Information Systems Frontiers (2021) 23:515–530
https://doi.org/10.1007/s10796-019-09973-3

An Efficient Classification of Fuzzy XML Documents Based on Kernel ELM

Zhen Zhao[1] · Zongmin Ma[2,3] · Li Yan[2]

Published online: 5 December 2019

Abstract
Data classification for distributed and heterogeneous XML data sources is always an open challenge. A considerable number of algorithms for classification of XML documents have been proposed in the literature. Yet, the existing approaches fall short in ability to classify the fuzzy XML documents. In this paper, we provide a KPCA-KELM classification framework for the fuzzy XML documents based on Kernel Extreme Learning Machine (KELM). Firstly, we propose a novel fuzzy XML document tree model to represent fuzzy XML documents. Secondly, we employ an effective vector space model to represent the semantic structure of fuzzy XML documents based on the proposed fuzzy XML document tree model. Thirdly, we classify the fuzzy XML document using KELM after feature extraction using Kernel Principal Component Analysis (KPCA). The corresponding experimental results demonstrate that our proposed KPCA-KELM approach shortens the training time while maintaining the same level of accuracy as the state-of-the-art baseline models.

Keywords Data classification · Fuzzy XML · Feature extraction · Kernel extreme learning machine (KELM)

1 Introduction

With the development of the Internet, XML (Extensible Markup Language) has become a de-facto standard for representing a large number of rapidly increasing Web data in numerous applications. Because of heterogeneous and distributed data source is utilized in a lot of applications, XML data integration becomes more and more imperative (Thomo and Venkatesh 2008; Guha et al. 2006). XML document classification plays a critical role in XML data integration. It is logical that there is a growing demand in the research of XML document classification (Brzezinski and Piernik 2015; Zhao et al. 2011; Thasleena and Varghese 2015).

The XML document classification is similar to plain text classification to a certain extent. The difference between them is that pure text classification only focuses on the semantic level, while XML document classification needs to consider both the structure and semantics of the XML document. Various approaches have been designed to perform XML document classification in (Zhao et al. 2011; Dalamagas et al. 2006; Tekli and Chbeir 2012; Maguitman et al. 2005; Tekli et al. 2015).The majority of the approaches exploit the direct comparison of the structure and semantics of the given paired XML documents, such as tree edit distance (Dalamagas et al. 2006; Tekli and Chbeir 2012; Maguitman et al. 2005; Tekli et al. 2015). Some of the approaches also exploit machine learning strategies (Zhao et al. 2011; Zhang et al. 2012) to solve this problem. Zhao et al. reported that XML documents have to be transformed into a specific representation model, which is then taken as an input of Extreme Learning Machine (ELM) (Zhao et al. 2011). Ribeiro and Härder introduced an approach of entity identification in XML documents based on approximate joins (Ribeiro and Härder 2006). Zhang et al. presented an approach of computing the similarity between XML documents, which make reference to the adjacency matrix of graph theory (Zhang et al. 2012). The authors introduced XML matching approaches and a template, called XML Matcher Template, which describes the main components of an XML Matcher in (Agreste et al. 2014).

XML documents which are previously classified are often deterministic. However, in fact, fuzzy information often emerges in some practical applications. Some data are

✉ Zongmin Ma
zongminma@nuaa.edu.cn

[1] College of Information Science and Technology, Bohai University, Jinzhou 121013, Liaoning, China

[2] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, Jiangsu, China

[3] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023, Jiangsu, China

inherently vague rather than definite since their values are subjective in real-world applications. Fuzzy sets theory which proposed by Zadeh has been widely used in numerous applications (Negoita et al. 1978). Fuzzy data are modeled in the XML document in (Nierrman and Jagadish 2002; Gaurav and Alhajj 2006; Oliboni and Pozzani 2008) and approaches representing and processing fuzzy information based on the XML data are proposed in (Turowski and Weng 2002; Abiteboul et al. 2006).

With the increasing of fuzzy XML data on the Web, it is necessary to classify fuzzy XML documents for further integrating multiple similar fuzzy XML documents into a single one. Unfortunately, to our best knowledge, there are not many reports discussing the fuzzy XML documents classification. Although uncertain XML document classification has been investigated (Zhao et al. 2016), the proposed approaches need to consume a large amount of time to enumerate all possible instances. Based on the above analysis, aiming at developing an effective approach to classify fuzzy XML documents, we devise a new fuzzy XML document tree model to represent semantic and structure of the fuzzy XML documents, and then we propose an integrated machine learning approach (KPCA-KELM) to classify fuzzy XML documents effectively.

In this paper, we concentrate on the classification of heterogeneous fuzzy XML documents which are collected from different data sources. The main contributions of the proposed work are as follows.

- We take a first step in the construction of a new fuzzy XML document tree model (FXDTM), which makes it easier to describe fuzzy data and capture the feature information in fuzzy XML documents. Based on the Structured Link Vector Model (SLVM) (Yang and Chen 2002), a Modified Structured Vector Space Model (MS-VSM) is employed in (Zhao et al. 2017) to represent fuzzy XML documents, which not only represents structural and semantic information of fuzzy XML documents, but also makes the feature vector carry fuzzy information. Considering that the accuracy and validity of feature extraction can be improved by using the FXDTM model to represent the key features of fuzzy XML document, in this paper, the FXDTM model is used as an intermediary instead of directly converting fuzzy XML document into the MS-VSM model as in (Zhao et al. 2017). The raw data coming from this MS-VSM model are taken as the input of classifiers.
- We propose an integrate machine learning approach (KPCA-KELM) to classify fuzzy XML documents. At first, to reduce the complexity of computation and exploit hidden information, we perform features extracting of the input raw data using Kernel Principal Component Analysis (KPAC) (Schölkopf et al. 1998; Iosifidis et al. 2015) approach. Different feature extraction methods can result in different classifications. The ELM is employed as feature extractor in (Zhao et al. 2017). Considering that the KPCA is very suitable for extracting the interesting non-linear features from high-dimensional space data, in this paper, we use the KPCA to capture the non-linear features of fuzzy XML documents to extract those decisive features. Secondly, due to Extreme Learning Machine (ELM) (Huang et al. 2006; Huang and Chen 2007; Huang et al. 2016; Tang et al. 2016) has a good classification accuracy and faster learning speed, we propose an effective algorithm based on Kernel Extreme Learning Machine (KELM) to achieve the classification of fuzzy XML document. In this way, we can classify the fuzzy XML documents collected from diverse data sources.

The remainder of this paper is organized as follows. Section 2 introduces the related works, including the fundamental concepts and theories of KPCA and KELM after a presentation of fuzzy XML documents. Section 3 describes the proposed Fuzzy XML Document Tree Model (FXDTM) and Modified Structured Vector Space Model (MS-VSM) of fuzzy XML documents. The KPCA-KELM framework based on KPCA and KELM is proposed in Section 4. Experimental evaluations are given in section 5. Finally, the conclusion is drawn in Section 6.

2 Related Works

To facilitate the understanding of the proposed approach, this section briefly reviews the related fundamental of concepts/theories of the fuzzy XML documents, KPCA and KELM. Though ELM unifies regression and classification tasks, we only focus on the classification in the following parts.

2.1 Fuzzy XML Document

A fuzzy XML document is a set composed of elements and attributes, linked together via the containment relation. To represent fuzzy information in fuzzy XML documents, a representation model based on “membership degree and possibility distributions” is developed in (Ma and Yan 2007; Yan et al. 2009). In this model, an element may be involved in a membership degree. The membership degree of an element indicates the possibility of being its parent’s child element. The attribute values of elements may be presented as possibility distributions in this representation model. Note that it is possible that some attributes can take multiple (conjunctive) values. In

contrast, some attributes can only take unique (disjunctive) value. That is, we have two kinds of fuzziness in the fuzzy XML document: one is the fuzziness in elements associated with membership degrees; another is the fuzziness in attribute values of elements represented with possibility distributions. The *Type* attribute as a child of element *Dist* is used to indicate the type of possibility distribution, having values of disjunctive or conjunctive. In addition, each *Dist* element has a *Val* element as its child. The *Poss* attribute as child of element *VAL* indicates the membership degrees of a given element.

Here we do not present the detailed definitions of fuzzy XML document representation model and only present an example fragment of fuzzy XML document shown in Fig. 1. One can refer to (Ma and Yan 2007; Yan et al. 2009; Li and Ma 2017) for more details.

