to evaluate chess situation. The independent variables of the function are the situation's properties such as relative location of chess pieces, different values of pieces, chess pieces' mobility, and their offensive and defensive relationships. Although these properties are representative, the dependent variable of situation evaluation function is a single value that cannot reflect the details of chess situation. This paper develops a group of Chinese chess situation features which are multi-dimensional to cover every corner of a chess situation. Then the deep stochastic weight assignment network (DSWAN) is proposed to judge the situation.

DSWAN is a multi-layer perception (MLP) whose former two hidden layers as a whole are an auto encoder and the last one hidden layer is a neural network classifier trained by stochastic weight assignment network (SWAN). SWAN is a new rising algorithm [13–16] to train the single-hidden layer feedforward neural networks (SLFNs). The core concept of SWAN is that the weights and biases between the input layer and the hidden one are randomly generated and the weights of output layer are analytically computed through solving a generalized inverse of matrix. Unlike some traditional learning algorithms, e.g., back-propagation (BP) [25] that the weights and biases should be tuned by gradient descent and errors back propagation in iterations, the iteratively tuning is not required in SWAN, so the learning speed is accelerated. And the smallest norm of weights [16] is included in the cost function that the generalization ability is improved. SWAN's fast learning speed and high testing accuracy have attracted many developers [2,3,12,20,36] to adopting it in machine learning projects. Auto encoder [5,10] in the former two hidden layers is an unsupervised feature extractor, in which the SWAN core concept is also adopted. Added $L_{1/2}$ regularization [33,34] in the auto encoder makes the Chinese chess situation data set sparser than the original data set. From the experiments, DSWAN's classification performance is illustrated on the Chinese chess situation data set which is founded from Chinese chess middlegame situations in 50 000 chess manuals.

This paper is organized as follows. Section 2 sets forth the Chinese chess situation feature extracting method and data set. In Section 3, basic SWAN algorithm and auto encoder structure are introduced. We present the proposed DSWAN framework and $L_{1/2}$ regularization sparsity-constrained auto encoder in Section 4. Experiment results are given in Section 5 and Section 6 concludes the paper.

2. Chinese chess situation feature extracting method

Feature extraction is the vital component of machine learning projects [1,4,8,9,23,28]. Highly representative features can accelerate the learning process. We propose a method to extract the features of Chinese chess situation. Chinese chess game [32] tools contain a rectanglular chess board which is printed with 10 horizontal lines and 9 vertical lines, thus there are 90 cross points on the board. On some of these cross points, there are 16 red

object to extract features. And all the pieces' features constitute the situation's features. The features of one Chinese chess piece are as follows.

(1) Existence, in a chess situation, not all the pieces exist on the board. If piece A has been captured, A's existence is assigned to 0, otherwise it is assigned to 1.
(2) Location, the chess board approximates a 10 by 9 matrix. Every piece on board has an independent location information. It is represented by an integer coordinate (X, Y), where X and Y are the row index and column index of Chinese chess board, respectively.
(3) Location-based value, Chinese chess piece's value varies with the change of piece's location. For example, there is Chu River Han Border (CRHB) which divides the board from the middle and 3 by 3 areas called Palace in both sides on the chess board. Soldier A which has crossed CRHB has the higher value than soldier B which has not crossed CRHB. Soldier A would get a higher value if it enters the opposite Palace.
(4) Move number, in the current situation, every piece would have countable legal move methods.
(5) Mobility, the more move methods, the more mobility. Different pieces have different mobility weights. The mobility of one piece is the product of mobility-move weight and the number of legal moves.
(6) Threats, in the current situation, one piece would be captured by an opposite piece in the next step. Every piece is assigned a unique number to be identified. And the threat of one piece equals to the product of the offensive piece's unique number and location-based value.
(7) Protection, in the current situation, one piece's location is in its own side piece's legal move paths. Then the protection of one piece equals to the product of its protective piece's unique number and location-based value.
(8) Assault, in the current situation, one piece can capture an opposite piece. Then the assault of one piece equals to the product of embattled piece's unique number and location-based value.

Eight features can be extracted from every piece. In Chinese chess, the total piece number is 32. So, the total feature number of Chinese chess situation is 256.

For training the performance of chess situation classifier, the game result of each Chinese chess manual is set as the class attribute of each chess situation, i.e., when red player wins the game, the class attribute is 1, otherwise it turns to be 2.

A Chinese chess manual records a complete game process which is from the opening to the end. Each line of the manual notes down one move of one piece. So, a manual consists of a string of chess situations. The middlegame of manual is the most complicated period of the game process so that it is very difficult for human to judge which player is in a dominant position. Therefore, making the middlegame situation as the research object is the most challenging and instructive issue. The Chinese chess situation feature data set of middlegame is designed.

