Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A knowledge matching approach based on multi-classification radial basis function neural network for knowledge push system*

Shu-you ZHANG, Ye GU, Guo-dong YI‡, Zi-li WANG

State Key Laboratory of Fluid Power and Mechatronic Systems, Zhejiang University, Hangzhou 310027, China

E-mail: zsy@zju.edu.cn; me_guye@zju.edu.cn; ygd@zju.edu.cn; ziliwang@zju.edu.cn

Received Jan. 31, 2019; Revision accepted May 9, 2019; Crosschecked Mar. 6, 2020; Published online Apr. 25, 2020

Abstract: We present an exploratory study to improve the performance of a knowledge push system in product design. We focus on the domain of knowledge matching, where traditional matching algorithms need repeated calculations that result in a long response time and where accuracy needs to be improved. The goal of our approach is to meet designers' knowledge demands with a quick response and quality service in the knowledge push system. To improve the previous work, two methods are investigated to augment the limited training set in practical operations, namely, oscillating the feature weight and revising the case feature in the case feature vectors. In addition, we propose a multi-classification radial basis function neural network that can match the knowledge from the knowledge base once and ensure the accuracy of pushing results. We apply our approach using the training set in the design of guides by computer numerical control machine tools for training and testing, and the results demonstrate the benefit of the augmented training set. Moreover, experimental results reveal that our approach outperforms other matching approaches.

Key words: Product design; Knowledge push system; Augmented training set; Multi-classification neural network; Knowledge matching

https://doi.org/10.1631/FITEE.1900057 **CLC number:** TP391

1 Introduction

Product design process is an inherently knowledge-intensive activity. Designers, especially the unskilled ones, need to frequently consult established design knowledge in the traditional design process. Knowledge push technology is developed to push the right knowledge to the right person at the right time in the right way (Schreiber et al., 2000). Knowledge push, as an application of knowledge-based engineering, can improve efficiency and accuracy in a design process (Fan et al., 2005). A standard process framework of knowledge push is shown in Fig. 1. Knowledge matching is the most important part of knowledge push, and consists of the mission of selecting the correct design knowledge relevant to design tasks or designers' demands.

In Zhang SY et al. (2018), we investigated two parts of knowledge push, i.e., knowledge matching and personalized pushing, and developed a new approach for these two parts. At the heart of our approach to knowledge matching is the applicable probability (AP) classifier between design knowledge and design content. The fundamental principle of the approach is based on a knowledge-case training set and the Bayesian theorem (Fig. 2a). We tested this approach in the design of guides by computer numerical control (CNC) machine tools with 35 training

‡ Corresponding author

* Project supported by the National Key R&D Project of China (No. 2018YFB1700700) and the National Natural Science Foundation of China (No. 51675478)

ORCID: Shu-you ZHANG, https://orcid.org/0000-0001-9023-5361; Guo-dong YI, https://orcid.org/0000-0002-7711-7982

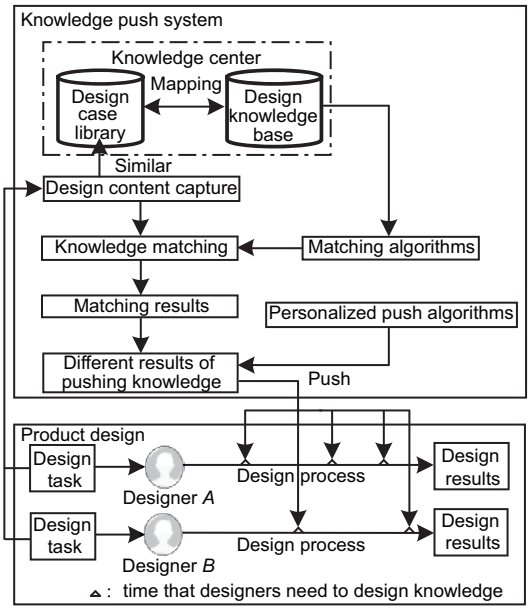


Fig. 1 Knowledge push framework

samples, and the accuracy and response time of the tests are shown in Fig. 2b. This approach enables the automation of knowledge matching and requires only a set of training samples for accuracy.

However, there are two obvious shortcomings in Zhang SY et al. (2018). The response time of knowledge matching (Fig. 2b) is long and impairs the human-computer interaction experience. The AP classifier needs to match the design knowledge with the design content one by one, and this is the main reason for the long response time. With the increase of design knowledge in the future, the problem of long response time urgently needs to be solved. In addition, the accuracy of knowledge matching is around 60%, which needs to be improved. The training set for the illustrative example has only 35 samples. We have discussed that more training samples can make the push knowledge more accurate (Zhang SY et al., 2018).

In this study, we shift our focus to shortening the response time and improving the accuracy in knowledge matching:

1. To shorten the response time, we completely replace the knowledge matching mode in Fig. 2a with artificial neural networks (ANNs). The neurons in the output layer represent the design knowledge, and the input is the original design content vector (i.e., **content**) (Fig. 2a). As a result, there are no required modifications to the original input and training set to accommodate the new matching approach. We propose some key enhancements to the ANNs to boost performance in knowledge matching.

2. To improve the accuracy, the limited training set in a practical operation needs to be augmented in knowledge matching. We investigate two methods to augment the training set, namely, oscillating the feature weight and revising the case feature in the case feature vectors (i.e., **pro**'s). Our technical approaches and experiments will be detailed later.

2 Related works

2.1 Knowledge push

Previous investigations of knowledge push use algorithms from a recommender system, including collaborative filtering (Xu et al., 2013; Feng et al., 2016), content-based filtering (Liu TY et al., 2016; Wang et al., 2016), and hybrid recommendation (Liang et al., 2015), which are adapted for a specialty in product design. The process framework

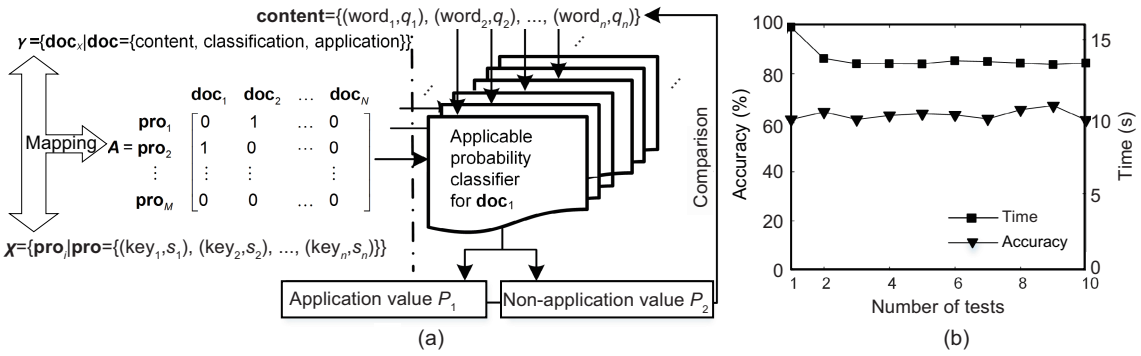


Fig. 2 Overview of the knowledge matching approach: (a) a knowledge-case training set and applicable probability classifiers; (b) accuracy and response time of the approach

of knowledge push (Fig. 1) shows some important fields for study. Knowledge push usually relies on a design activity workflow (Le et al., 2010; Jiang et al., 2017) or multidimensional hierarchical context (Zhang FP and Li, 2016; Zhang SY et al., 2018). To capture the design intent in a design process for an active knowledge push service, Xu et al. (2013) developed the collaborative design and an active knowledge service platform to realize intent capture and intent evolution. Wang et al. (2016) built a design intent model with a formal expression of a target function of conceptual design. Zhang SY et al. (2019) proposed trust-aware and item-cluster strategies to extract two nearest neighbors of the designers and the design knowledge to improve the quality of knowledge push. Knowledge base construction has been developed, e.g., affinity management of knowledge (Zhang LL et al., 2009), the knowledge ontology model (Ji et al., 2013), and the variable-weight layered spreading activation model (Liang et al., 2015). Knowledge matching is the most important field, and is detailed in Section 2.2. Furthermore, personalized push is an advanced requirement for knowledge push and is developing. While some techniques have been proposed (Xiao et al., 2010; Chen et al., 2015; Wang et al., 2016), we think that these techniques are difficult to apply to practical operations in a short time.