In the fuzzy XML document of Fig. 1, *Poss* is adopted together with a fuzzy construct denoted by *Val* to specify the possibility of a given element in the fuzzy XML document. In line 2 of Fig. 1, <Val Poss = 0.9 > states that the membership degree of the given element department being Information Science and Technology is equal to 0.9. Another fuzzy construct called *Dist* can be used to express possibility distribution of attribute values. Since we have two types of possibility distribution, *Type* is adopted to indicate the type of possibility distribution, being disjunctive or conjunctive. Lines 18–22 are

Fig. 1 A fragment of fuzzy XML document

```
1.  <college Cname = "liaoning college">
2.    <Val Poss = 0.9>
3.      <department Dname = "Information Science and Technology">
4.        <teacher Tid = "007100">
5.          <Dist Type = "disjunctive">
6.            <Val Poss = 0.8>
7.              <Tname>Ford George</Tname>
8.              <title>Associate Professor</title>
9.            </Val >
10.           <Val Poss = 0.6>
11.             <Tname>Ford George</Tname>
12.             <title>Professor</title>
13.           </Val >
14.         </Dist>
15.       </teacher>
16.       <student SID = "20130425">
17.         <age>
18.           <Dist Type = "disjunctive">
19.             <Val Poss = 0.8>26</Val>
20.             <Val Poss = 0.9>28</Val>
21.             <Val Poss = 0.8>29</Val>
22.           </Dist>
23.         </age>
24.         <email>
25.           <Dist Type = "conjunctive">
26.             <Val Poss = 0.6>John_Smith@yahoo.com</Val>
27.             <Val Poss = 0.8>John_Smith@hotmail.com</Val>
28.             <Val Poss = 0.5>JSmith@hotmail.com</Val>
29.           </Dist>
30.         </email>
31.       </student>
32.     </department >
33.   </Val>
34. </college>
```

the disjunctive *Dist* construct for the age of the student with SID 20130425. Lines 25–29 are the conjunctive *Dist* construct for the email of the student with SID 20130425.

2.2 Kernel Principal Component Analysis (KPCA)

Kernel Principal Component Analysis (KPCA) (Schölkopf et al. 1998; Iosifidis et al. 2015), which is derived from Principal Component Analysis (PCA), has been widely used for nonlinear feature extraction. To be fluent for reading, let us introduce the essential knowledge of KPCA. More details about KPCA can be found in (Schölkopf et al. 1998; Iosifidis et al. 2015).

As a nonlinear method, KPCA is nothing but the PCA in the feature space associated with a kernel function. KPCA maps all samples from the linear data space to the nonlinear feature space, and the PCA features are extracted in the feature space. To the data space R^d, consider a feature space by a possibly nonlinear map Φ: $R^d \rightarrow F, x \rightarrow \Phi(x)$. The map Φ is defined implicitly with kernel function.

Let us consider a set of N training samples in a d-dimensional space $x_i \in R^d$, $i = 1, \ldots, N$. If the training samples have been mapped into a feature space by a nonlinear function Φ, we may perform PCA in feature space F. We assume that all their mapped samples $\Phi(x_1)$, $\Phi(x_2)$, …, $\Phi(x_N)$ are centered, that is $\sum_{i=1}^{N} \Phi(x_i) = 0$. The correlation matrix of the mapped samples in feature space F can be computed by

$$C = \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathrm{x_i}) \Phi(\mathrm{x_i})^T \tag{1}$$

The extracted features in feature space must be from the set of the eigenvectors of C (Schölkopf et al. 1998). With these extracted features, we can reconstruct the samples with the minimum mean-square error. KPCA seeks to find eigenvalues λ and the associated eigenvectors V satisfying

$$CV = \lambda V \tag{2}$$

The eigenvalue equation can be written as

$$\left(\Phi(x_\mu)^T \cdot CV\right) = \lambda \left(\Phi(x_\mu)^T \cdot V\right) \quad for\ all \quad \mu = 1, \ldots, N \tag{3}$$

And there exists coefficients $\alpha_j, j = 1, \ldots, N$, such that

$$V = \sum_{j=1}^{N} \alpha_j \Phi(x_j) \tag{4}$$

Combining Eqs. (1), (3) and (4), we have

$$\frac{1}{N} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} \left(\Phi(x_\mu)^T \cdot \Phi(x_i) \Phi(x_i)^T \cdot \alpha_j \Phi(x_j) \right) \right) = \lambda \sum_{j=1}^{N} \left(\Phi(x_\mu)^T \cdot \alpha_j \Phi(x_j) \right) \tag{5}$$

Now we define an $N \times N$ kernel matrix K by $K_{ij} = (\Phi(x_i)^T \cdot \Phi(x_j))K_{ij} = (\Phi(x_i)^T \cdot \Phi(x_j))$. Furthermore, the following equation can be derived: $K^2\alpha = \lambda NK\alpha$, where $\alpha = [\alpha_1, \ldots, \alpha_N]$. In fact, the coefficient vector α is the eigenvector corresponding to non-negative eigenvalue of the kernel matrix K. Finding solutions of this equation is equivalent to solve the eigenvalue problem of $K\alpha = N\lambda\alpha$ for nonzero eigenvalues. Then this yields eigenvectors $\alpha^1, \ldots, \alpha^N$ corresponding to eigenvalues $\lambda^1 > \ldots > \lambda^N$.

The dimensionality of the input data is reduced by retaining only the first p most representative eigenvectors associated with the first p largest eigenvalues. That is, the first p nonlinear principal components are used to describe the input data. Note that in KPCA, the number of principal components (PC) p may exceed the input dimensionality d. The maximum number of PCs in KPCA is N.

The corresponding eigenvectors $\alpha^1, \ldots, \alpha^p$ of eigenvalue $\lambda^1, \ldots, \lambda^p$ can be normalized by requiring the corresponding vectors $V^1, \ldots, V^p$ in F be normalized, that is

$$\begin{aligned} \left(\mathbf{V}^k\right)^T \cdot \mathbf{V}^k &= \sum_{j,l=1}^{n} \alpha_j^k \alpha_l^k \left(\Phi^T(\mathrm{x}_j) \cdot \Phi(\mathrm{x}_l)\right) \\ &= \sum_{j,l=1}^{n} \alpha_j^k \alpha_l^k \cdot k(\mathrm{x}_j, \mathrm{x}_l) \\ &= \left(\boldsymbol{\alpha}^k\right)^T \mathbf{K} \boldsymbol{\alpha}^k \\ &= \lambda^k \left(\boldsymbol{\alpha}^k\right)^T \boldsymbol{\alpha}^k \\ &= \lambda^k \left\|\boldsymbol{\alpha}^k\right\|^2 \\ &= 1 \end{aligned} \tag{6}$$

when $\left\|\boldsymbol{\alpha}^k\right\|^2 = \frac{1}{\lambda^k}$, for all k = 1, 2, …, p, $\boldsymbol{V}^k$ is normalized.

Principal components of a test sample y are obtained by projecting Φ(y) onto the eigenvectors $\boldsymbol{V}^k$ in F: $k = 1, \ldots, p$

$$\begin{aligned} &\Phi^T(y)\left[\mathbf{V}_1, \ldots, \mathbf{V}_p\right] \\ &= \left[k(y, \mathrm{x}_1), \ldots, k(y, \mathrm{x}_n)\right]\left[\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^p\right] \in R^{1 \times p} \end{aligned} \tag{7}$$

Here $\boldsymbol{\alpha}^k = \left[\alpha_1^k, \alpha_2^k, \ldots, \alpha_N^k\right]^T$.

Projection of test sample set $\{y_1, y_2, \ldots, y_M\}$ on V^k

$$\begin{aligned} &\left[\Phi(y_1), \ldots, \Phi(y_M)\right]^T \left[\mathrm{V}_1, \ldots, \mathrm{V}_p\right] \\ &= K_{test}\left[\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^p\right] \in R^{M \times p} \end{aligned} \tag{8}$$

Here $K_{test} = [\Phi^T(y_k)\Phi(\mathrm{x}_j)]_{M \times N} = [k(y_k, \mathrm{x}_j)]_{M \times N}$

The algorithm of KPCA is presented as Algorithm 1.

2.3 Kernel Extreme Learning Machine (KELM)

The Extreme Learning Machine (ELM) model was originally proposed in (Huang et al. 2006; Huang and Chen 2007) and developed in (Huang et al. 2014; Huang 2014), which

implements a single-hidden layer feed forward neural network (SLFN). The biggest advantage of ELM is that it can provide extremely fast learning speed and good generalization performance compared with the traditional neural network. The essence of ELM is that the network's hidden layer (mapping) can be randomly assigned and the output weights can be calculated by matrix operations without iteratively tuning.

Consider N arbitrary samples $(x_i, t_i) \in R^{n \times m}$. Then ELM is modeled as

$$f(x) = \sum_{j=1}^{L} \beta_j g_j(x_i) = h(x)\beta \quad i = 1, \ldots, N \tag{9}$$

Here L is the number of hidden layer nodes, $g(\bullet)$ is activation function, β_j is the weight vector between the jth hidden node and the output nodes. $\beta = \left[\beta_1^T, \ldots, \beta_L^T\right]_{L\times m}^T$ is the vector of the output weights between the hidden layer of L nodes and the output node and $h(x) = [g_1(x), \ldots, g_L(x)]^T$ is the output (row) vector of the hidden layer with respect to the input x. $h(x)$ actually maps the data from the n-dimensional input space to the L-dimensional hidden-layer feature space (ELM feature space).

According to the ELM theory (Huang et al. 2012) and Karush-Kuhn-Tucher (KKT) (Fletcher 1981) theorem, ELM aims to minimize the training errors and the output weights. This objective function can be expressed as follows:

$$\min_{\beta,\xi} \frac{1}{2}\|\beta\|^2 + \frac{C}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \; s.t. \; h(x_i)\beta = t_i^T - \xi_i^T \tag{10}$$

Here ξ is the training error matrix on training data, $\xi_i = [\xi_{i1}, \xi_{i2}, \ldots, \xi_{im}]^T$ $\xi_i = [\xi_{i1}, \xi_{i2}, \ldots, \xi_{im}]^T$is the ith column of ξ with respect to the training sample x_i, N and m are the number of training samples and classes, and C is a regularization parameter which trades off the norm of output weights and training errors.