Table 1 denotes the data set which is extracted from the middlegame situations of Chinese chess manual. In this data set, from F_1 to F_8 are eight features of Chinese chess piece A, e.g., F_1 represents whether piece A exists on the chess board; F_2 represents piece A's location which is transformed from the integer coordinate (X, Y); F_3 represents piece A's location-based value.

3. Reviews of SWAN and auto encoder

pieces and 16 black pieces which can be categorized to seven

gorithm are escribed in

Table 1
Chinese chess situation feature data set.

X&F	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	$\cdots$	F_{256}	Class
X_1	1	215	2	3	0	0	3880	0	$\cdots$	0	1
X_2	1	206	11	2	0	0	2231	0	$\cdots$	0	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_N	1	204	11	2	0	0	12571	0	$\cdots$	0	1

3.1. Swan

Stochastic weight assignment network (SWAN) is a fast learning algorithm [15,16] which has excellent regression and classification capability to train the single-hidden layer neural networks (SLFNs).

For N samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ is the data of conditional attributes and $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in R^m$ is the data of class attributes, standard SLFNs with $\tilde{N}$ hidden neurons and activation function g (x) are modeled as

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i \cdot x_j + b_i\right) = o_j, j = 1, \ldots, N \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector connecting the ith hidden neuron and the input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden neuron and the output neurons, and b_i is the threshold of the ith hidden neuron. $w_i \cdot x_j$ denotes the inner product of w_i and x_j.

This model can approximate these N samples with zero error means that $\sum_{j=1}^{N} \left\| o_j - t_j \right\| = 0$, i.e.,

$$\sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i \cdot x_j + b_i\right) = t_j, j = 1, \ldots, N. \tag{2}$$

The above N equations can be written compactly as follows:

$$H\beta = T \tag{3}$$

where

$$H\left(w_1, \ldots, w_{\tilde{N}}, b_1, \ldots, b_{\tilde{N}}, x_1, \ldots, x_N\right) = \begin{bmatrix} g\left(w_1 \cdot x_1 + b_1\right) & \cdots & g\left(w_{\tilde{N}} \cdot x_1 + b_{\tilde{N}}\right) \\ \vdots & \ddots & \vdots \\ g\left(w_1 \cdot x_N + b_1\right) & \cdots & g\left(w_{\tilde{N}} \cdot x_N + b_{\tilde{N}}\right) \end{bmatrix}_{N \times \tilde{N}} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m} \tag{5}$$

where H expresses the output matrix of the hidden layer in neural network; the ith column of H is the ith hidden neuron's output vector.

The essence of SWAN is that the weight w_i and threshold b_i are all randomly generated which is inspired by biological learning. Some parts of brain should have random neurons with parameters independent of the environment. So, the brain can work universally for the feature extraction, clustering, regression and classification within almost zero time.

If the number of hidden neurons is equal to the number of distinct training samples, the matrix H will be square and invertible, and SLFNs can approximate these training samples with zero error. The solution of output weights β is equal to $H^{-1}T$. But generally, the number of hidden neurons is much less than the number of training samples. Thus, H is a non-square matrix and we cannot expect an exact solution of the system. In this case, Eq. (2) could be derived:

$T = H\beta + E$ (6)

where $E = [e_1 \cdots e_N]^T_{N \times m}$ is the training error, so a cost function is defined as follows:

$$J = \sum_{j=1}^{N} \left(\sum_{i=1}^{\tilde{N}} \beta_i g\left(w_i, b_i, x_j\right) - t_j \right)^2 \tag{7}$$

which can be simplified:

$$J = (H\beta - T)^T (H\beta - T) \tag{8}$$

Then the problem turns to be cost function minimization. The solution $\hat{\beta}$ of minimum mean square error (MMSE) is requested:

$$\left\| H\hat{\beta} - T \right\| = \min_{\beta} \|H\beta - T\| \tag{9}$$

with minimum norm of the output weight β for the best generalization performance where $\|\cdot\|$ denotes 2-norm.

The solution of MMSE of above equation is as follows:

$$\hat{\beta} = H^{\dagger}T \tag{10}$$

where $H^{\dagger}$ is the Moore–Penrose generalized inverse [16] of matrix H.

The SWAN training algorithm can be summarized as follows:

Given a training data set $\left\{(x_i, t_i) \,|x_i \in R^n, t_j \in R^m, i = 1, \ldots, N\right\}$ where n is the dimension of input layer and m denotes the dimension of output layer.

(1) Randomly assign the hidden neuron parameters, e.g., the input weights w_i and biases b_i for hidden neurons, i $= 1, \ldots, \tilde{N}$;
(2) Calculate the hidden layer output matrix H;
(3) Obtain the output weight vector using (10).

The orthogonal projection method can be efficiently used for the calculation MP inverse to accelerate the learning speed:

$$\hat{\beta} = \left(H^T H\right)^{-1} H^T T \tag{11}$$

where $H^{\dagger} = \left(H^T H\right)^{-1} H^T$.