2.2 Knowledge matching

Traditional knowledge matching algorithms calculate the similarities between designers or design tasks, including similarity of text (Yan et al., 2016), attribute similarity algorithm (Zhang FP and Li, 2017; Wu LJ et al., 2018), and semantic distance (Xu et al., 2013; Wang et al., 2016). Shortcomings of these matching algorithms are data sparsity and repeated calculations. Feng et al. (2016) mapped the semantic matrix to a low-dimensional space by singular value decomposition. Zhang K et al. (2019) developed a quality function knowledge deployment (QFKD) based knowledge push system. Requirement-function (RF) and function-knowledge (FK) mapping modes and similarity calculation are used for knowledge matching. Other investigations use intelligence techniques for knowledge matching, e.g., artificial immune algorithm (Dong et al., 2013), clustering collection model (Li et al., 2017), ontology mapping-merging-reasoning (Zhang C et al., 2018), and applicable probability matching algorithm (Zhang SY et al., 2018). More and more intelligence algorithms have been introduced in knowledge matching, and perform better than traditional algorithms. Knowledge push is a multi-classification problem and also an engineering application example of recommender systems. Although there has been a boom in deep learning, neural network based models have not been used in knowledge matching.

2.3 Artificial neural networks

ANNs have been applied to solve the multi-classification problems in many different recommender systems, except knowledge push in the product design process. ANNs are used to learn and predict the user-item rating data in recommender systems for a collaborative filtering algorithm (Gupta and Tripathy, 2014; Paradarami et al., 2017). Gabrani et al. (2017) reviewed the artificial intelligence techniques used in recommender systems. Devi et al. (2010) and Bahramian et al. (2017) discovered that ANNs are suitable for approximation and classification and effective in handling cold start problems. Data sparsity needs to be resolved in user similarity calculation. Devi et al. (2010) calculated the trust between users based on a rating matrix in a probabilistic neural network, and relieved the sparsity by smoothing a sparse rating matrix. Pushpa et al. (2013) found that the radial basis function (RBF) neural network produces the most relevant results compared with an aggregation technique in web page recommendations. Twardowski (2016) combined explicit context modeling with factorization methods and a novel approach with a recurrent neural network in the recommendation process. Sunhem and Pasupa (2016) compared linear discriminant analysis (LDA), ANN, and support vector machine (SVM), and found that SVM with RBF is the best in hairstyle recommendation. Xue et al. (2017) proposed a novel matrix factorization model with a neural network architecture to predict personalized ranking. Guo et al. (2018) employed an improved RBF to determine the weights of recommendations in a mobile e-commerce recommendation system. Wu H et al. (2018) used dual-regularized matrix factorization with deep neural networks to exploit description documents of both users and items in matrix factorization. Furthermore, a neural network was used to extract

features from users' profiles and tag spaces in depth layer by layer, for higher accuracy in a recommender system (Zuo et al., 2016). ANN or other deep learning methods can avoid repeated calculations in matching the items with users and improve the performance compared with traditional methods. The neural network architectures need to be improved in different recommender systems for different characteristics.

3 Technical approach

The approach proposed for knowledge matching largely relies on Zhang SY et al. (2018). We provide two key modifications to the original work, which are detailed in this section. Unless otherwise stated, we use the same parameters as in Zhang SY et al. (2018) (Table 1).

Table 1 Symbols used in this paper

Symbol	Definition
content	Design content vector, **content**= $\{(\text{word}_1, q_1), (\text{word}_2, q_2), \ldots, (\text{word}_n, q_n)\}$
pro	Case feature vector, **pro**= $\{(\text{key}_1, s_1), (\text{key}_2, s_2), \ldots, (\text{key}_n, s_n)\}$
doc	Knowledge description vector
$\boldsymbol{\gamma}$	Knowledge base, $\boldsymbol{\gamma}=\{\mathbf{doc}_1, \mathbf{doc}_2, \ldots, \mathbf{doc}_N\}$
$\boldsymbol{\chi}$	Design case library, $\boldsymbol{\chi}=\{\mathbf{pro}_1, \mathbf{pro}_2, \ldots, \mathbf{pro}_N\}$
$\boldsymbol{A}$	Boolean matrix, generated by $\boldsymbol{\gamma}$ and $\boldsymbol{\chi}$ (Fig. 2a)
key	Union of keys in each **pro** (Fig. 3)
$\boldsymbol{s}$	Corresponding weight matrix in $\boldsymbol{\chi}$ (Fig. 3)
$\boldsymbol{r}$	Matching results of knowledge push
M	Number of design cases in $\boldsymbol{\chi}$
N	Number of pieces of knowledge in $\boldsymbol{\gamma}$
n	Number of keys in **key**

3.1 Augmented training set

The training set for knowledge matching includes knowledge base $\boldsymbol{\gamma}$, design case library $\boldsymbol{\chi}$, and a Boolean matrix $\boldsymbol{A}$. The case feature vectors are integrated as the design case library in Fig. 3, where key and s ($s \in [0,1]$) are the case feature and feature weight in **pro**, respectively. In addition, $\mathbf{key}^i$ and $\boldsymbol{s}^i$ are the matrices of key and s in $\mathbf{pro}^i$, respectively, and the elements in these two sets are placed in one-to-one correspondence. In $\boldsymbol{\chi}$, the case feature vectors are integrated by the case features, where $\mathbf{key} = \bigcup_{k=1}^{n}\bigcup_{i=1}^{M}\text{key}_k^i$. Moreover, $\boldsymbol{s}$ is the sparse matrix, and s_k^i is assigned to $s_{i,m}$ when key_k^i (belonging to $\mathbf{key}^i$) is equal to key_m (belonging to **key**); otherwise, $s_{i,m}=0$. We can augment the training set by applying transformations to each case feature vector **pro**, such as the feature weight s or the case feature key.

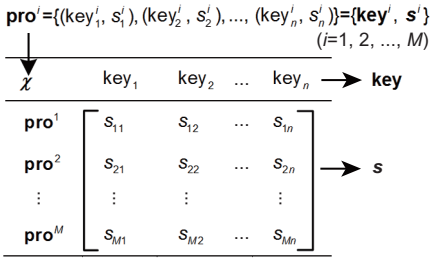


Fig. 3 Integration process in design case library $\boldsymbol{\chi}$

3.1.1 Oscillation of the feature weight s

We found that the appropriate changes in s may not affect the Boolean matrix $\boldsymbol{A}$ and that the pushing results are used in the hierarchical design content model (Zhang SY et al., 2018). Two oscillatory functions, i.e., $f_1(s)$ and $f_2(s)$, are constructed to expand the value of s. $f_1(s)$ is based on the trigonometric function and $f_2(s)$ on the exponential function, expressed as follows:

$$f_1(s)=A_0\left[A_1\arctan(\omega_1 s+\varphi_1)+A_2\arcsin(\omega_2 s+\varphi_2)+A_3\arccos(\omega_3 s+\varphi_3)+A_4\right], \quad (1)$$

$$f_2(s)=B_1 e^{\delta_1[(x+\mu_1)/\phi_1]^2}+B_2 e^{\delta_2[(x+\mu_2)/\phi_2]^2}+B_3. \quad (2)$$

These two functions need to oscillate along the direction of $y=x$. In this study, we set $A_0=\pi/2$, $A_1=A_2=1$, $A_3=0$, $A_4=\pi$, $\omega_1=10$, $\omega_2=2$, $\varphi_1=-5$, $\varphi_2=-1$, $\omega_3=\varphi_3=0$, $B_1=0.99$, $B_2=0.35$, $B_3=0$, $\delta_1=\delta_2=-1$, $\mu_1=-0.95$, $\mu_2=-0.39$, $\phi_1=0.38$, and $\phi_2=0.19$. The oscillatory functions are shown in Fig. 4, with the initial value of weight (s) on the x axis against the expanded value of weight (s') on the y axis. Then, the number of design cases in $\boldsymbol{\chi}$ is $3M$. There are other oscillatory functions when changing the parameters in Eqs. (1) and (2).