The optimization problem in Eq. (10) can be efficiently solved. According to (Huang et al. 2012), the optimal β which minimizes Eq. (10) can be analytically obtained as

$$\beta = \mathrm{H}^{\mathrm{T}}\left(\frac{\mathrm{I}}{\mathrm{C}} + \mathrm{HH}^{\mathrm{T}}\right)^{-1}\mathrm{T} \tag{11}$$

Here I is an identity matrix.

$$T = \begin{bmatrix} t_1^T & \ldots & t_N^T \end{bmatrix}_{N\times m}^T$$
$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} g_1(x_1) & \ldots & g_L(x_1) \\ \vdots & \ldots & \vdots \\ g_1(x_N) & \ldots & g_L(x_N) \end{bmatrix}_{N\times L}$$

In ELM, $h(x_i)$ denotes the output of the hidden-layer with regard to the input sample x_i. H is called the feature mapping matrix due to the fact that it represents the corresponding feature space mapping of the given N training samples. Feature mapping $h(x_i)$ maps the data x_i from the input space to the hidden-layer feature space, and the feature mapping matrix H is irrelevant to the training target values t_is.

After obtaining the optimal output weights β, the decision score of the corresponding to a new data points (x_t) can be predicted fast and accurate by

$$f(x_t) = h(x_t)\beta \tag{12}$$

However, in this specific case, the feature mapping $h(x)$ may not be known to users; instead, its corresponding kernel $K(u, v)(e.g., K(u, v) = exp(-\gamma\|u - v\|^2))$ is given to users. The dimensionality L of the feature space (number of hidden nodes) need not be given either. The ELM like this is called the Kernel ELM (KELM). If a feature mapping $h(x)$ is unknown to the users, one can apply Mercers conditions on ELM. A kernel matrix for ELM is defined as follows (Huang et al. 2012):

$$\Omega_{\mathrm{ELM}} = \mathrm{HH}^{\mathrm{T}} : \Omega_{ELM_{i,j}} = h(x_i)h(x_j) = K(x_i, x_j) \tag{13}$$

The kernel matrix Ω_{ELM} is neither relevant to the number of output nodes m nor to the training target values t_is. The kernel matrix $\Omega_{\mathrm{ELM}} = HH^T$ is only related to the input data x_i and the number of training samples N. The size of kernel matrix $\Omega_{\mathrm{ELM}} = HH^T$ is $N \times N$.

Then, the output function of KELM classifier can be written compactly as

$$\begin{aligned} \mathrm{f(x)} &= \mathrm{h(x)H}^{\mathrm{T}}\left(\frac{\mathrm{I}}{\mathrm{C}} + \mathrm{HH}^{\mathrm{T}}\right)\mathrm{T} \\ &= \begin{bmatrix} K(\mathrm{x}, x_1) \\ \vdots \\ K(\mathrm{x}, x_N) \end{bmatrix}\left(\frac{\mathrm{I}}{\mathrm{C}} + \Omega_{ELM}\right)^{-1}\mathrm{T} \end{aligned} \tag{14}$$

The algorithm of KELM is presented as Algorithm 2.

3 Representation Model of Fuzzy XML Document

3.1 Fuzzy XML Document Tree Model

Obviously, finding all element/attribute features of the fuzzy XML documents is a core task. To captures features of fuzzy XML documents, we need first to establish a suitable model to represent the fuzzy XML document. XML documents are presented as ordered labeled trees in most existing approaches in (Dalamagas et al. 2006; Tekli and Chbeir 2012; Maguitman et al. 2005; Tekli et al. 2015). In this ordered labeled tree, the nodes represent elements/ attributes and are labeled with the corresponding element/ attribute label names, which are

ordered following their orders of appearance in the fuzzy XML document.

A fuzzy XML document which represents hierarchically structured information can be also presented as a rooted ordered labeled tree. But the fuzzy XML document is clearly different from the crisp XML document because the fuzzy XML document contains several special fuzzy constructions, which is attribute *Poss* and *Type*, in addition to element *VAL* and *Dist*. These fuzzy constructions contain fuzzy information of associated tree nodes. To reduce the model complexity, the redundant elements/attributes are deleted and the pertinent information (including fuzzy information) of the elements/ attributes is encapsulated in the node of the fuzzy XML document tree. We adopt the following considerations in order to reach such a goal.

We note that *Type* always appears as the first child attribute of the *Dist* and *Poss* always appears as the first child attribute of the *VAL*. They are considered redundant data and increase the computational complexity. Therefore, all of *Type* and *Poss* attribute are no longer reserved when a fuzzy XML document is mapped into a fuzzy XML document tree. But *Val* values are remained because they need to be considered while calculating the values of TF-IDF (cf. Section 3.2). So, we need to copy *Val* values into its sibling (element/attribute) nodes in processing of transformation. Similarly, we can disregard the *Type* and *Dist* elements if it is not affecting the tree structure and the depth of the other node.

Based on the discussion above, we present a new **F**uzzy **X**ML **D**ocument **T**ree **M**odel (FXDTM for short).

Definition 1 (Fuzzy XML document tree) Formally, we model a fuzzy XML document as a rooted ordered labeled tree FXDTM = {*N*, *E*, *L*, *T*, *FT*, *AC*}. Here.

- *N* is the set of nodes in tree FXDTM.
- *E* is the set of edges, which reflect the hierarchical structure of the tree FXDTM.
- *L* is the set of labels of the elements and attributes corresponding to the nodes in *N*.
- *T* is the set of data types, including the basic element data types and attribute data types.
- *FT* is the set of fuzzy data types of nodes *N*.
- *P* is the set of *Poss* values of nodes *N*.

We need to hold related information of elements/ attributes (e.g. label, the fuzzy value) when the fuzzy XML document is transformed into the fuzzy XML document tree model (FXDTM) we proposed. Hence, following our tree representation model, a fuzzy XML document tree node is modeled as follows:

Definition 2 (Fuzzy XML document tree node) A node $n \in N$ of FXDTM is represented by a quintuple n = {*NodeLabel*, *NodeDepth*, *NodeFuzzy*, *NodeFuzzyType*, *NodePossValue*}. Here.

- *NodeLabel*$\in L$ is the label name of the node.
- *NodeDepth* is the nesting depth of the node in the fuzzy XML document. The depth of the root node is defined to be 1. If the parent of node n is at depth d, then the depth of node n is d + 1.
- *NodeFuzzy* is used to indicate if the node is a fuzzy or crisp one. If the value of *NodeFuzzy* is 1, the node is a fuzzy one. If the value of *NodeFuzzy* is 0, the node is a crisp one. In particular, the value of *NodeFuzzy* of the node *Dist* or *Val* is 0.
- *NodeFuzzyType*$\in FT$ denotes the type of possibility distribution, disjunctive or conjunctive distributions. For a crisp node, its value is equal to Null.
- *NodePossValue* is the memberships of the node or the possibility of attribute values of elements. For a crisp node, the value of *NodePoss* is equal to 1.

To sum up, FXDTM is a rooted ordered tree, in which the nodes represent the fuzzy XML elements/ attributes. Nodes are ordered following their order of appearance in the fuzzy XML document. And FXDTM representation model that we proposed here considers the most common characteristics (e.g., label, the fuzzy values) of fuzzy XML documents.

After the fuzzy XML document mentioned in Section 2.1 is converted into the FXDTM tree model in this way, we can extract the feature information of nodes and apply the approach based on KELM to classify the fuzzy XML documents represented by the proposed FXDTM tree model. Figure 2 shows a FXDTM tree instance that basically comes from the corresponding fragment of a fuzzy XML document is described in Fig. 1. Without loss of generality, when we use "node" we mean "element node" or "attribute node" in following section.

3.2 Representation Structured Vector Space Model of Fuzzy XML Document Tree

Although we use a new fuzzy XML document tree model FXDTM to better represent the feature information of the fuzzy XML document, a fuzzy XML document needs to be represented in the form of eigenvectors in order to apply machine learning algorithm based classification (Tang et al. 2016). Therefore, in order to classify the fuzzy XML documents, it is necessary to transform the fuzzy XML documents from FXDTM model to vector model. That is, instead of directly transforming the vector model from the fuzzy XML document, we have gone through the intermediate transformation of the FXDTM tree. The advantage of this method is that FXDTM tree can more accurately reflect the hierarchical structure and fuzzy information of feature terms compared with plain text documents.