Given the testing data set x, the output function of SWAN is

$$\mathrm{f}(x) = h\,(x)\,\hat{\beta} \tag{12}$$

where $h\,(\cdot)$ is the output matrix of the hidden layer with the same input weights and biases as in the training procedure.

3.2. Multi-layer auto encoder

Given a simple SLFN, the learning object of the training data set is the training data itself, i.e., the output matrix of the SLFN is equal to the input matrix, then the auto encoder (AE) [5,10] is founded. The main function of the AE is dimension transformation and feature extraction. From Section 2, the dimension of the Chinese chess situation feature data set is 256. Their effects to predict the advantages and disadvantages of chess situation are different. Some deep-seated relationships of these features are hiding. AE can help change the dimension of data set and the more representative features will be extracted.

$T = H\beta + \varepsilon$ (6) features will be extracted.

For N samples x_i, where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ is the input matrix and output matrix meanwhile. The neurons' number of input layer and output layer of AE are both set as n. The number of the hidden neurons is $\tilde{L}$ which is also the objective dimension of data set to transfer. There are three different types of AE according to the hidden neurons' number:

(1) If $\tilde{L} < n$, the dimension of the coded data set is compressed.
(2) If $\tilde{L} > n$, the dimension of the coded data set is sparse.
(3) If $\tilde{L} = n$, the dimension of the coded data set is equal to the original data set.

The training of AE is an unsupervised learning process, where the SWAN algorithm [18] can be also adopted. The input weights and hidden neurons' biases of the AE are randomly assigned. According to Eq. (10), the output weight $\hat{\beta}_1$ is computed by

$$\hat{\beta}_1 = H^{\dagger}X \tag{13}$$

Then the transposition of output weight $\hat{\beta}_1^T$ is the encoding factor. The coded matrix X_1 of the original data set X is

$$X_1 = g\left(X\hat{\beta}_1^T\right) \tag{14}$$

where $g(\cdot)$ denotes the activation function.

After adding a hidden layer in AE, and the first hidden layer's output matrix X_1 will become the input matrix of the second layer, and in this way, a multi-layer AE is built.

Applying the same method as (14), the coded matrix X_2 of the matrix X_1 is as follows:

$$X_2 = g\left(X_1\hat{\beta}_2^T\right) \tag{15}$$

The main learning process of the multi-layer AE is the confirmation of output weights 2-tuple $\left\langle\hat{\beta}_1, \hat{\beta}_2\right\rangle$. Once the AE is trained to extract proper features, the output weights 2-tuple is fixed and need not be fine-tuned.

4. Deep stochastic weight assignment network

In this section, deep stochastic weight assignment network (DSWAN) for the Chinese chess situation predicting is proposed. A new $L_{1/2}$ Sparsity-Constrained AE is also presented, which is an element of DSWAN.

4.1. DSWAN framework

The deep stochastic weight assignment network (DSWAN) is a five-layer perception shown in Fig. 1 which is constructed from one input layer, one output layer, three hidden layers and parameters among them. The whole framework is divided into two different-acting phases [22,29]. In the former, a multi-layer AE is used to represent the features of the data set. For the latter, the basic SWAN algorithm is utilized for the final decision making.

Given the N-sample data set (x_i, t_i) where $x_i \in R^n$ and $t_i \in R^m$. In the procedure of representation of data set features, the hidden neurons of each hidden layer of AE is L, i.e., the objective dimension of data set features to transform by AE is L. The new feature mapping of the original data set is N by L matrix X_2 which is derived by (14, 15). The features of matrix X_2 is more representative to approximate the class attributes data t_i than the original data set. The new training data set $[X_2|T]$ is the substitution of the original N-sample data set. Then the basic SWAN algorithm is applied as a classifier in the final layer.

Unlike some traditional deep learning frameworks which the parameters need to be retrained iteratively using back propagation (BP) algorithm, the random feature mapping and universal approximation capability of SWAN are fully exploited both in AE and latter phase classifier of DSWAN.

4.2. $L_{1/2}$ sparsity-constrained AE

From Section 3.2, the universal approximation capability of SWAN algorithm is applied in the AE. Then the $L_{1/2}$ regularization [34] is added into the AE optimization to enhance the data set's feature sparsity. It has been proved by Xu [33] that the L_q regularization $(0 < q < 1)$ can generate more sparse solutions than L_1 regularization and meanwhile, the index 1/2 plays a representative role: whenever $q \in [1/2, 1)$, the smaller q is, the sparser solutions of L_q regularization will be obtained. Whenever $q \in (0, 1/2)$, the performance of L_q regularization has no obvious difference, but the solution of $L_{1/2}$ regularization problem is easier to get than the L_0 regularization problem. There have been some works [19] applying the $L_{1/2}$ regularization to deal with sparsity problem. The $L_{1/2}$ regularization problem is as follows:

$$\min_{x\in R^N}\left\{\|H\beta - X\|^2 + \lambda\|\beta\|_{1/2}\right\} \tag{16}$$

where X represents the input data, H denotes the random mapping output, and β is the hidden layer weight to be obtained. Because of the random mapping of the hidden layer feature representation, H is the output of hidden layer that it need not be optimized.