3.1.2 Revision of the case feature key

When changing the key in **pro**, the s in **pro** and the corresponding design knowledge in $\boldsymbol{A}$ also need to be changed to generate a new case feature vector in a process called "revision." Sometimes, two design cases in $\boldsymbol{\chi}$ are similar to a few different keys in

pro, and then the pushing knowledge in $\boldsymbol{A}$ is different. As shown in Fig. 5, $\mathbf{pro}^i$ and $\mathbf{pro}^j$ are similar except for one key, and the corresponding feature weights are certainly different. The special different key causing the different pushing knowledge can be found as $\mathbf{pro}^i - \mathbf{pro}^j$ with $\boldsymbol{A}(i,:) - \boldsymbol{A}(j,:)$ and $\mathbf{pro}^j - \mathbf{pro}^i$ with $\boldsymbol{A}(j,:) - \boldsymbol{A}(i,:)$ (Fig. 5). Then we can revise a new case feature vector $\mathbf{pro}^k$ without the special different key. The generated vector $\mathbf{pro}^{k_1}$ with the corresponding design knowledge $\boldsymbol{A}(k_1,:)$ is shown in Fig. 5, where $\mathbf{pro}^{k_1} = \mathbf{pro}^k \cup (\mathbf{pro}^j - \mathbf{pro}^i)$, $\boldsymbol{A}(k_1,:) = \boldsymbol{A}(k,:) \cup (\boldsymbol{A}(j,:) - \boldsymbol{A}(i,:))$. Also, we can revise other case feature vectors to augment the training set under rational conditions.

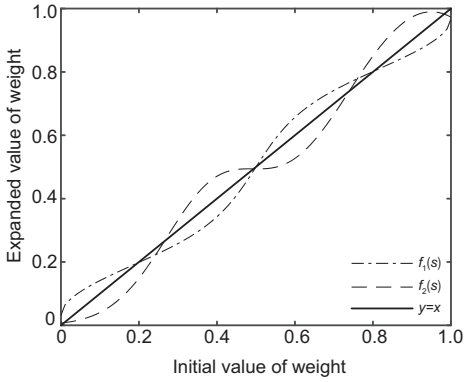


Fig. 4 Two oscillatory function curves

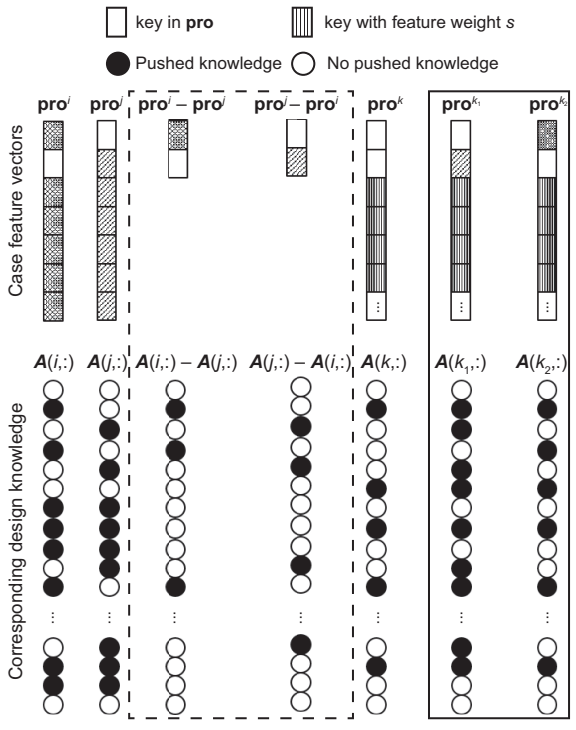


Fig. 5 Revision of the case feature key

3.2 Multi-classification radial basis function neural network

ANNs can match the knowledge from the knowledge base once, radically shortening the response time. Fig. 6a gives an illustration of a neural network for knowledge matching. As shown in Fig. 6a, the **pro**s in the case library and the Boolean matrix $\boldsymbol{A}$ are used for training the neural network, and then the trained neural network is used for testing or application. The input neurons are key_m $(m = 1, 2, \ldots, n)$ in **key** in $\boldsymbol{\chi}$ (Fig. 3), and the output neurons are knowledge $\mathbf{doc}_x$ $(x = 1, 2, \ldots, N)$. The training input data is s in different **pro**s and the training output data is the design knowledge corresponding to **pro**.

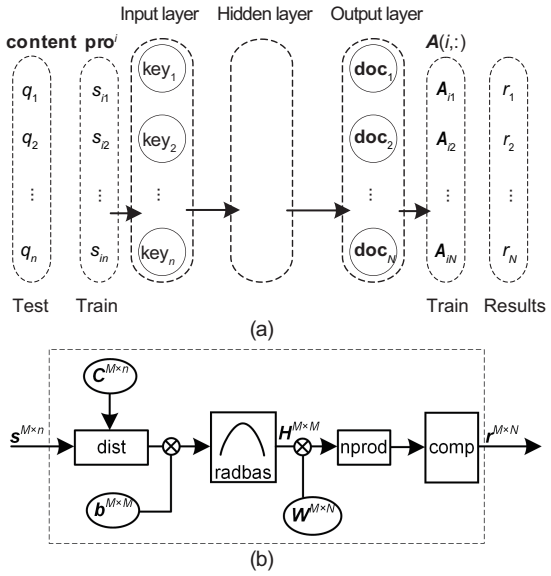


Fig. 6 The proposed neural network for knowledge matching: (a) input and output layers; (b) hidden layer

The input vector for testing is the design feature vector (**content**) with the same form as **pro**. Results for testing are $\{r_x | x \in [1, N]\}$, where r_x is 0 or 1, and where the matched knowledge $\{\mathbf{doc}_x | r_x = 1\}$ will be pushed to designers in their own design process. The input and output layers are fixed in Fig. 6a, and more neurons in the output layer can cater to the increase of design knowledge. The hidden layer is the key point for the performance of the neural network.

Different from the traditional classification problems, knowledge push is a multi-classification problem, where we want to match the knowledge once to shorten the response time. In this study, we propose a multi-classification radial (MCR) basis

function neural network instead of the popular back propagation (BP) neural network. BP has been widely used to solve some multi-classification problems. The main disadvantages are the uncertain hidden layer and the different neurons in the input and output layers of different designs of products. Researchers need to continually test and debug arrangements, which is repeated and tedious. Different designs of products have different neurons in the input and output layers. BP has a disadvantage in the local optimum. SVM is widely used in dual classification. SVM needs to be improved to solve the multi-classification problem and is not good with a large-scale training dataset. However, the knowledge push system in the design of complex equipment usually has large-scale datasets. The proposed MCR is based on the traditional RBF. RBF has a certain hidden layer and a shorter training time than BP, and can deal with large-scale data. The hidden layer of the neural network is shown in Fig. 6b, where the input is $\boldsymbol{s}^{M\times n}$ (Fig. 3) and the output is $\boldsymbol{r}^{M\times N}$ (Fig. 2a). The boxes in Fig. 6b represent different functions; i.e., "dist" is the distance between input $\boldsymbol{s}$ and center matrix $\boldsymbol{C}$ (Eq. (3)), "radbas" is a Gaussian function as the activation function of neurons (Eq. (4)), "nprod" is the normalized operation of an element in $\boldsymbol{H}\times\boldsymbol{W}$ (Eq. (5)), and "comp" is the competitive function to obtain the matched knowledge in the competitive neurons (Eq. (6)):

$$\mathrm{dist}_{i,j}=\|\boldsymbol{s}^i-\boldsymbol{C}^j\|=\sqrt{\sum_{m=1}^{n}(s_{i,m}-C_{j,m})^2}, \tag{3}$$

$$H_{i,j}=\mathrm{radbas}(\mathrm{dist}_{i,j}\cdot b_{i,j})=\mathrm{e}^{-(\mathrm{dist}_{i,j}\cdot b_{i,j})^2}, \tag{4}$$

$$\mathrm{nprod}_{i,j}=\mathrm{norm}(\boldsymbol{H}\cdot\boldsymbol{W}) \tag{5}$$

$$=\frac{\sum\limits_{k=1}^{M}H_{i,k}W_{k,j}-\min\limits_{l}\left(\sum\limits_{k=1}^{M}H_{i,k}W_{k,l}\right)}{\max\limits_{l}\left(\sum\limits_{k=1}^{M}H_{i,k}W_{k,l}\right)-\min\limits_{l}\left(\sum\limits_{k=1}^{M}H_{i,k}W_{k,l}\right)},$$

$$r_{i,j}=\mathrm{comp}(\mathrm{nprod}_{i,j})=\begin{cases}1, & \mathrm{nprod}_{i,j}>\alpha,\\ 0, & \mathrm{nprod}_{i,j}\le\alpha,\end{cases} \tag{6}$$

where $\boldsymbol{s}^i$ is the i^{th} row in $\boldsymbol{s}$, $\boldsymbol{C}^j$ the j^{th} row in the center matrix $\boldsymbol{C}$ with the size of $M\times n$, $\boldsymbol{b}$ the threshold matrix with the size of $M\times M$, $\boldsymbol{W}$ the weight matrix with the size of $M\times N$, and $l=1,2,\ldots,N$.