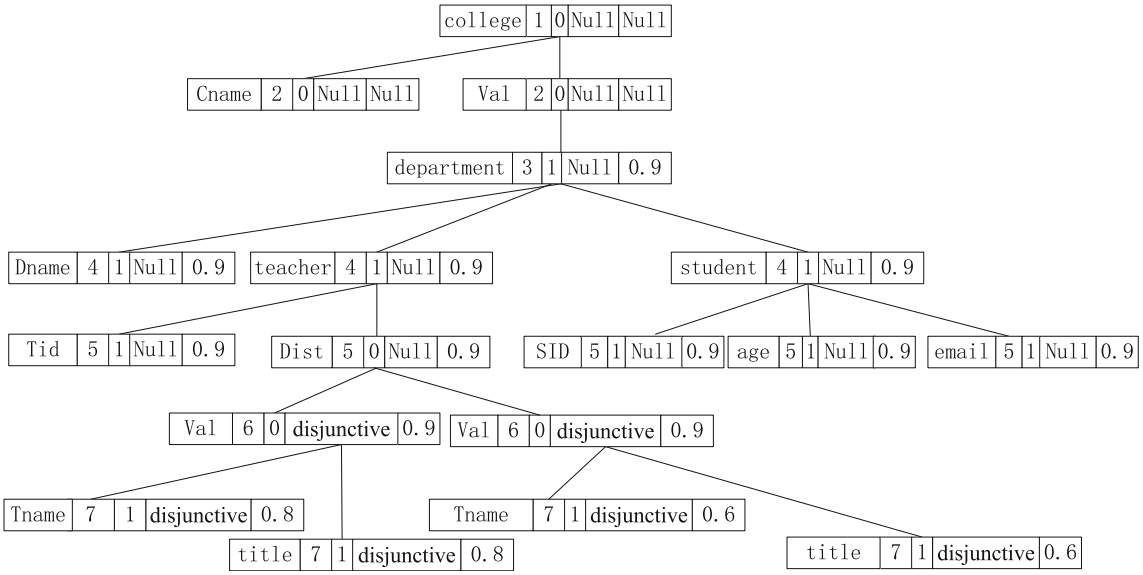

Fig. 2 A FXDTM tree corresponding to the fuzzy XML document in Fig. 1

Vector Space Model (VSM) (Salton and McGill 1983) is often used to represent plain text documents, which takes distinct feature terms TF-IDF value as feature vectors. However, VSM cannot directly represent structural information of a semi-structured (XML) document. Structured Link Vector Model (SLVM) is proposed in (Yang and Chen 2002) based on VSM to represent semi-structured documents, which contains both semantic and structural information. Given a set of XML documents D, an XML document $d \in D$. Structured Vector is defined as

$$d_{sv} =< d_1, \ldots, d_n > \tag{15}$$

where d_i is a feature vector of the ith XML document feature terms calculated as

$$d_i = \sum_{j=1}^{m} \left(TF\left(w_i, Doc.e_j\right)\cdot\varepsilon_j\right)\cdot IDF(w_i) \tag{16}$$

where w_i is the ith term, ε_j is a unit vector corresponding to the element node e_j, and m is the number of nodes corresponding to term w_i.

In SLVM, each d_{sv} is a feature matrix R$n\times m$. d_i is a m dimensional feature vector which consists of XML element node the corresponding to a same feature term.

A Modified Structured Vector Space Model (MS-VSM) based on SLVM to represent fuzzy XML documents is proposed in (Zhao et al. 2017), which not only achieves the representing of semantic and structural information of fuzzy XML documents, but also makes the feature vector to carry fuzzy information.

Given a set of fuzzy XML documents FD, a fuzzy XML document $x \in$FD. Modified Structured Vector Space Model (MS-VSM) to represent fuzzy XML document is described as

$$x_{MS-VSM} =< x_1, \ldots, x_n > \tag{17}$$

where x_i is a feature vector of the ith fuzzy XML document feature terms calculated as

$$x_i = \sum_{j=1}^{m} \left(TF\left(L_i, FXDoc.n_j\right)\cdot\varepsilon_j\right)\cdot IDF(L_i) \tag{18}$$

where m is the number of nodes corresponding to feature terms L_i (Label) in fuzzy XML document FXDoc, $FXDoc.n_j$ is the jth node corresponding to feature terms L_i in fuzzy XML document FXDoc, ε_j, which is the unit vector of $FXDoc.n_j$ and define in similarity matrix by users according to their concrete application in SLVM (Yang and Chen 2002), is now the product of the reciprocal of node depth ($n_j.NodeDepth$) and node fuzzy value ($n_j.NodePossValue$).

$$\varepsilon_j = \frac{1}{n_j.NodeDepth}\cdot n_j.NodePossValue \tag{19}$$

Example 1 shows the calculation process of eigenvectors corresponding to feature terms.

Example 1 To illustrate the calculation process of eigenvectors, we calculate the corresponding eigenvectors for the two feature terms "teacher" and "title" in Fig. 2:

$$x_{teacher} = \frac{1}{18}\times\frac{0.9}{4}\times log\frac{100}{1+80}$$
$$x_{title} = \left(\frac{1}{18}\times\frac{0.8}{7}+\frac{1}{18}\times\frac{0.6}{7}\right)\times log\frac{100}{1+50}$$

The FXDTM tree has 18 nodes in Fig. 2. Here, suppose that the sum of documents in the Corpora is 100, the number of documents in which the feature term "teacher" appears is

80, the number of documents in which the feature term "title" appears is 50. The eigenvectors of a fuzzy XML document are derived from the corresponding eigenvectors of several feature terms.

4 Classification of Fuzzy XML Document with KPCA-KELM

An ELM-based double hidden layer framework is proposed in (Zhao et al. 2017). The proposed framework includes the feature extraction with Extreme Learning Machine and the classification with Kernel Extreme Learning Machine. Considering that the KPCA is very suitable for extracting the interesting non-linear characteristics from high- dimensional space data, we choose the KPAC as feature extractor in this paper. In this section, first of all, we introduce preprocessing of fuzzy XML documents, which produce raw data of feature representation of fuzzy XML documents. And then we propose a new KPCA-KELM framework for fuzzy XML document classification. The overall architecture of the proposed KPCA-KELM includes the training stage and the testing stage.

4.1 Preprocessing

In (Zhao et al. 2017), the fuzzy XML document is directly represented as a Modified Structured Vector Space Model (MS-VSM). In this paper, we use the method described in Section 3 to represent the fuzzy XML document samples in order to represent more accurately information.

To obtain the raw data of the characteristics of the fuzzy XML document, we need to transform the fuzzy XML document into FXDTM tree, and calculate the feature value of each element. For each sample either for training or for testing, the necessary pre-processes such as size normalization are implemented to facilitate the feature extraction process. We propose Algorithm 3 to implement this preprocessing. Assume that there is a fuzzy XML document set $FD = \{x_i\}$ $(i = 1, \ldots, N)$. Then, the raw data of the train sample set (D) and test sample set (T) is obtained. The next process is to extract the KPCA features from the raw data and realize the final classification of the test sample. This preprocessing can be realized as Algorithm 3.

4.2 KPCA-KELM Classification Architecture

The starting-point of the proposed KPCA-KELM is as follows. At first, one can extract more information by using suitable nonlinear features from a given raw dataset. The KPCA is very well suited to extract interesting nonlinear features in the data. It is particularly important that KPCA has the advantage of capturing the nonlinear features in high dimensional space (Schölkopf et al. 1998; Iosifidis et al. 2015). The KPCA maps the raw data obtained in the pre-processing stage to nonlinear feature space. In addition, it is easy to understand that dimensionality reduction before performing classification tasks can improve classification accuracy and reduce training time. At the same time, a feature selection technique is employed for pre-processing before the classification models are constructed. We try to find out which features are most relevant to fuzzy XML documents through feature selection. Feature selection is applied to identify the informative features, and after then several feature subsets with top-p ranked features are fed to the classifier for classification and performance evaluation. Secondly, in order to solve the classification problem, we need to find a suitable machine learning classification algorithm. Compared with SVM, the KELM can achieve comparative or better performance with much easier implementation and faster training speed in many classification tasks (Huang et al. 2014; Huang 2014; Huang et al. 2012). That is, the KELM requires much less learning time and has good generalization accuracy. These characteristics inspire us to use the KELM to solve the fuzzy XML document classification problem. In a word, we try to use KELM classifier to classify fuzzy XML documents by combining the KPCA feature extraction method and suitable feature selection.

It is shown that the proposed KPCA-KELM architecture is structurally divided into two separate steps: 1) feature extraction and 2) classification. In the first step, a KPCA-based extractor is developed to extract sparse features of the input raw data, which can help to exploit hidden information among training samples. The KPCA first maps the raw input data into some high dimensional feature spaces via a nonlinear kernel function (e.g. Gaussian kernels and polynomial kernels) and then performs linear PCA on the mapped data. After a series of matrix operations, the largest first p eigenvalues and the corresponding eigenvectors need to be found. Through such feature selection and dimensionality reduction, the most representative relevant features can be identified. The extracted features can be used as a basis for classification after feature selection. It should be noted that the actual physical meaning of extracted features is not very explicit. The eigenvectors in the feature space do not correspond to the feature terms in the input raw data space. For example, the FXDTM tree in Fig. 2 has 14 feature terms, which are "college", "Cname", "Val", "department", "Dname", "teacher", "student, "Tid", "Dist", "SID", "age", "email", "Tname", and "title". Suppose we want to select the first 5 eigenvectors in the feature space. The extracted eigenvectors only include the largest common information of 14 feature terms. The first p principal components have the largest amount of common information relative to the input raw data. The second step of the KELM-based classification is performed for final decision making by using

projections of feature extraction results as the inputs of classifier. Although the SVM has gained much more popularity due to its mature theory as well as the satisfactory classification performance, the KELM still has its own advantages in some aspects. The KELM method improves the disadvantage of ELM method: random assignment and hidden layer bias cause unstable output results. At the same time, the hidden layer feature mapping does not need to be known. For example, the Gaussian kernel function is applied in this paper. Two main parameters presented in KELM with Gaussian kernel are penalty parameter C and kernel parameterγ, which play an important role in model construction. Through these two steps, the KPCA-KELM model provides an improved performance for the classification of fuzzy XML documents. The proposed framework of KPCA-KELM is shown in Fig. 3.