An iterative half thresholding algorithm [33] proposed by Xu provides a fast and effective solver for $L_{1/2}$ regularization problem. It is defined as follows:

$$\beta_{n+1} = G_{\lambda_n\mu_n,\frac{1}{2}}\left(\beta_n + \mu_n H^T(X - H\beta_n)\right) \tag{17}$$

where n denotes the iteration times and $G_{\lambda\mu,1/2}$ is the half thresholding operator:

$$G_{\lambda\mu,1/2}(x) = \begin{cases} f_{\lambda\mu,1/2}(x), & |x| > \frac{\sqrt[3]{54}}{4}(\lambda\mu)^{2/3} \\ 0, & otherwise \end{cases} \tag{18}$$

where

$$f_{\lambda\mu,1/2}(x) = \frac{2}{3}x\left(1 + \cos\left(\frac{2\pi}{3} - \frac{2}{3}\varphi_{\lambda\mu}(x)\right)\right) \tag{19}$$

with

$$\varphi_{\lambda\mu}(x) = \cos^{-1}\left(\frac{\lambda\mu}{8}\left(\frac{|x|}{3}\right)^{-\frac{3}{2}}\right). \tag{20}$$

According to the parameter-setting strategies in [33], Scheme 1 is applied: $\mu_n = \mu_0$; λ_n is chosen by cross-validation. Because the sparsity of the Chinese chess situation feature data set is unknown. If the Chinese chess situation evaluation in this paper is a k-sparsity problem, λ_n can be derived by the method provided in Section III-A in [33]. But the unknown nature of sparsity causes the cross-validation of λ_n, which is barely satisfactory. Here

$$\mu_0 = \frac{(1-\varepsilon)}{\|H\|^2} \text{ with any small } \varepsilon \in (0, 1) \tag{21}$$

The target output matrix of AE is equal to the input matrix X. So ε is equal to 0. Thus, $\mu_0 = \frac{1}{\|H\|^2}$ where $\|H\|^2 = \max\left(eig\left(H^TH\right)\right)$.

Begin the iteration by taking $\beta_0 = 0$ and compute the hidden layer weights β iteratively utilizing (17), we could perfectively extract the representative and compact data set features. Moreover, compared with the traditional AE with least square method, the $L_{1/2}$ regularization optimization is a better solution for the feature extraction and dimension transformation.

5. Performance evaluation and analysis

The experiments are conducted to evaluate the Chinese chess situation predicting performance of DSWAN which is compared to the basic SWAN algorithm. The experiments platform configurations are all listed as follows: Desktop PC, Intel-i5 3.30G CPU, 4G DDR3L RAM, MATLAB R2013a

phase classifier of DSWAN.

DDR3L RAM, MATLAB R2013a.

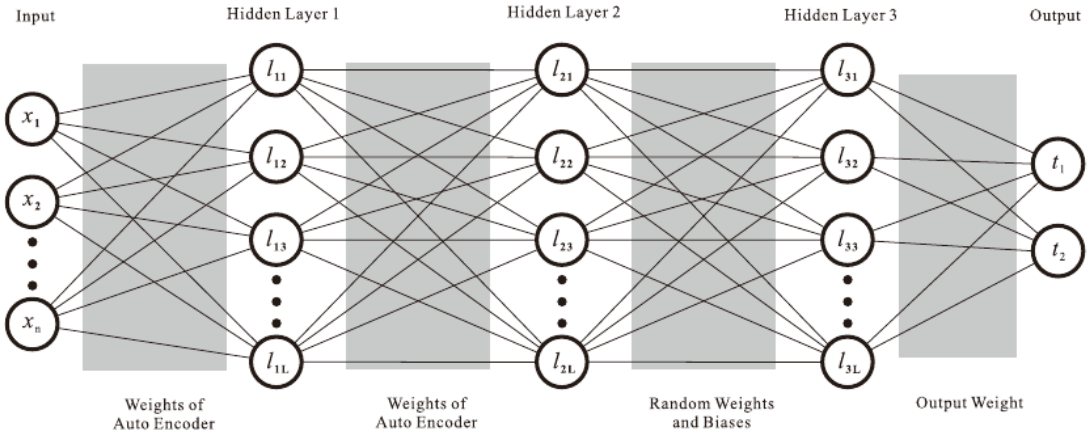


Fig. 1. The structure of DSWAN.

5.1. Chinese chess data set establishment

Chinese chess game is a new field in Machine Learning. The data set of Chinese chess to machine game research job is still a gap. So, the establishment of Chinese chess data set is essential and meaningful. All of the Chinese chess manuals are collected from the Internet and Chinese chess books. Each sample is extracted from one Chinese chess situation which is the middlegame of a match using the method proposed in Section 2. In a manual, only one middlegame situation can be chosen to extract features. The total sample size of data set is 50 000. The victory and defeat of Chinese chess game should be balanced, so the samples' number whose class attributes are 1 or 2 is identical to avoid the imbalance problem in machine learning.