We enhance RBF in two parts, i.e., "nprod" and "comp," for the characteristics in knowledge matching. A normalized operation has been used in many neural networks (Yang et al., 2007; Liu HM et al., 2009). The input is the feature weight $s\in[0,1]$. The values of training data are widely distributed with "dist" and "radbas," and the output values belong to $(0,1)$. To ensure the accuracy and speed when training the network, the values in $\boldsymbol{H}\times\boldsymbol{W}$ are normalized in the fixed interval of $[0,1]$, and there is no need for anti-normalization. Owing to the characteristics of knowledge matching, we need to obtain the specific design knowledge for pushing. The value of output is 1 or 0, with 1 denoting that the knowledge matches the design content and 0 denoting that the knowledge does not match the design content. Thus, "comp" is necessary for classification, as shown in Eq. (6), and α is the classification threshold, rigorously defined as 0.9 in this study.

$\boldsymbol{C}$, $\boldsymbol{b}$, and $\boldsymbol{W}$ are the unknown parameters in the neural network. The training set $\{\boldsymbol{s}^{M\times n},\boldsymbol{A}^{M\times N}\}$ is used to train the neural network and to calculate these three parameters. We use a supervised learning algorithm based on gradient descent to train the neural network (Karayiannis, 1999). The loss function E is

$$E=\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N}(r_{i,j}-A_{i,j})^2, \tag{7}$$

where $\boldsymbol{r}=\boldsymbol{W}\exp[-(\|\boldsymbol{s}-\boldsymbol{C}\|\cdot\boldsymbol{b})^2]$. The gradient descent algorithm for training the neural network is shown in Algorithm 1, where η is the value of the step size and ε the convergence threshold.

In Algorithm 1, $\boldsymbol{C}_0$, $\boldsymbol{b}_0$, and $\boldsymbol{W}_0$ are the initial values from the initialized cluster centers, as shown in Eqs. (8), (9), and (10), respectively. In particular, we have

$$C_{0,j,i}=\min i+\frac{\max i-\min i}{2M}+(j-1)\frac{\max i-\min i}{M}, \tag{8}$$

where $i=1,2,\ldots,n$, $j=1,2,\ldots,M$, $\max i$ is the maximum value and $\min i$ the minimum value of the i^{th} input neuron.

$$b_{0,i,j}=\frac{1}{n}\sum_{k=1}^{n}n(s_{i,k}-C_{j,k}), \tag{9}$$

where $i=1,2,\ldots,n$, $j=1,2,\ldots,M$.

$$W_{0,i,j}=\min i+j\frac{\max i-\min i}{N+1}, \tag{10}$$

where $i=1,2,\ldots,n$, $j=1,2,\ldots,N$, $\max i$ is the expected maximum value and $\min i$ the expected minimum value of the i^{th} output neuron.

Algorithm 1 Training the proposed neural network

Input: $\boldsymbol{C}_0$, $\boldsymbol{b}_0$, $\boldsymbol{W}_0$, η, and ε
Output: $\boldsymbol{C}_{t+1}$, $\boldsymbol{b}_{t+1}$, and $\boldsymbol{W}_{t+1}$
$t=0$
while $\Delta E>\varepsilon$ **do**
  $\boldsymbol{C}_{t+1}=\boldsymbol{C}_t-\eta\frac{\partial E}{\partial \boldsymbol{C}}|_t$,
  $\boldsymbol{b}_{t+1}=\boldsymbol{b}_t-\eta\frac{\partial E}{\partial \boldsymbol{b}}|_t$,
  $\boldsymbol{W}_{t+1}=\boldsymbol{W}_t-\eta\frac{\partial E}{\partial \boldsymbol{W}}|_t$,
  $\Delta E=E_{t+1}-E_t$,
  $t=t+1$
end while

$\frac{\partial E}{\partial \boldsymbol{C}}$, $\frac{\partial E}{\partial \boldsymbol{b}}$, and $\frac{\partial E}{\partial \boldsymbol{W}}$ are the partial derivatives of E with respect to $\boldsymbol{C}$, $\boldsymbol{b}$, and $\boldsymbol{W}$, respectively, expressed as

$$\frac{\partial E}{\partial \boldsymbol{C}}=2(\boldsymbol{r}-\boldsymbol{A})\boldsymbol{W}\boldsymbol{b}^2\exp[-(\|\boldsymbol{s}-\boldsymbol{C}\|\cdot\boldsymbol{b})^2], \quad (11)$$

$$\frac{\partial E}{\partial \boldsymbol{b}}=-2(\boldsymbol{r}-\boldsymbol{A})\boldsymbol{W}\|\boldsymbol{s}-\boldsymbol{C}\|^2\exp[-(\|\boldsymbol{s}-\boldsymbol{C}\|\cdot\boldsymbol{b})^2], \quad (12)$$

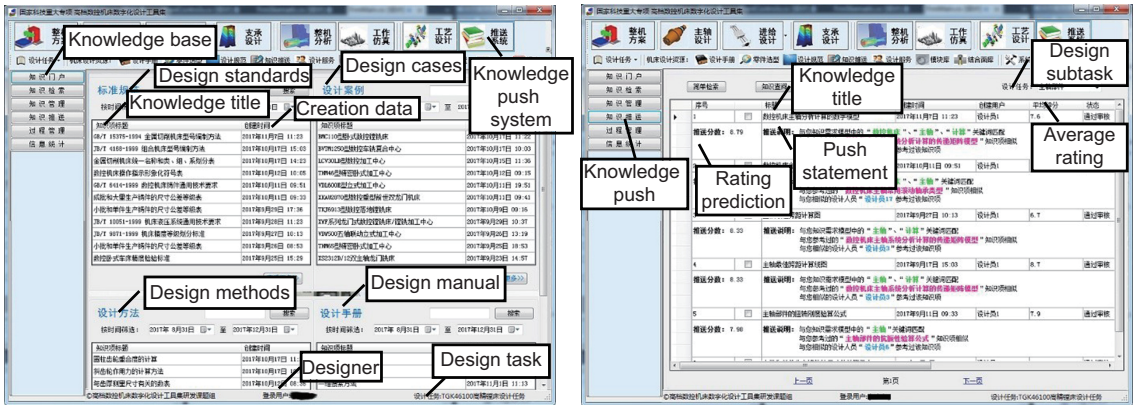
$$\frac{\partial E}{\partial \boldsymbol{W}}=(\boldsymbol{r}-\boldsymbol{A})\exp[-(\|\boldsymbol{s}-\boldsymbol{C}\|\cdot\boldsymbol{b})^2]. \quad (13)$$

After finishing the training work in MCR, we can obtain $\boldsymbol{C}$, $\boldsymbol{b}$, and $\boldsymbol{W}$ to characterize the performance of knowledge matching, which will be detailed in Section 4.

4 Experiments

4.1 Data

We developed a CNC machine tool design knowledge push system (Fig. 7a), where text boxes explain the main components of the interface. We collected 3852 pieces of design knowledge related to CNC machine tools and uploaded these to the web, as shown in the Chinese website in Fig. 7b. As is typical in complex mechanical equipment, the design of a new CNC machine tool was decomposed into many parts and components. In this study, the design of guides by CNC machine tools was used in the experiments. There were 35 design cases and 86 pieces of



(a)

(b)

Fig. 7 Knowledge push system for computer numerical control machine tools: (a) platform interface; (b) knowledge base

design knowledge in the knowledge base.