In first (KPCA) step, the nonlinear projection produces almost orthogonal eigenvectors V^k which are linearly independent in the training stage. In the testing stage, the principal components of the test sample y can be extracted by projecting $\Phi(y)$ onto the eigenvectors V^k in F as Eq. (7). In second (KELM) step, Ω_{train} is calculated by Eq. (13) in the training stage. In the testing stage, the final classification of the sample is realized by Eq. (14). The algorithm of KPCA-KELM is presented as Algorithm 4.

Overall, to capture the principal component features, a representative raw dataset is constructed with Algorithm 3 from the fuzzy XML document set. And then, the KPCA features are extracted from these representative raw datasets. Finally, in the new feature space, the KELM classifier is used to realize classification.

5 Experimental Evaluation

In this section, we present experiments conducted in order to verify the effectiveness and efficiency of the proposed KPCA-KELM framework. We compare it with the other existing main relevant state-of-the-art classifiers (original ELM, SVM etc.) in the literature in terms of classification accuracy and training time cost on various datasets.

5.1 Experimental Settings

Although a variety of classification methods are presented in the literature (Huang et al. 2006; Suykens and Vandewalle 1999; Blatman and Sudret 2011; Paliwal and Kumar 2009; Kamgar-Parsi and Kanal 2010; Gupta et al. 2019; Palshikar et al. 2018), we choose popular techniques to compare the computational performance of different classifier.

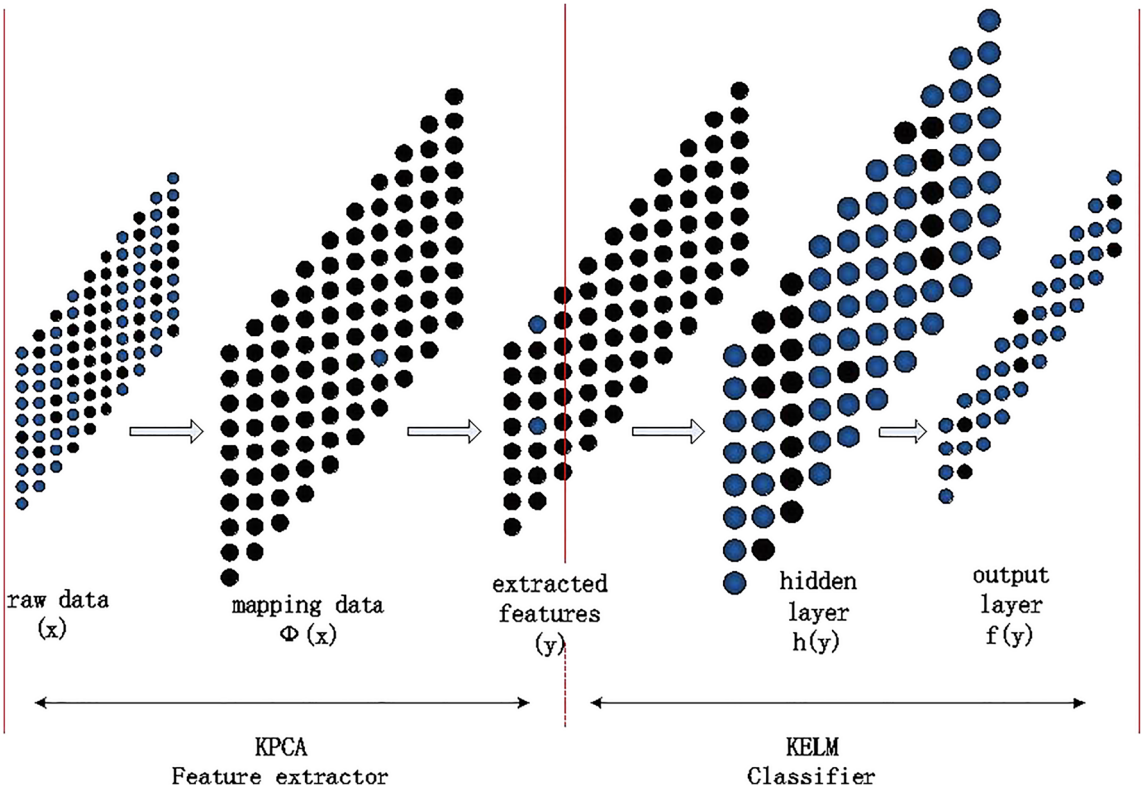

Fig. 3 The model architecture of KPCA-KELM

The current version of ELM and Kernel ELM[1] has been employed for the performance analysis. For SVM classifier, the classic algorithm (Suykens and Vandewalle 1999) has been rewritten to verify its performance in terms of different parameters which are consistent with other classifiers.

In all the experiments, we apply the original ELM (Eq. (12)) and the Kernel ELM (Eq. (14)) algorithms for different ELM mapping space.

In the case of original ELM space mapping, we have employed the Sigmoid activation function:

$$g(a,b,x) = \frac{1}{1 + exp(-(a \cdot x + b))} \tag{20}$$

Here all the hidden node parameters a and b are randomly generated. The number of hidden nodes L is set as $\{2^6, 2^7, \ldots, 2^{15}\}$. The regularization parameter C is set as $\{2^{-10}, 2^{-9}, \ldots, 2^9\}$.

In the case of Kernel ELM space determination, we employed the Gaussian kernel function:

$$K(u,v) = exp\left(-\gamma \|u - v\|^2\right)\Big) \tag{21}$$

where the value γ of the Gaussian kernel is set in the range $\{2^{-5}, 2^{-4}, \ldots, 2^4\}$. The optimal value for the regularization parameter C is set as $\{2^{-10}, 2^{-9}, \ldots, 2^9\}$.

The same Gaussian kernel function is adopted in the KPCA-KELM and SVM. In order to gain an unbiased estimate of the generalization accuracy, the all experiments will be repeated and averaged over 20 runs to obtain results for the classification comparison, and the mean and standard deviation are given.

All the experiments were conducted on a computer with an Intel Core i7 processor, 8GB RAM. All the simulations are carried out in MATLAB R2014a and JDK 1.6 environment, which runs on Windows 7 operating system with synthetic and real datasets.

5.2 Datasets

To test our method's effectiveness, we used the following synthetic and real datasets for the experimental evaluation.

We experimented with synthetic datasets[2] from five different domains. Each domain has a corresponding XML DTDs. The characteristics of these XML DTDs represented different application domains are shown in Table 1. To obtain fuzzy XML documents, at first, we used our adaptation of the XML documents generator to generate the corresponding XML documents based on these XML DTDs. And then, we manually add fuzzy construct to these XML documents during the top-down traversal of a XML document. To be specific, at each visited node N, we randomly generate a fuzzy construct (*Dist* node D with *Type* node as first children) to be the child of N during the top-down traversal of an XML document. The original children of N become the children of the newly generated *Dist* node D, which have fuzzy construct (*Val* node with *Poss* node as first children) being assigned with random probabilities to specify the membership degree of the given elements.

Table 1 Characteristics of the synthetic datasets

Domain	DTD	No.Docs	Element	Max-depth
Auction	yahoo.dtd	40	10	5
University	reed.dtd	40	12	5
Protein	psd7003.dtd	20	18	7
Dblp	dblp.dtd	60	24	6
SigmodRecord	SigmodRecord.dtd	40	16	6

For real datasets, we considered the ABC News,[3] IBM Developer Works[4] and Wikipedia[5] as three original real datasets. Of course, we need to make use of a random data generation method that transformed the XML documents in original real datasets into a fuzzy XML document. That is, we artificially add fuzzy nodes to the XML documents using same approaches to deal with XML documents in synthetic datasets.

After obtaining fuzzy XML documents of synthetic and real datasets, we choose a part of them as their subsets. For example, IBM Developer Works is composed of around 1200 XML documents classified in 6 classes, out of which we choose 3 classes and 600 random XML documents as a subset. In this way, synthetic dataset is randomly split into 4 subsets (T1, T2, T3, T4), and real dataset is randomly split into 6 subsets (T5, T6, T7, T8, T9, T10), respectively.

Subsequently, we transform these fuzzy XML documents in synthetic and real datasets (T1-T10) ordered by increasing samples size into MS-VSM which we proposed using Algorithm 3. Consequently, we obtain 10 datasets represented with MS-VSM as input raw data of KPCA-KELM frameworks and other classifiers. The key characteristics of the real datasets are summarized in Table 2.

For each dataset, 80% samples are used for training while the rest 20% samples are used for testing. We have normalized all input variables to the (−1, 1) range for all the classifiers.