5.2. Comparison between DSWAN and SWAN

In this section, we will evaluate the classification performance of DSWAN and SWAN on the Chinese chess situation data set. The major difference of DSWAN and basic SWAN is that before the SWAN-based classification step, DSWAN utilizes a multi-layer AE to gain the sparse representation of the input data. But, in the basic SWAN, the input data is used for classification directly. The hidden neurons in DSWAN and basic SWAN are both activated by the sigmoid function. All the experiments' results are obtained via 10-fold cross validation.

From Section 3.1, the only specified parameter in the basic SWAN algorithm is the number of hidden neurons L which is set as $\{100, 200, \ldots, 4000\}$.

Fig. 2 shows the training and testing accuracy of the basic SWAN applied in the L hidden neurons' SLFNs. It can be seen that although training accuracy is increasing with the additive hidden neurons, the testing accuracy has nearly arrived the maximum value when L is bigger than 2000.

To fully realize the $L_{1/2}$ sparsity-constrained AE's potential capability in feature extraction, the parameter λ_n in Section 4.2 should be determined by cross validation. For the exploration of the best λ_n, λ_n is set as $\left\{10^{-10}, 10^{-9}, \ldots, 10^{9}, 10^{10}\right\}$. For eliminating the impact of classification step in DSWAN, the neuron number of the last hidden layer is fixed that $l_3 = 1200$. And the number of neurons of the former two hidden layers is equal that $l_1 = l_2$ belongs to {100, 500, 1000, 1500, 2000}. The accuracy of DSWAN is shown in Fig. 3. It can be vividly illustrated that no matter which l_1 (l_2) is chosen, the DSWAN can reach its peak testing accuracy when $\lambda_n = 10^2$. Hereinafter the experiments will all apply this λ_n.

For the experiments conducted utilizing DSWAN, the hidden neurons' number L is also a vital parameter that needs to be

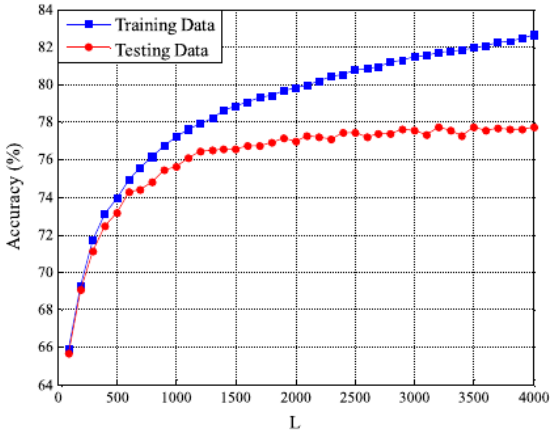


Fig. 2. Training and testing accuracy for the basic SWAN algorithm.

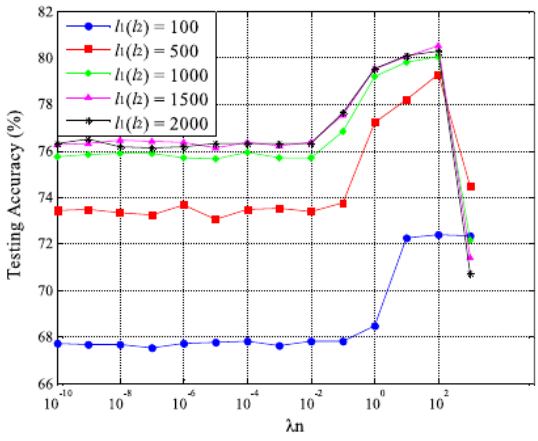


Fig. 3. Accuracy of DSWAN in terms of λ_n.

fixed. Aiming at maximizing the performance of DSWAN, before the evaluation of L, the relationship of the hidden neurons' number of different hidden layers should be sought out. From Section 4.1 we can see that the first two hidden layers of DSWAN are the hidden layers of the multilayer AE. The DSWAN has the same number of neurons as the multilayer AE. Then the relationship of the last one hidden layer and the former two hidden

neurons' number L is also a vital parameter that needs to be tionship of the last one hidden layer and the former two hidde