The original training set contained 35 training samples and the total number of pieces of corresponding knowledge was 86. We used the original training set called "T35" in the experiments, which was constructed in detail in Zhang SY et al. (2018). Each training sample had a case feature vector **pro** and the corresponding design knowledge. A design case library $\boldsymbol{\chi}$ with 34 case features related to the design of guides is shown in Table A1 in the appendix. The Boolean matrix $\boldsymbol{A}$ was with the size of 35×86. In addition, the training sets were augmented by the methods mentioned in Section 3.1. The F70 dataset contained 70 training samples augmented with two oscillators (Fig. 4). The F181 dataset contained 181 training samples augmented by the revision method (Section 3.1.2). The F251 dataset contained 251 training samples by adding F70 and F181. F70, F181, and F251 were all the datasets generated without the original dataset T35. The difference between the training and testing datasets was ensured.

4.2 Methods

The goal of the experiments is to characterize the performance of the subroutines by our approach. The two subroutines of interest are the augmented training set (Section 3.1) and knowledge matching (Section 3.2). Zhang SY et al. (2018) focused only on the evaluation of the results of knowledge matching and the feasibility, which limits the performance of knowledge matching compared with different approaches. In this study, we evaluated different approaches for knowledge matching to determine those approaches that yield state-of-the-art performances and to evaluate the areas requiring improvement in the future. Table 2 summarizes the datasets and approaches explored in the experiments.

We conducted four independent experiments for four different datasets. Each experiment comprised 10 separate trials. For each trial, four different approaches were used separately for the training and testing samples from the appropriate dataset. Zhang SY et al. (2018) evaluated only the performance with dataset 1 and approach 1 in Table 2. We did not allow the testing samples to be involved during the training process, to avoid overfitting. The implementation was developed in MATLAB (R2018a), and all experiments were performed on an Intel Core i5-6400 CPU with 16 GB RAM.

4.3 Evaluation metrics

The metrics for evaluating the knowledge matching performance include the accuracy of matching results and the response time of the approach. For the accuracy of matching results, we employed evaluation of binary classifiers, because this is a well-established technique in computer science. We classified the sets of designers' knowledge demands and the actual knowledge push, as shown in Table 3.

Four metrics were used to evaluate the accuracy of knowledge matching, i.e., Acc, Precision, Recall, and F_{score} (van Rijsbergen, 1979). Acc is the proportion of true pushing results (true positives and true negatives) among the total number of pieces of

Table 3 A binary classification of knowledge push

Total number of pieces of knowledge (N)		Knowledge demand	
		Need (KN)	No need
Knowledge push	Push (KP)	TP	FP
	No push	FN	TN

Assuming the sets of the whole knowledge, the knowledge that designers need, and the pushed knowledge are X, Y, and Z, respectively, we have $N=|X|$, KN$=|Y|$, KP$=|Z|$, TP$=|Y \cap Z|$, FN$=|Y \cap \complement_X Z|$, FP$=|Z \cap \complement_X Y|$, TN$=|\complement_X Z \cap \complement_X Y|$, KP=TP+FP, KN=TP+FN, and N=TP+FP+TP+FN

Table 2 Description of datasets and approaches for experiments

No.	Dataset		Approach	
	Training	Testing	Name	Description
1	T35[a]	T35[a]	Applicable probability matching	Refer to Zhang SY et al. (2018)
2	F70	T35[b]	Back propagation neural network	Three layers with the same size of 34-10-86
3	F181	T35[b]	Radial basis function neural network	Traditional radial basis function neural network without normalized operation
4	F251	T35[b]	Multi-classification radial basis function neural network	Refer to Section 3.2

[a] Randomly selecting 30 training samples and the rest five samples for testing; [b] randomly selecting 30 testing samples

design knowledge, expressed as

$$\mathrm{Acc}=\frac{\mathrm{TP}+\mathrm{TN}}{N}. \tag{14}$$

Precision is the ratio of pushing knowledge that is actually relevant to the design process, expressed as

$$\mathrm{Precision}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}. \tag{15}$$

Recall is the ratio of pushing knowledge that can be located, expressed as

$$\mathrm{Recall}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \tag{16}$$

Increasing the number of pieces of pushing knowledge tends to reduce Precision and increase Recall. F_{score} is used to achieve a trade-off between Precision and Recall by assigning equal weights to them, expressed as

$$F_{\mathrm{score}}=\frac{2\cdot\mathrm{Precision}\cdot\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}. \tag{17}$$

4.4 Results and discussion

The evaluation results of different matching approaches for the four different datasets (Table 2) are provided in Fig. 8. In this study, two key modifications, i.e., an augmented training set and an MCR, were proposed to improve the performance of knowledge matching. AP, BP, and RBF are traditional approaches in knowledge matching. We evaluated the performance of knowledge matching in terms of response time (Figs. 8a–8d) and accuracy (Figs. 8e–8t).

Response time is a metric for the speed of the knowledge push system. As shown in Figs. 8a–8d, with the number of training samples increasing in the datasets, the response time is almost invariable except for that of BP. When training BP, the larger the number of input samples, the longer the training time. In Zhang SY et al. (2018), AP needs to match the design knowledge with the design content one by one, and the response time is relevant to the number of pieces of design knowledge. Thus, AP and BP are not suitable for knowledge matching with the increase of the number of pieces of design knowledge or training samples. RBF performs the best (most response time$<$0.1 s) among these three traditional approaches. There is only one hidden layer in RBF, and the local approximation can reduce the time of calculation and network training. Thus, the proposed MCR is based on RBF with a short response time. The input of MCR is a matrix with the size of $M\times n$. The output of MCR is N neurons, which can match the knowledge once. This mode of input and output can extremely shorten the required time.

There are four metrics for evaluating the accuracy: Acc, Precision, Recall, and F_{score}. In T35-T35, the four approaches perform similarly and poorly, as shown in Figs. 8e, 8i, 8m, and 8q. The values of Acc, Precision, Recall, and F_{score} are approximately 60%, 40%, 45%, and 45%, respectively, which need to be improved. The reason is that the 35 training samples are too few to correspond to the 34 case features. The few training samples cannot match the accurate design knowledge, so we proposed an approach called "augmented training set" in Section 3.1. Comparing Figs. 8e–8h, 8i–8l, 8m–8p, and 8q–8t, we found that the augmented training set works in the evaluation of accuracy, except for RBF. It is interesting that the values of the RBF method in Figs. 8e–8t are almost unchanged, unlike other approaches that perform better than T35-T35. The proposed method is not useful in improving the performance of RBF. For oscillating the feature weight s (Section 3.1.1), the local approximation cannot set up the new local center compared with the original training set. The accuracy of F70-T35 is similar to that of T35-T35, as shown by the blue dotted lines with diamonds in Figs. 8f, 8j, 8n, and 8r. To revise the case feature key (Section 3.1.2), RBF transfers the nonlinear problem in the low-dimensional space to a linear problem in the high-dimensional space. Another reason for the results of these two augmentation methods is that the training data is widely dispersed and can affect the error function in each dimension while training the neural network. AP and BP can perform better with more training samples. As such, to make up the shortcomings in RBF and keep the advantages in response time, we changed and enhanced RBF to MCR in Section 3.2. The "nprod" and "comp" components (Fig. 6b) can improve the accuracy and can be applied to the key characteristics in knowledge matching. The green dotted lines in Fig. 8 show that MCR is the most advanced in terms of both accuracy and response time among these four approaches.