5.3 Performance Comparison and Discussion

In this part, we compare the performance of the above-mentioned methods for classification over ten raw datasets,

[1] http://www.ntu.edu.sg/home/egbhuang/
[2] http://www.cs.washington.edu/research/xmldatasets/
[3] http://abcnews.go.com/
[4] http://www.ibm.com/developerworks/develop/
[5] http://wikipedia.c3sl.ufpr.br/

Table 2 Key Characteristics of the datasets

Dataset	Samples	Features	Classes	Description
T1	80	12	2	Synthetic datasets
T2	100	16	3	Synthetic datasets
T3	160	18	4	Synthetic datasets
T4	200	24	5	Synthetic datasets
T5	200	32	3	ABC News
T6	400	32	6	ABC News
T7	600	45	3	IBM Developer Works
T8	1200	45	6	IBM Developer Works
T9	2400	56	10	Wikipedia
T10	4800	56	20	Wikipedia

including both classification accuracy and training time cost, to demonstrate both the effectiveness and the computational efficiency of the proposed KPCA-KELM framework.

First, we demonstrate the selection of parameters in original ELM and KPCA-KELM frameworks on synthetic datasets (T1-T4). Compared with the traditional learning algorithms, fewer parameters need to be chosen in the training process of them.

The performance of original ELM is mainly influenced by the cost parameter C for the regularized least mean square calculation and the number of hidden neurons L. Here, these two key factors will be examined in detail.

Figure 4 shows the 3-D testing accuracies curves of original ELM in terms of (C, L). It is shown that the influence of C and L on the performance of original ELM is different. As shown in Fig. 4, the accuracy curves become smoother and steadier as L increases when C has different values. On the other hand, it is shown that the trend of accuracy curves does not change very much as C increases if L is set large enough (e.g., $L > 2^{10}$ in our simulations). That is, selection of C is less important, while L should be large enough.

Being different from original ELM, the performance of KPCA-KELM is mainly influenced by the cost parameter C and kernel parameter γ in Gaussian kernel function. In order to achieve good generalization performance, the cost of parameter C and kernel parameter γ and KPCA-KELM also need to be chosen appropriately. Thus, the best combination of (C, γ) of KPCA-KELM with Gaussian kernel needs to be found.

Figure 5 shows the 3-D testing accuracies curves of KPCA-KELM in terms of (C, γ). We can clearly see that the testing accuracy of KPCA-KELM is heavily influenced by the parameter γ. The testing accuracy is getting maximum value when the value of γ is close to 1. Compared to the parameter γ, the parameter C is not sensitive to the performance of KPCA-KELM when C is large enough. That is, the classification accuracy keeps stable as C increases if γ is set to a fixed value (e.g., $\gamma = 1$).

It should be mentioned that the KPCA-KELM seems to achieve a similar testing accuracy compared with the original ELM, and its performances tend to be quite stable in a wide range of C ($C > 2^6$). This improvement may be because the KPCA-KELM framework is able to effectively capture the nonlinear relationship existed in the fuzzy XML document datasets with the aid of the Gaussian kernel. Meanwhile, the performance of KPCA-KELM is more sensitive to the parameter C than that of the original ELM. As C increases, the accuracy of KPCA-KELM is larger fluctuating than ELM.

To investigate whether feature extraction can further improve the performance of KPCA-KELM for fuzzy XML documents classification, we further conduct the experiments in the reduced feature space which was obtained using KPCA algorithm.

This parameter that affects the performance of the KPCA-KELM framework is the number of principal components (p) of feature extraction in KPCA Algorithm 1, We discuss the impacts of the parameter (p) on the performance of the KPCA-KELM framework in detail.

Figure 6 shows the change curves of testing accuracies under different number of the extracted features (p). It is shown that KPAC-KELM gets different classification results on different parameter p, and the trends of classification

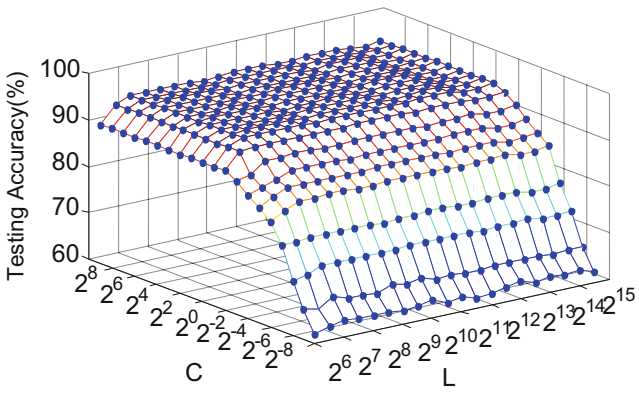

Fig. 4 Testing accuracy curve of ELM in terms of (C, L)

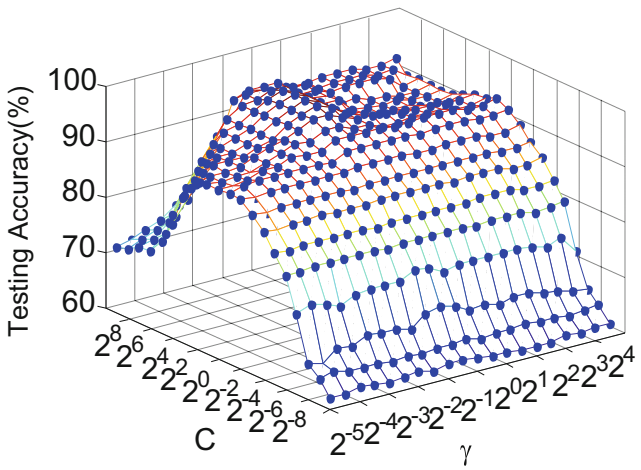

Fig. 5 Testing accuracy curve of KPCA-KELM in terms of (C, γ)

accuracy seems to be increasing when the value of parameter p is growing. It can be also seen that the testing accuracies curves keep stable on different real datasets (T5-T9) when the value of parameter p is large enough.

Figure 7 shows the change curves of training time in terms of the number of extracted features (p). We can see that the training time seems to be increasing when the value of parameter p is growing. Moreover, the training time is increasing with the increase of the number of samples when the value of p is the same.

From the above analysis, we can find that KPCA-KELM framework has improved its performance for fuzzy XML document classification with the aid of feature extraction using KPCA.

Furthermore, in order to evaluate the effectiveness of the proposed KPCA-KELM approach, the SVM was also implemented for comparison. Here we only considered the Gaussian kernel for SVM classification. The optimal parameter pair (C, γ) was employed to construct the SVM predictive model.

In summary, user-specified optimal parameters used in our simulations are adopted as follows: (C, L) for original ELM is (50, 1000), (C, γ) for KPCA-KELM with Gaussian kernel is (25, 1), and the same optimal parameters are adopted in the SVM.

Hereunder, we used more complicated real datasets to verify the classification performance of KPCA-KELM over the other existing main relevant state-of-the-art classifiers (original ELM, SVM, kernel ELM) in the literature.

For KPCA-KELM, feature extraction is performed using KPCA, and the resulting features are randomly projected before ELM-based classification; for ELM and kernel ELM, the randomly mapped raw data is input for classification. For fair comparison, in the feature classification stage, the same Gaussian kernel function is adopted in the KPCA-KELM, kernel ELM and SVM. In addition, the sigmoid activation function is adopted in original ELM.

To analyze numerical results, two performance metrics have been graphed: testing accuracy, training time. The average performance comparison of different methods was gathered in Table 3 and plotted in Figs. 8 and 9.

One of the most important evaluation criteria is the testing accuracy. As shown in Table 3 and Fig. 8, we can conclude that KPCA-KELM and KELM have similar testing accuracy performance for all of datasets because the same kernel function and parameter are adopted in each algorithm. Moreover, the testing accuracy comparison shows that three ELM-based algorithms are similar to the SVM-based method. Importantly, we observe that the KPCA-KELM algorithms cannot obtain a huge advantage compared with other algorithms on classification accuracy. Furthermore, Fig. 8 shows the standard deviations for testing accuracy on the same dataset are generally identical due to the similar accuracies for the four different classifiers. Finally, it is shown that there are several low testing accuracies such as T1, T3 for small datasets. This indicates that the performance of the classifiers really depends on the nonlinear nature of datasets themselves rather than the number of samples.