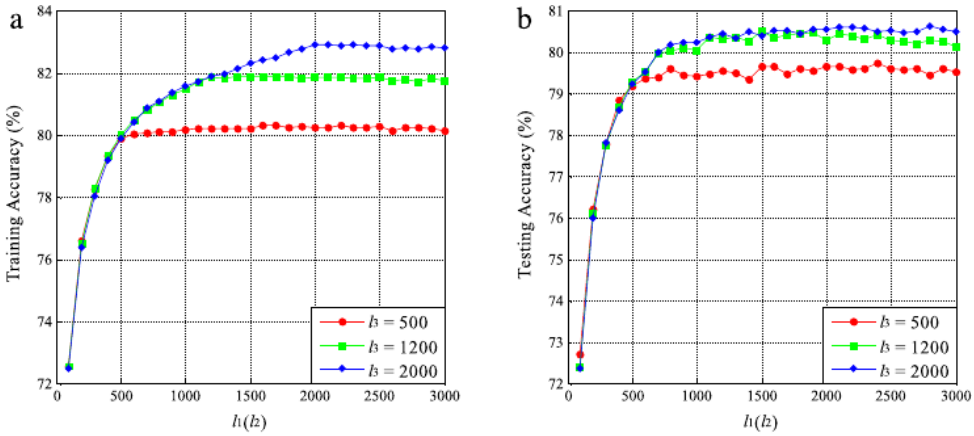


Fig. 4. Accuracy of DSWAN in terms of l_1, l_2 and l_3. (a) Training accuracy of DSWAN. (b) Testing accuracy of DSWAN.

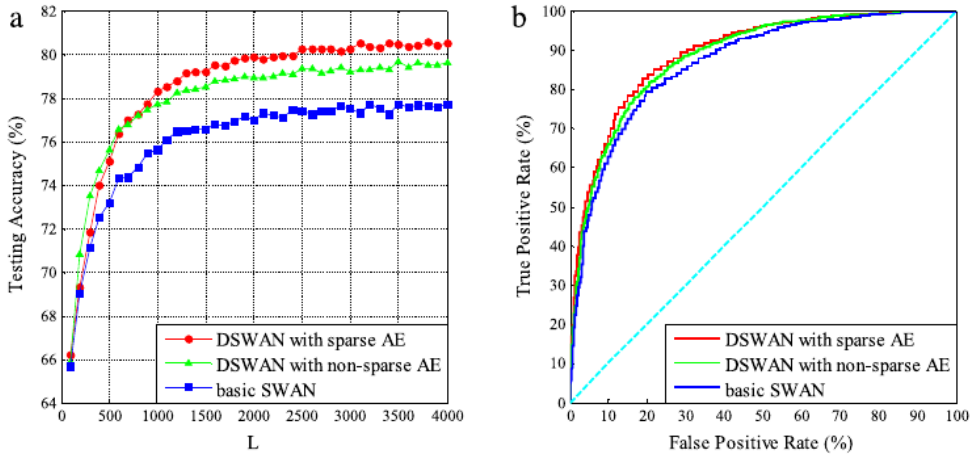


Fig. 5. Performance comparison of DSWAN with sparse AE, DSWAN with non-sparse AE, and basic SWAN. (a) Testing accuracy in terms of L. (b) ROC curve.

layers of DSWAN should be estimated. Suppose that the number of neurons of the former two hidden layers is equal so that $l_1 = l_2$ is the element of $\{100, 200, \ldots, 3000\}$ and the last one layer l_3 is set as $\{500, 1200, 2000\}$. The training and testing accuracies are generated on the Chinese chess situation data set, respectively.

Fig. 4(a) and (b) show the training accuracy and testing accuracy of DSWAN with different hidden layer arrangements, respectively. Because the number of hidden neurons of l_1 and l_2 is equal, their numbers are both represented by $l_1\,(l_2)$ below. Moreover, l_3 represents the number of hidden neurons of the final hidden layer. As $l_1\,(l_2)$ increases, the training accuracy DSWAN changes before the occasion that $l_1\,(l_2) = l_3$. When $l_1\,(l_2) > l_3$, the training accuracy is approximately equal to the accuracy when $l_1\,(l_2) = l_3$. As for the testing accuracy of DSWAN, the accuracy curves with $l_3 = 1200$ and $l_3 = 2000$, respectively, nearly coincide. So, the testing accuracy of DSWAN has reached the convergence to the best performance before $l_1\,(l_2)$ increased to 1200. On the contrary, testing accuracy curve of DSWAN with $l_3 = 500$ is lower than the former two curves when $l_1\,(l_2) > l_3$. Apparently, the DSWAN has not achieved the optimal effect even when $l_1\,(l_2) > l_3$. The conclusion is that in DSWAN, when l_3 is fixed, the structure that $l_1\,(l_2)$ equals to l_3 is the optimum choice for evaluating the best performance. Then in next experiments, the structures of DSWAN are all designed to contain the same size hidden layers.

For comparing and contrasting with the basic SWAN, the total neurons of three hidden layers of DSWAN are set the same number as the basic SWAN. Fig. 5(a) and (b) depicts the performance of DSWAN with sparse AE, DSWAN with non-sparse AE, and basic SWAN. From picture (a), the testing accuracy comparison of these three models is exhibited. From the testing accuracy comparison of these three models, we can conclude that DSWAN's classification capacity is better than the basic SWAN. The feature extraction function of multi-layer AE can really promote the classifier's work. But the DSWAN does not escape the overfitting problem that the testing accuracy increases slower with more hidden neurons. The $L_{1/2}$ regularization added in the multi-layer AE can release the overfitting problem to a certain degree. When the number of hidden neurons is no more than 600, the testing accuracy of the sparse DSWAN is lower than the non-sparse DSWAN. But as the hidden neurons number increases, the testing accuracy of the sparse DSWAN is higher than the non-sparse DSWAN.