When comparing the two different augmentation methods, we found that generally revision method is better than oscillation method in

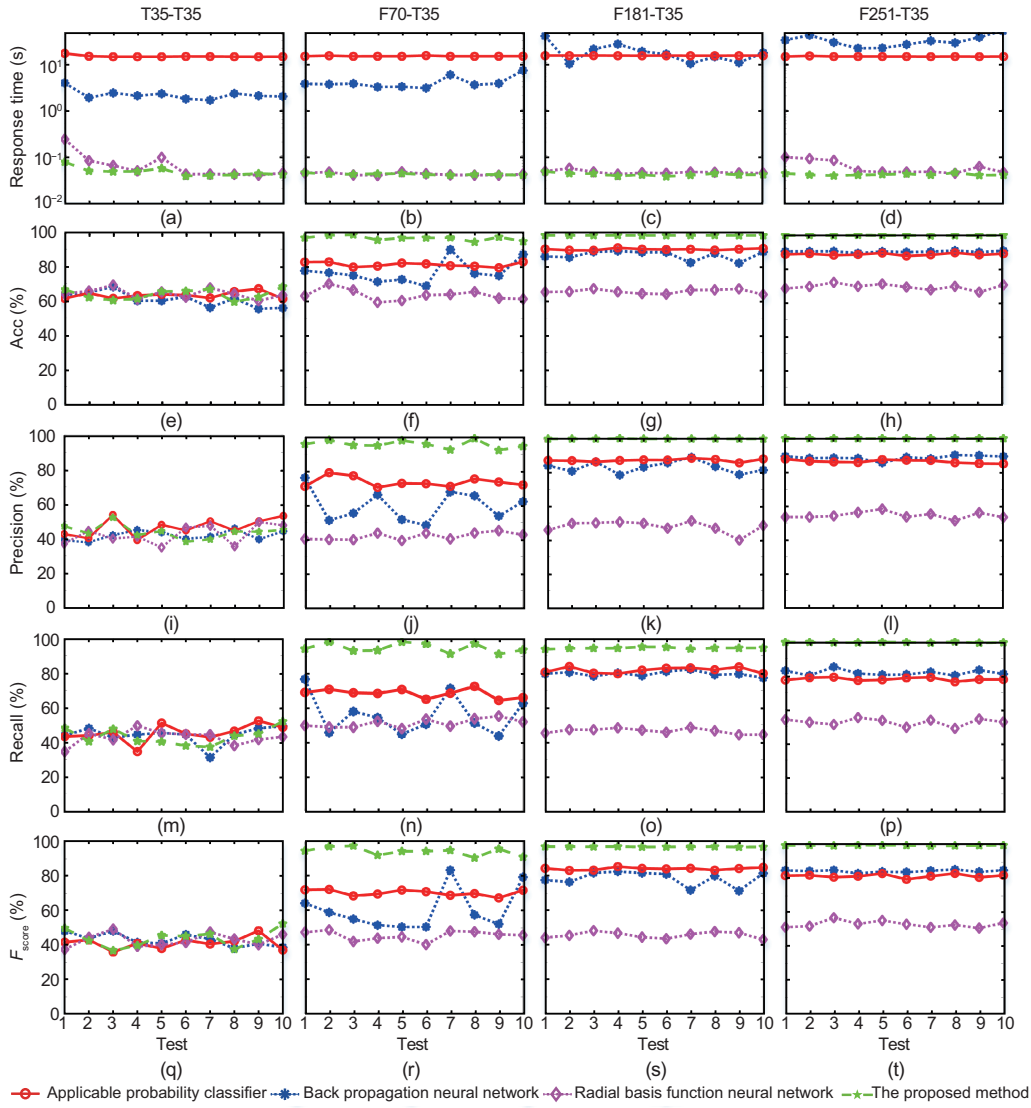


Fig. 8 Evaluations for different matching approaches with different datasets: (a)–(d) response time; (e)–(h): Acc; (i)–(l): Precision; (m)–(p): Recall; (q)–(t): F_{score}

References to color refer to the online version of this figure

accuracy. The oscillation method just changes the feature weight s from a single value to interval values, and the weakness of the original training set cannot be improved, such as the signal design type and the design content. When combining these two augmentation methods, the results are not significantly improved. We found that the values in Figs. 8g and 8h, 8k and 8l, 8o and 8p, and 8s and 8t are similar, respectively. One reason is that the oscillation method cannot make up the weakness of F181, and the other reason is that the optimization of augmentation methods has been fully developed. For these four metrics, we can conclude that the limited training set in practical operations can be augmented for high performance in knowledge matching. This conclusion is especially applicable to AP in Zhang SY et al. (2018) and MCR in the present work.

AP and BP both have a long response time, and another shortcoming of BP is the uncertainty of hidden layers in the neural network. RBF is weak in accuracy, and for this reason, we developed an MCR based on RBF by increasing "nprod" and "comp" (Fig. 6b). Experimental results indicated that MCR has a higher accuracy and a shorter response time than those of AP in Zhang SY et al. (2018).

Specifically, we compared the detailed knowledge matching results in Fig. 9. Acc_case is the value of Acc in every design case and Acc_knowledge is the value of Acc in each design knowledge during the test experiments, expressed as

$$\text{Acc\_case}_i = \frac{|r(i,:) \cap A(i,:)|}{N}, \tag{18}$$

$$\text{Acc\_knowledge}_j = \frac{|r(:,j) \cap A(:,j)|}{N}, \tag{19}$$

where $\boldsymbol{r}$ denotes the matrix of the results in the tests with the size of $m \times N$, $\boldsymbol{A}$ the ground truth, and m the number of design cases in the tests.

Overall, the matching results of AP in Fig. 9a are of low accuracy for few training samples. The knowledge matching results are partially missing with the unknown design tasks, such as K27 and K34–K48 (K+number represents the different numbered design knowledge). The knowledge that designers need may not be pushed, which is detrimental to the accuracy of the knowledge push system. We found that Acc cannot be the only metric to evaluate the accuracy, as shown by the matching results in case30 in Fig. 9a. In Fig. 9a, AP matches only 16 pieces of knowledge and the actual number of pieces of required knowledge is 29 (KP = 16 and KN = 29), but $\text{Acc\_case}_{30} \approx 70\%$ is not low. Precision, Recall, and F_{score} are necessary in the evaluation, as all being approximately equal to 40% is a disadvantage in knowledge push. Furthermore, the matching results of MCR in Fig. 9b are better than those of AP, except for K29–K34. We found that the few samples with K29–K34 in the training set affect the matching results in tests. The remaining knowledge is matched well, as shown in Fig. 9b.

For the experimental results, we can draw the following conclusions: (1) Augmenting the training set can improve the performance of knowledge matching, especially when the training samples are limited in practical operations; (2) MCR is better than AP (Zhang SY et al., 2018), and also better than other traditional matching approaches, such as BP and RBF; (3) Various and numerous training samples are beneficial to the matching performance.

5 Conclusions

In this study, we aimed to improve the performance of knowledge matching in a knowledge push system, i.e., increasing the accuracy and shortening

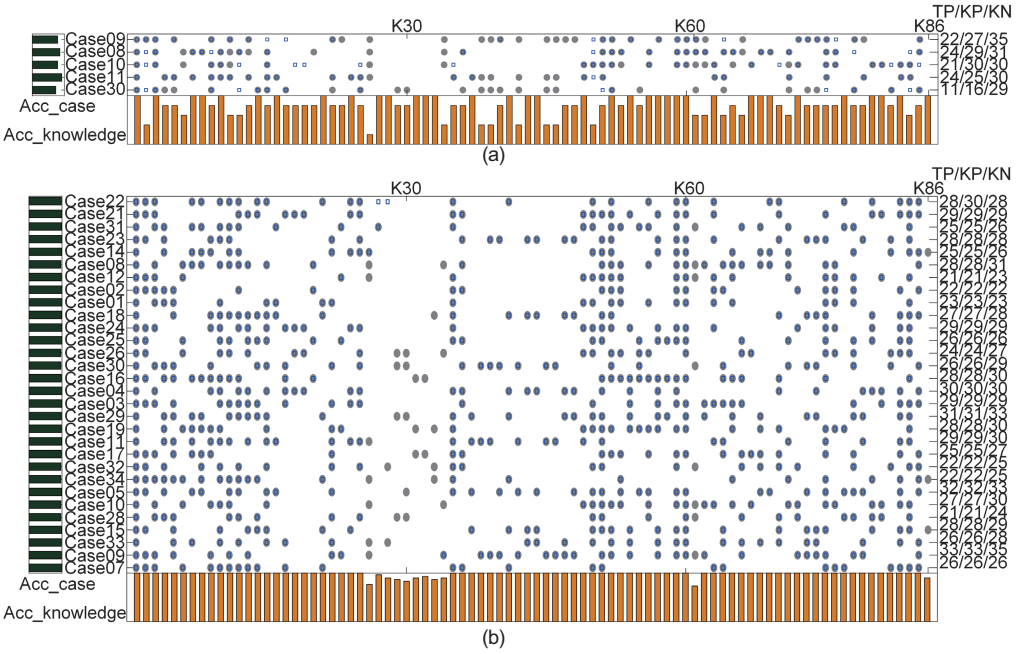


Fig. 9 Detailed knowledge matching results: (a) AP in T35-T35; (b) MCR in F181-T35

Left of the subgraph is the accuracy of cases, the bottom is the accuracy of knowledge, the right represents the values of TP, KP, and KN (Table 3), and the middle represents the matching results of knowledge demand (gray filled circle) and knowledge push (blue hollow square). References to color refer to the online version of this figure

the response time. We improved our previous study by augmenting the training set and changing the algorithm structure of knowledge matching. Our contributions are as follows:

1. We proposed two methods to augment the training set, i.e., oscillating the feature weight and revising the case feature in the case feature vectors. Augmenting the training set is positive when using artificial intelligence algorithms in industrial and production engineering, especially when the training set is limited in practical operations.