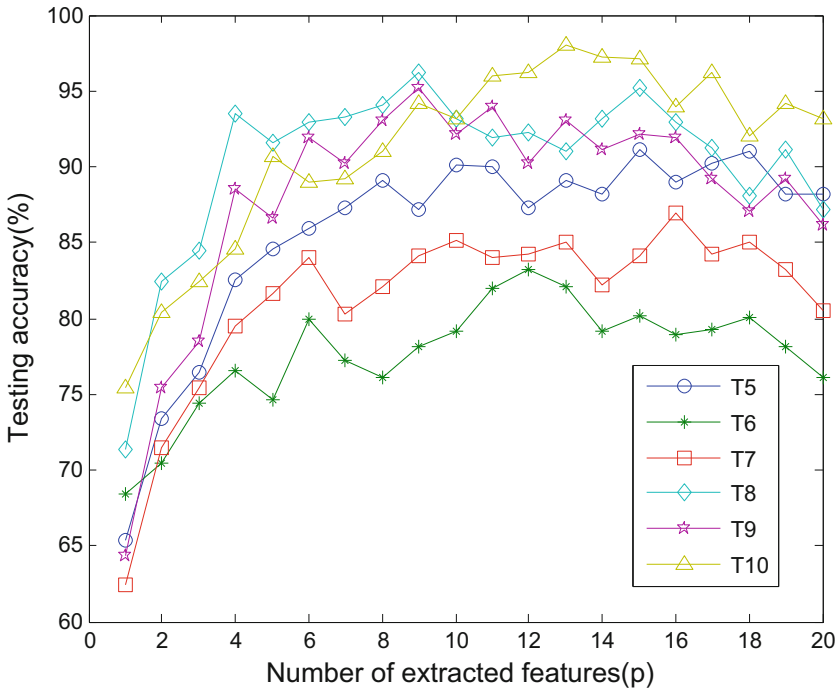

Fig. 6 Testing accuracy curve of KPCA-KELM in term of p

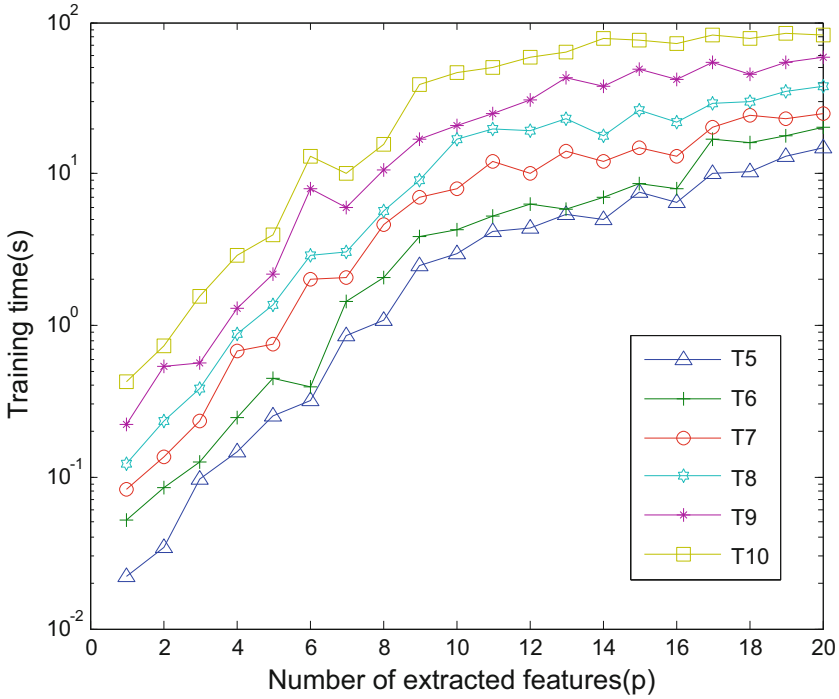

Fig. 7 Training time curve of KPCA-KELM in term of p

Another important evaluation criterion of learning algorithm is the training time, of which the comparison is presented in Fig. 9. The Fig. 9 shows the change curves of training times (in a logarithmic scale) of the four classifiers: KPCA-KELM, ELM, KELM and SVM on the different datasets. For almost all the data sets, KPCA-KELM is the fastest. However, the ELM, which is the second slowest one in the smaller datasets, is similar to or slightly faster than KELM for the bigger datasets. It seems that, for the large datasets, the computational cost of process the whole training set (N samples) in KELM is higher than the cost of calculate the matrix H in ELM. Moreover, from the experimental results we can see that SVM is a significantly more time-consuming algorithm than ELM-based algorithms for all datasets. The difference between KPCA-KELM/ELM and SVM increases with the size of the dataset. Since the major part of the training time is the matrix calculation, the feature extraction method in KPCA-KELM removes redundant features, the effect of feature extraction is obvious and the training time of KPCA-KELM decreases to a great extent.

In summary, the above results validate that the KPCA-KELM learning framework achieves a significant improvement on the performance of fuzzy XML documents classification compared with the existing other methods although the classification accuracy of the KPCA-KELM is slightly higher than that of others. However, it is acceptable to the common

Table 3 Performance of classifiers on the different datasets

Dataset	ELM(L = 1000)		SVM		KELM		KPCA-KELM	
	Testing Accuracy(%)(SD)	Training Time(s)	Testing Accuracy(%)(SD)	Training Time(s)	Testing Accuracy(%)(SD)	Training Time(s)	Testing Accuracy(%)(SD)	Training Time(s)
T1	63.8(±1.25)	0.19	64.13(±1.85)	1.65	64.28(±1.13)	0.056	63.96 (±0.53)	0.022
T2	76.32(±1.80)	0.22	75.28(±3.52)	5.84	76.05(±1.71)	0.071	76.23(±0.23)	0.034
T3	73.08(±2.35)	0.46	72.56(±2.38)	6.32	72.84(±1.65)	0.75	71.95(±0.61)	0.26
T4	77.92(±1.36)	0.52	76.78(±2.35)	10.47	77.25(±2.13)	0.92	77.93(±0.82)	0.68
T5	92.94(±0.15)	1.23	93.14(±3.20)	12.36	92.05(±1.34)	1.98	93.26(±0.58)	0.25
T6	86.14(±0.85)	0.96	85.82(±1.80)	32.51	86.38(±1.38)	1.83	85.42(±1.54)	0.62
T7	89.26(±1.43)	3.25	89.45(±1.08)	48.79	89.42(±1.05)	8.12	88.51(±1.39)	0.25
T8	98.24(±0.67)	2.05	98.14(±1.23)	146.35	98.05(±0.84)	3.59	98.26(±1.18)	0.98
T9	92.35(±0.88)	6.18	94.78(±2.45)	236.54	94.09(±1.36)	16.41	93.25(±1.29)	1.05
T10	96.36(±1.28)	10.15	96.15(±0.81)	1020.05	98.40(±1.15)	30.25	97.83(±0.95)	3.01

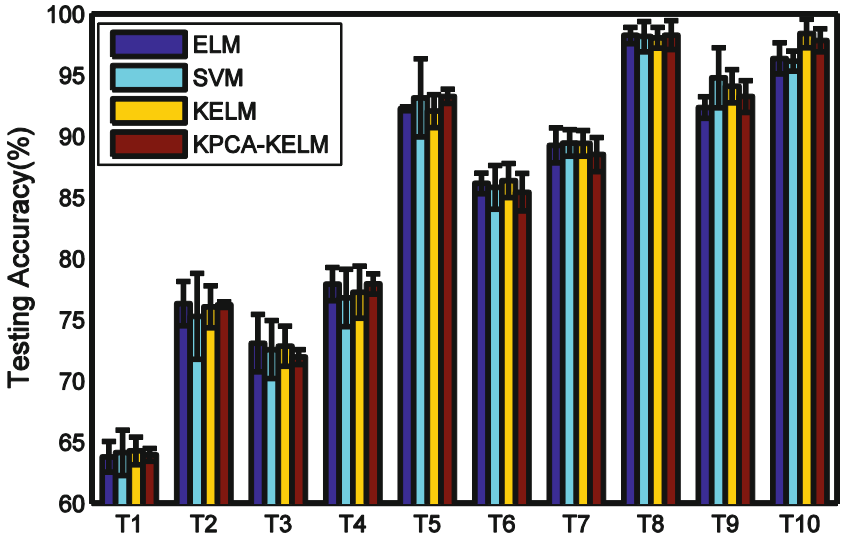

Fig. 8 Testing accuracy of different classifiers

personal computer with the implementation environment mentioned at the beginning of this section. And, in the meantime, the training time of KPCA-KELM is extremely faster than the above other methods.

The major difference between KPCA-KELM and other methods is that before feature classification, KPCA-KELM uses KPCA algorithm to obtain sparse representation of the input raw data. The compact features can help to remove redundancy of the original inputs, and thus improve the overall learning performance.

Then, we discuss some limitations of KPCA-KELM. The tunable parameter of KPCA-KELM framework is the kernel spread γ, the regularization parameter C, and number of feature extraction p, which must be tuned for each particular dataset. Therefore, the classification performance of KPCA-KELM not only depends on the size (number of patterns, inputs and classes) of the dataset, and but also on result of feature extraction. However, we can clearly see that feature extraction can further decrease the training time of the KPCA-KELM.