Referring to picture (b) in Fig. 5, a receiver operating characteristic (ROC) curve of these three models are painted. ROC curve represents an evaluation criterion of a binary classifier system as its discrimination threshold is varied. The ideal prediction classifier would generate a point in the upper left corner of the ROC picture which denotes that all the samples are classified correctly. The smaller the distance between the ROC curve and the upper left corner of the picture is, the better performance the classifier has. As we can see, the curve of DSWAN is closer to the upper left corner of ROC picture than the curve of basic SWAN. And the curve of sparse DSWAN is on the top left of the curve of non-sparse DSWAN. So, DSWAN can perform better than basic SWAN on Chinese chess

neurons of three hidden layers of DSWAN are set the same number so, DSWAN can perform better than basic SWAN on Chinese chess

situation data set. And the sparsity from $L_{1/2}$ regularization is positive to the classifier.

6. Conclusion

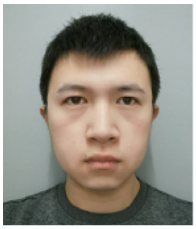
In this paper, we have proposed a set of Chinese chess situation features which can represent the advantages or disadvantages of chess situation to some extent. The Chinese chess situation data set has been found by utilizing these features. In experiments, it has been proved that the Chinese chess situation data set is representative and it fills the gap of lacking the Chinese chess relevant data set. A neoteric MLP which contains $L_{1/2}$ sparsity-constrained AE and SWAN classifier is proposed. DSWAN can achieve the feature extraction and classification on the Chinese chess situation data set. The dimension of Chinese chess situation data set is so high that overfitting problem arises during the learning process. $L_{1/2}$ regularization added in the AE can make mapped features of Chinese chess situation data set sparser. The dimension can be effectively reduced before the SWAN classification phase and overfitting problem is remitted.

Acknowledgments

This work is supported by Basic Research Project of Knowledge Innovation Program in Shenzhen (JCYJ20150324140036825), and National Natural Science Foundation of China (61473194 and 71371063).