2. We proposed a new knowledge matching approach called MCR based on the traditional RBF. We enhanced the neural network in two parts, i.e., "nprod" and "comp," to adapt to the multi-classification problem of knowledge matching. Then, a supervised learning algorithm based on gradient descent was used for training MCR.

3. We compared the performances of different approaches in different datasets about the design of guides. The original dataset has been constructed in Zhang SY et al. (2018). Experimental results indicated that MCR outperforms the existing approaches in knowledge matching. Our proposed approach satisfies the requirements for high accuracy and rapid response.

The proposed knowledge matching approach can improve the performance of a knowledge push system, but it needs to be tested in the design of different products. In the future, we will focus on the universality of our knowledge push technology application, and investigate the improvement of personalization.

Contributors

Shu-you ZHANG guided the research. Ye GU designed the research and drafted the manuscript. Shu-you ZHANG and Guo-dong YI helped organize and revise the manuscript. Zi-li WANG finalized the paper.

Compliance with ethics guidelines

Shu-you ZHANG, Ye GU, Guo-dong YI, and Zi-li WANG declare that they have no conflict of interest.

References

Bahramian Z, Abbaspour RA, Claramunt C, 2017. A cold start context-aware recommender system for tour planning using artificial neural network and case based reasoning. *Mob Inform Syst*, 2017:9364903. https://doi.org/10.1155/2017/9364903

Chen S, Yang ZY, Sun LY, et al., 2015. Research on design knowledge analytical method during sketching—combining acoustic energy feature and creative segment theory. *J Zhejiang Univ Eng Sci*, 49(11):2073-2082 (in Chinese).

Devi MKK, Samy RT, Kumar SV, et al., 2010. Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative recommender systems. Proc IEEE Int Conf on Computational Intelligence and Computing Research, p.1-4. https://doi.org/10.1109/ICCIC.2010.5705777

Dong SY, Xu JX, Wang KQ, et al., 2013. Active push model of manufacturing process knowledge in CAD platform based on immune process. *Comput Integr Manuf Syst*, 19(7):1520-1531 (in Chinese). https://doi.org/10.13196/j.cims.2013.07.82.dongsy.016

Fan ZP, Feng Y, Sun YH, et al., 2005. A framework on compound knowledge push system oriented to organizational employees. Proc 1st Int Workshop on Internet and Network Economics, p.622-630. https://doi.org/10.1007/11600930_62

Feng YX, Zhang SY, Gao YC, et al., 2016. Intelligent push method of CNC design knowledge based on feature semantic analysis. *Comput Integr Manuf Syst*, 22(1):189-201 (in Chinese). http://doi.org/10.13196/j.cims.2016.01.018

Gabrani G, Sabharwal S, Singh VK, 2017. Artificial intelligence based recommender systems: a survey. Proc 1st Int Conf on Advances in Computing and Data Sciences, p.50-59. https://doi.org/10.1007/978-981-10-5427-3_6

Guo Y, Yin CX, Li MF, et al., 2018. Mobile e-commerce recommendation system based on multi-source information fusion for sustainable e-business. *Sustainability*, 10(1):147. https://doi.org/10.3390/su10010147

Gupta A, Tripathy BK, 2014. A generic hybrid recommender system based on neural networks. Proc IEEE Int Advance Computing Conf, p.1248-1252. https://doi.org/10.1109/IAdCC.2014.6779506

Ji X, Gu XJ, Dai F, et al., 2013. Technology for product design knowledge push based on ontology and rough sets. *Comput Integr Manuf Syst*, 19(1):7-20 (in Chinese). https://doi.org/10.13196/j.cims.2013.01.9.jix.008

Jiang H, Yin P, Guo L, et al., 2017. Knowledge push based on design flow and user capacity model. Proc MATEC Web Conf, Article 12.

Karayiannis NB, 1999. Reformulated radial basis neural networks trained by gradient descent. *IEEE Trans Neur Netw*, 10(3):657-671. https://doi.org/10.1109/72.761725

Le CY, Dai F, Ji X, et al., 2010. Domain knowledge actively pushing system driven by process. *Comput Integr Manuf Syst*, 16(12):2720-2727 (in Chinese). https://doi.org/10.13196/j.cims.2010.12.178.yuechy.018

Li XR, Yu SH, Chu JJ, et al., 2017. Double push strategy of knowledge for product design based on complex network theory. *Discr Dynam Nat Soc*, Article 2 078 626. https://doi.org/10.1155/2017/2078626

Liang Y, Zhang S, Liu X, et al., 2015. Product design knowledge dynamic push technology based on variable-weight layered spreading activation model. *Comput Integr Manuf Syst*, 21(12):3107-3118 (in Chinese). https://doi.org/10.13196/j.cims.2015.12.002

Liu HM, Wang HQ, Li X, 2009. A study on data normalization for target recognition based on RPROP algorithm.

Mod Radar, 31(5):55-60. https://doi.org/10.3969/j.issn.1004-7859.2009.05.014

Liu TY, Wang HF, He Y, 2016. Intelligent knowledge recommending approach for new product development based on workflow context matching. *Concurr Eng*, 24(4):318-329. https://doi.org/10.1177/1063293X16640319

Paradarami TK, Bastian ND, Wightman JL, 2017. A hybrid recommender system using artificial neural networks. *Expert Syst Appl*, 83:300-313. https://doi.org/10.1016/j.eswa.2017.04.046

Pushpa CN, Ashvini P, Thriveni J, et al., 2013. Web page recommendations using radial basis neural network technique. Proc 8th Int Conf on Industrial and Information Systems, p.501-506. https://doi.org/10.1109/ICIInfS.2013.6732035

Schreiber AT, Schreiber G, Akkermans H, et al., 2000. Knowledge engineering and management: the common KADS methodology. MIT Press, Cambridge, MA, USA.

Sunhem W, Pasupa K, 2016. An approach to face shape classification for hairstyle recommendation. Proc 8th Int Conf on Advanced Computational Intelligence, p.390-394. https://doi.org/10.1109/ICACI.2016.7449857

Twardowski B, 2016. Modelling contextual information in session-aware recommender systems with neural networks. Proc 10th ACM Conf on Recommender Systems, p.273-276. https://doi.org/10.1145/2959100.2959162

van Rijsbergen CJ, 1979. Information Retrieval (2nd Ed.). Butterworth, London, UK.