6 Conclusion

In order to deal with the classification of the fuzzy XML documents effectively, in this paper, we first propose a novel tree representation model to capture the node information of fuzzy XML documents, and then a vector space model of representing the semantic and structure of fuzzy XML documents is employed based on the proposed fuzzy XML document tree model to extract feature of fuzzy XML document. We propose KPCA-KELM framework to classify the fuzzy XML document based on the vector space model. The core component of the proposed framework is the KELM classifier. With the aid of the feature extraction techniques (KPCA), the performance of KELM classifier is improved with extracted

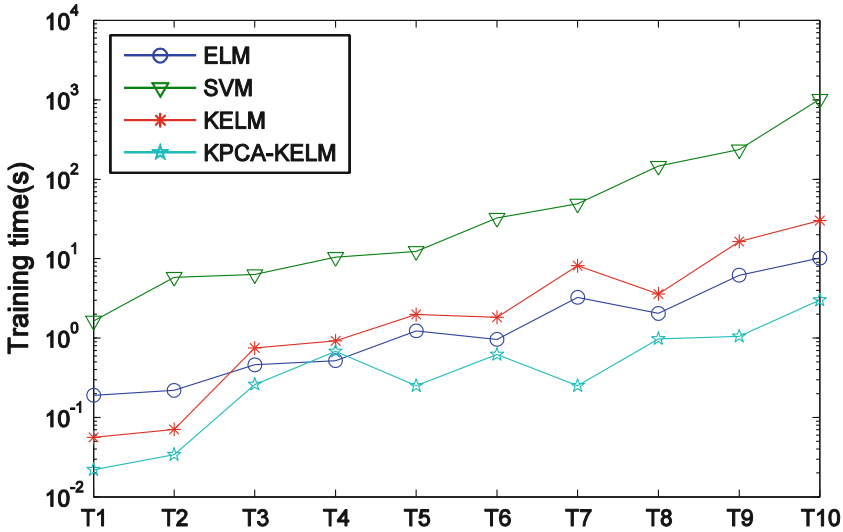

Fig. 9 Training time of different classifiers

features. Finally, the experimental results show that our algorithms can efficiently perform classification on the fuzzy XML documents. Note that our proposed KPCA-KELM approach achieves a significant reduction in training time while maintaining the same level of accuracy as the state-of-the-art baseline models. In the future, duo to the computation cost of KPCA increases dramatically with the sample size, it is intuitive to accelerate the proposed KPCA-KELM algorithm by using the cloud computing technology on a large dataset. We also plan to develop a prototype system based on the proposed framework and then simulate it for other real application systems.

Acknowledgements The authors wish to thank the anonymous referees for their valuable comments and suggestions, which improved the technical content and the presentation of the paper. This work was supported by the *National Natural Science Foundation of China* (61772269, 61370075 & 61976027) and the *Scientific Research Projects of Liaoning Educational Committee* (LQ2017003).

References

Abiteboul, S., Segoufin, L., & Vianu, V. (2006). Representing and querying XML with incomplete information. *ACM Transactions on Database Systems, 31*(1), 208–254.

Agreste, S., Meo, P. D., Ferrara, E., & Ursino, D. (2014). XML Matchers: approaches and challenges. *Knowledge-Based Systems, 66*, 190–209.

Blatman, G., & Sudret, B. (2011). Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics, 230*(6), 2345–2367.

Brzezinski, D., & Piernik, M. (2015). Structural XML classification in concept drifting data streams. *New Generation Computing, 33*(4), 345–366.

Dalamagas, T., Cheng, T., Winkel, K. J., et al. (2006). A methodology for clustering XML documents by structure. *Information Systems, 31*(3), 187–228.

Fletcher, R. (1981). Practical methods of optimization. *Constrained Optim., 2*.

Gaurav A, Alhajj R (2006) Incorporating fuzziness in XML and mapping fuzzy relational data into fuzzy XML. In: Proceedings of the 2006 ACM symposium on applied computing, ACM, Dijon, pp. 456–460

Guha, S., Jagadish, H. V., Koudas, N., & Srivastava, D. (2006). Integrating XML data sources using approximate joins. *ACM Transactions on Database Systems, 31*(1), 161–207.

Gupta, P., Chauhan, S., & Jaiswal, M. P. (2019). Classification of smart city research - a descriptive literature review and future research agenda. *Information Systems Frontiers, 21*(3), 661–685.

Huang, G. B. (2014). An insight into extreme learning machines: Random neurons, random features and kernels. *Cognitive Computation, 6*(3), 376–390.

Huang, G. B., & Chen, L. (2007). Convex incremental extreme learning machine. *Neurocomputing, 70*(16), 3056–3062.

Huang, G., Song, S., Gupta, J. N. D., & Wu, C. (2014). Semi-supervised and unsupervised extreme learning machines. *IEEE Trans. Cybern., 44*(12), 2405–2417.

Huang, S., Wang, B., et al. (2016). Parallel ensemble of online sequential extreme learning machine based on MapReduce. *Neurocomputing, 174*, 352–367.

Huang, G. B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern. Part B, 42*(2), 513–529.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing, 70*(1), 489–501.

Iosifidis, A., Tefas, A., & Pitas, I. (2015). On the kernel extreme learning machine classifier. *Pattern Recognition Letters, 54*, 11–17.

Kamgar-Parsi, B., & Kanal, L. N. (2010). An improved branch and bound algorithm for computing k-nearest neighbors. *Pattern Recognition Letters, 3*(1), 7–12.

Li, T., & Ma, Z. M. (2017). Object-stack: an object-oriented approach for top-k keyword querying over fuzzy xml. *Information Systems Frontiers, 19*(3), 669–697.

Ma, Z. M., & Yan, L. (2007). Fuzzy XML data modeling with the UML and relational data models. *Data & Knowledge Engineering, 63*, 972–996.

A.G. Maguitman, F. Menczer, H. Roinestad, et al., (2005) Algorithmic detection of semantic similarity. In: Proc. of the 14th International Conference on World Wide Web, ACM, Chiba, pp. 107–116.

Negoita, C., Zadeh, L. A., & Zimmermann, H. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems, 1*, 3–28.

Nierrman, A., & Jagadish, H. V. (2002). *ProTDB: Probabilistic data in XML, in: Proceedings of the 28th international conference on vary large data bases* (pp. 646–657). Hong Kong: VLDB Endowment.

Oliboni, B., Pozzani, G. (2008) Representing fuzzy information by using XML schema, in: Proceedings of the 19th international conference on database and expert systems application, Turin, pp. 683–687

Paliwal, M., & Kumar, U. A. (2009). Neural networks and statistical techniques: A review of applications. *Expert Systems with Applications an International Journal, 36*(1), 2–17.

Palshikar, G. K., Apte, M., & Pandita, D. (2018). Weakly supervised and online learning of word models for classification to detect disaster reporting tweets. *Information Systems Frontiers, 20*(5), 949–959.

L. Ribeiro, T. Härder (2006) Entity identification in XML documents. In: 18th GI-Workshop on the Foundations of Databases, pp. 130–134.

Salton, G., & McGill, M. (1983). *Introduction to modern information retrieval*. New York: McGrawHill Book Company.

Schölkopf, B., Smola, A., & Müller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation, 10*(5), 1299–1319.

Suykens, J., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters, 9*(3), 293–300.

Tang, J., Deng, C., & Huang, G. B. (2016). Extreme Learning Machine for Multilayer Perceptron. *IEEE Transactions on Neural Networks & Learning Systems, 27*(4), 809–821.

Tekli, J., & Chbeir, R. (2012). A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics. *Web Semantics: Science, Services and Agents on the World Wide Web, 11*(3), 14–40.

Tekli, J., Chbeir, R., et al. (2015). Approximate XML structure validation based on document-grammar tree similarity. *Information Sciences, 295*, 258–302.

Thasleena, N. T., & Varghese, S. C. (2015). Enhanced associative classification of XML documents supported by semantic concepts. *Procedia Computer Science, 46*, 194–201.

Thomo, A.., Venkatesh, S. (2008) Rewriting of visibly pushdown languages for xml data integration. In: Proc. of the 17th ACM Conference on Information and Knowledge Management, ACM, Napa Valley, pp. 521–530

Turowski, K., & Weng, U. (2002). Representing and processing fuzzy information-an XML-based approach. *Knowledge-Based Systems, 15*(1), 67–75.

Yan, L., Ma, Z. M., & Liu, J. (2009). *Fuzzy data modeling based on XML schema, in: Proceedings of 2009 ACM symposium on applied computing* (pp. 1563–1567). Honolulu: ACM.

Yang, J., & Chen, X. (2002). A semi-structured document model for text mining. *Journal of Computer Science and Technology, 17*(5), 603–610.

Zhang, X. L., Yang, T., Fan, B. Q., et al. (2012). A Novel Method for Measuring Structure and Semantic Similarity of XML Documents Based on Extended Adjacency Matrix. *Physics Procedia, 24*, 1452–1461.

Zhao, X., Bi, X., et al. (2016). Uncertain XML documents classification using extreme learning machine. *Neurocomputing, 174*, 375–382.

Zhao, Z., Ma, Z. M., Zhang, F., et al. (2017). Classification of fuzzy XML documents based on double hidden layer ELM. *Computer Engineering and Applications, 53*(4), 19–24.

Zhao, X., Wang, G., Bi, X., et al. (2011). XML document classification based on ELM. *Neurocomputing, 74*(16), 2444–2451.

Zhen Zhao received the B.S. degree in Computer Science Education from Bohai University in 2002, and he obtained his M.S. degree in Computer Application Technology from Dalian Jiaotong University in 2009. He obtained his Ph.D. degree in Computer Application Technology from Northeastern University in 2017. His research interests include fuzzy data management and knowledge graph.

Zongmin Ma is currently a full professor at Nanjing University of Aeronautics and Astronautics, China. His research interests include databases, the Semantic Web, and knowledge representation and reasoning with a special focus on information uncertainty. He has published more than two hundred papers in international journals and conferences on these topics. In addition, he (co-)authors five monographs with Springer. He is a Fellow of the IFSA and a senior member of the IEEE.

Li Yan is currently a full professor at Nanjing University of Aeronautics and Astronautics, China. Her research interests include fuzzy data modeling and spatiotemporal data management. She has published more than fifty papers on these topics. She is the author of three monographs published by Springer.