References

[1] K.S. Arun, V.K. Govindan, A context-aware semantic modeling framework for efficient image retrieval, Int. J. Mach. Learn. Cybern. 8 (4) (2017) 1259–1285.
[2] Nasser L. Azad, Ahmad Mozaffari, An optimal learning-based controller derived from Hamiltonian function combined with a cellular searching strategy for automotive coldstart emissions, Int. J. Mach. Learn. Cybern. 8 (3) (2017) 955–979.
[3] S. Balasundaram, Deepak Gupta, On optimization based extreme learning machine in primal for regression and classification by functional iterative method, Int. J. Mach. Learn. Cybern. 7 (5) (2016) 707–728.
[4] Neha Baranwal, G.C. Nandi, An efficient gesture based humanoid learning using wavelet descriptor and MFCC techniques, Int. J. Mach. Learn. Cybern. 8 (4) (2017) 1369–1388.
[5] Yoshua Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[6] Shen Cai, Improved pruning strategy of Chinese chess machine game, Foreign Electron. Meas. Technol. 35 (3) (2016) 47–49.
[7] Murray Campbell, A. Joseph Hoane Jr., Feng-hsiung Hsu, Deep blue, Artificial Intelligence 134 (1–2) (2002) 57–83.
[8] Yu Cheng, Zhigang Jin, Hongcai Chen, A fast and robust face recognition approach combining Gabor learned dictionaries and collaborative representation, Int. J. Mach. Learn. Cybern. 7 (1) (2016) 47–52.
[9] Rita Rana Chhikara, Prabha Sharma, A hybrid feature selection approach based on improved PSO and filter approaches for image steganalysis, Int. J. Mach. Learn. Cybern. 7 (6) (2016) 1195–1206.
[10] Shifei Ding, Nan Zhang, Jian Zhang, Unsupervised extreme learning machine with representational features, Int. J. Mach. Learn. Cybern. 8 (2) (2017) 587–595.
[11] Qiang Gao, Chen Guo, Technology of hashing and its application research in hybrid game tree search engine of chinese chess, Sci. Technol. Eng. 8 (17) (2008) 4869–4872.
[12] Yu-Shan Hsu, Sin-Jin Lin, An emerging hybrid mechanism for information disclosure forecasting, Int. J. Mach. Learn. Cybern. 7 (6) (2016) 943–952.
[13] Guangbin Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Trans. Neural Netw. 14 (2) (2003) 274–281.
[14] Guangbin Huang, H.A. Babri, Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions, IEEE Trans. Neural Netw. 9 (1) (1998) 224–229.
[15] Guangbin Huang, Qinyu Zhu, Chee-Kheong Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of International Joint Conference on Neural Networks, vol. 2, 2004, pp. 985–990.
[16] Guangbin Huang, Qinyu Zhu, Chee-Kheong Siew, Extreme learning machine: Theory and applications, Neuralcomputing 70 (1) (2006) 489–501.
[17] Shangbin Jiao, Ding Liu, Research on translation table heuristic algorithm, Comput. Eng. Appl. 46 (6) (2010) 42–45.
[18] L.L.C. Kasun, H. Zhou, G. Huang, C.M. Vong, Representational learning with extreme learning machine for big data, IEEE Intell. Syst. 28 (6) (2013) 31–34.
[19] Zhenni Li, Takafumi Hayashi, Shuxue Ding, Dictionary learning with the $\ell 1/2$-regularizer and the coherence penalty and its convergence analysis, Int. J. Mach. Learn. Cybern. (2017). http://dx.doi.org/10.1007/s13042-017-0649-9.
[20] Peng Liu, Yihua Huang, Lei Meng, Siyuan Gong, Two-stage extreme learning machine for high-dimensional data, Int. J. Mach. Learn. Cybern. 7 (5) (2016) 765–772.
[21] Shuying Liu, Yuanbiao Mu, Hong Li, Game design of chinese chess based on alpha-beta pruning search algorithm, Inf. Commun. (8) (2015) 47–48.
[22] Yang Luo, Benqiang Yang, Lisheng Xu, Segmentation of the left ventricle in cardiac mri using a hierarchical extreme learning machine model, Int. J. Mach. Learn. Cybern. (2017). http://dx.doi.org/10.1007/s13042-017-0678-4.
[23] Iqbal Nouyed, Bruce Poon, M. Ashraful Amin, A study on the discriminating characteristics of Gabor phase-face and an improved method for face recognition, Int. J. Mach. Learn. Cybern. 7 (6) (2016) 1115–1130.
[24] Shouyu Qian, Minfang Ge, The computer aid analysis for game of Chinese chess, Microelectron. Comput. (5) (1987) 39–42.
[25] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representation by back-propagation errors, Nature 323 (323) (1986) 533–536.
[26] David Silver, Aja Huang, Chris J. Maddison, Mastering the game of Go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[27] Boris Stilman, Proximity reasoning for discoveries, Int. J. Mach. Learn. Cybern. 7 (1) (2016) 53–84.
[28] Xiao-yun Sun, Feng-ning Kang, Ming-ming Wang, Improved probabilistic neural network PNN and its application to defect recognition in rock bolts, Int. J. Mach. Learn. Cybern. 7 (5) (2016) 909–919.
[29] Jiexiong Tang, Chenwei Deng, Guangbin Huang, Extreme learning machine for multilayer perceptron, IEEE Trans. Neural Netw. Learn. Syst. 27 (4) (2016) 809–821.
[30] Qingang Wei, Jiao Wang, Xinhe Xu, A study and design of opening-book of computer chinese chess, Caai Trans. Intell. Syst. 2 (1) (2007) 85–89.
[31] Shunqin Xu, Donghui Huang, Design of Chinese chess knowledge base, in: Proceedings of the National Computer Conference in 1985, 1985, pp. 505–509.
[32] Xinhe Xu, Jiao Wang, Key technologies analysis of Chinese chess computer game, J. Chin. Comput. Syst. 27 (6) (2006) 961–969.
[33] Zongben Xu, Xiangyu Chang, Fengmin Xu, $L_{1/2}$ regularization: a thresholding representation theory and a fast solver, IEEE Trans. Neural Netw. Learn. Syst. 23 (7) (2012) 1013–1027.
[34] Zongben Xu, Hai Zhang, Yao Wang, $L_{1/2}$ regularization, Sci. China Inf. Sci. 53 (6) (2010) 1159–1169.
[35] Jinpeng Yue, Su Feng, Application of game tree search algorithm in computer Chinese chess, Comput. Syst. Appl. 18 (9) (2009) 140–143.
[36] Jian Zhang, Shifei Ding, Nan Zhang, Incremental extreme learning machine based on deep feature embedded, Int. J. Mach. Learn. Cybern. 7 (1) (2016) 111–120.



Zhi Wang received the B.Sc. degrees from Hebei University, Baoding, China, in 2014. He is currently a M.Sc. candidate of the School of Computer Science and Technology, Hebei University. His main research interests include machine game and neural networks.

Xi-Zhao Wang received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1998. He is currently a professor with the College of Computer Science and Software Engineering, Shenzhen University, Guangdong, China. From September 1998 to September 2001, he was a research fellow with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His current research interests include learning from examples with fuzzy representation, fuzzy measures and integrals, neuro fuzzy systems and genetic algorithms, feature extraction, multi-classifier fusion, and applications of machine learning.