Wang ZS, Tian L, Wu YH, et al., 2016. Personalized knowledge push system based on design intent and user interest. *Proc Inst Mech Eng Part C*, 230(11):1757-1772. https://doi.org/10.1177/0954406215584395

Wu H, Zhang ZX, Yue K, et al., 2018. Dual-regularized matrix factorization with deep neural networks for recommender systems. *Knowl-Based Syst*, 145:46-58. https://doi.org/10.1016/j.knosys.2018.01.003

Wu LJ, Gou BC, Wen CG, 2018. Research on knowledge-push driven by workflow and knowledge points. *Comput Eng Appl*, 54(4):231-236 (in Chinese). https://doi.org/10.3778/j.issn.1002-8331.1609-0014

Xiao Y, Lou CQ, Liu G, 2010. Personalized knowledge push service based on semantic web. Int Conf on E-Business and E-Government, p.1872-1875. https://doi.org/10.1109/ICEE.2010.473

Xu YH, Yin GF, Nie Y, et al., 2013. Research on an active knowledge push service based on collaborative intent capture. *J Netw Comput Appl*, 36(6):1418-1430. https://doi.org/10.1016/j.jnca.2013.04.010

Xue HJ, Dai XY, Zhang JB, et al., 2017. Deep matrix factorization models for recommender systems. Proc 26th Int Joint Conf on Artificial Intelligence, p.3203-3209. https://doi.org/10.24963/ijcai.2017/447

Yan Y, Yang N, Hao J, et al., 2016. A context modeling method of knowledge recommendation for designers. Proc Int Conf on Information System and Artificial Intelligence, p.492-496. https://doi.org/10.1109/ISAI.2016.0111

Yang XH, Huang JF, Wang JW, et al., 2007. Estimation of vegetation biophysical parameters by remote sensing using radial basis function neural network. *J Zhejiang Univ-Sci A (Appl Phys Eng)*, 8(6):883-859. https://doi.org/10.1631/jzus.2007.A0883

Zhang C, Zhou GH, Bai QD, et al., 2018. HEKM: a high-end equipment knowledge management system for supporting knowledge-driven decision-making in new product development. Proc ASME Int Design Engineering Technical Conf and Computers and Information in Engineering Conf, Article V0 1BT 02A 014. https://doi.org/10.1115/DETC2018-85151

Zhang FP, Li L, 2016. Research on knowledge push method for business process based on multidimensional hierarchical context model. Proc IEEE Int Conf on Industrial Engineering and Engineering Management, p.656-660. https://doi.org/10.1109/IEEM.2016.7797957

Zhang FP, Li L, 2017. Research on knowledge push method for business process based on multidimensional hierarchical context model. *J Comput-Aided Des Comput Graph*, 29(4):751-758 (in Chinese). https://doi.org/10.3969/j.issn.1003-9775.2017.04.021

Zhang K, Zhao W, Wang J, et al., 2019. Knowledge push technology based on quality function knowledge deployment. *Proc Inst Mech Eng Part C*, 233(4):1119-1138. https://doi.org/10.1177/0954406218768843

Zhang LL, Nie GL, Zhang YJ, et al., 2009. A way to implement intelligent knowledge push in knowledge management system. Proc Int Joint Conf on Computational Sciences and Optimization, 1:746-749. https://doi.org/10.1109/CSO.2009.102

Zhang SY, Gu Y, Liu X, et al., 2018. A knowledge push technology based on applicable probability matching and multidimensional context driving. *Front Inform Technol Electron Eng*, 19(2):235-245. https://doi.org/10.1631/FITEE.1700763

Zhang SY, Gu Y, Yi GD, 2019. A hybrid knowledge push method based on trust-aware and item-cluster oriented to product design. *New Gener Comput*, 37:339-357. https://doi.org/10.1007/s00354-019-00053-3

Zuo Y, Zeng J, Gong M, et al., 2016. Tag-aware recommender systems based on deep neural networks. *Neurocomputing*, 204:51-60. https://doi.org/10.1016/j.neucom.2015.10.134

Appendix: Case feature vector in the design case library (part)

Table A1 Case feature vector in the design case library (part)

Case feature	$\mathbf{pro}_1$	$\mathbf{pro}_2$	$\mathbf{pro}_3$	$\mathbf{pro}_4$	$\mathbf{pro}_5$	$\mathbf{pro}_6$	$\mathbf{pro}_7$	$\mathbf{pro}_8$	$\mathbf{pro}_9$	$\mathbf{pro}_{10}$	$\mathbf{pro}_{11}$	$\mathbf{pro}_{12}$	$\mathbf{pro}_{13}$	$\mathbf{pro}_{14}$	...	$\mathbf{pro}_{35}$
Horizontal lathe	0.5163	0.3483	0	0	0	0	0	0	0	0	0	0	0.5518	0	...	0
Boring	0	0	0	0	0.8222	0.4759	0	0	0	0	0	0	0	0	...	0
CNC	0	0	0	0	0	0	0	0	0.6617	0	0	0.7611	0	0	...	0
Grinding	0	0	0	0	0	0	0	0	0	0.3116	0	0	0	0.1340	...	0
Vertical lathe	0	0	0.3733	0.4694	0	0	0	0	0	0	0	0	0	0	...	0
Drilling	0	0	0	0	0	0	0	0	0	0	0.8861	0	0	0	...	0
Milling	0	0	0	0	0	0	0	0.7628	0.0193	0	0	0	0	0	...	0
Planer	0	0	0	0	0	0	0.8377	0	0	0	0	0	0	0	...	0.3058
Precision	0.6172	0	0	0.2221	0.0055	0.1467	0	0.7226	0.0536	0.6422	0	0.3612	0	0.6060	...	0
Function	0.4198	0	0.7978	0.5891	0	0	0	0.9738	0.1108	0	0	0.7769	0	0.4495	...	0.5787
Sliding	0.8143	0.7434	0.1881	0.8008	0.4841	0	0.9211	0.0927	0.8225	0.6607	0.9885	0.5732	0.4298	0.4748	...	0.7153
Rectilinear	0.7061	0.3665	0.1180	0.9497	0.0709	0.6599	0.1967	0.8596	0.3660	0.2664	0.0438	0	0.0434	0.0363	...	0
Swing	0	0	0	0	0	0	0	0	0	0	0	0.7914	0	0.4809	...	0.2921
Multiple	0	0.9614	0	0	0.1975	0	0.5370	0.5796	0.6927	0	0.7448	0	0	0	...	0.1719
Cast iron	0.8423	0.6050	0.3923	0.5305	0.7392	0	0.4435	0.6564	0	0.1022	0.0438	0	0.0434	0.0363	...	0.3611
Steel	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
Plastic	0	0	0	0	0	0	0	0	0	0.5468	0	0.9898	0	0	...	0
Open	0.8052	0	0	0	0.0788	0	0.4600	0	0.3844	0.7846	0.7448	0	0.1355	0	...	0.5865
Closed	0	0	0	0.9620	0	0	0	0	0	0	0	0	0	0.3717	...	0
Inlaid	0	0.3165	0.3045	0	0	0.4838	0	0.1818	0	0	0	0	0	0	...	0
Flat	0	0.2144	0	0	0	0.7553	0	0.9332	0	0	0	0	0	0	...	0
Elbow	0	0	0.5281	0	0	0	0	0	0	0	0	0	0	0	...	0
Main motion	0	0.1619	0.6942	0.7553	0.6758	0	0.7421	0.1099	0.0312	0	0	0.677	0.0076	0.3116	...	0.6189
Feed	0.8446	0	0	0	0	0.5310	0	0	0.0517	0.9123	0.4194	0	0	0	...	0
Crawl	0.5432	0	0	0	0.2029	0	0	0	0.9958	0	0.3653	0	0	0.6422	...	0
Triangular	0	0	0	0	0	0	0.9344	0	0	0	0	0	0	0	...	0
Rectangular	0	0	0	0	0	0.5793	0	0	0.3089	0	0	0	0.3581	0	...	0
Dovetail	0.9168	0.8320	0.6760	0	0.3070	0	0	0.0533	0	0.3294	0.1129	0	0.7774	0	...	0.1318
Circular	0	0	0	0.8537	0	0	0	0	0	0	0	0.4841	0	0.6607	...	0
Gap	0.6578	0.1619	0.3321	0.1438	0.0914	0.4812	0	0.1897	0.4941	0.9587	0.7075	0.3501	0.2103	0.4495	...	0.0506
Unloading	0.4803	0.3602	0.2048	0.8095	0.9955	0.1978	0.9952	0.4631	0.2125	0.7038	0.5146	0.9056	0.5973	0.4748	...	0.9741
Processing	0.2578	0.3580	0.1760	0.0635	0.5285	0.1338	0.4471	0.4444	0.6892	0.6747	0.5353	0.2616	0.2890	0.4809	...	0.6058
Mechanical	0	0.6258	0	0.8433	0.8984	0	0.4848	0.0338	0.7419	0.8999	0.8527	0.0193	0.6426	0.0363	...	0.4521
Hydraulic	0.0842	0	0.6160	0	0.0071	0.5781	0.5064	0.9283	0	0	0.6039	0.5546	0	0.3717	...	0