Front. Comput. Sci., 2021, 15(1): 151305
https://doi.org/10.1007/s11704-019-8395-7

RESEARCH ARTICLE

A framework based on sparse representation model for time series prediction in smart city

Zhiyong YU[1], Xiangping ZHENG[1], Fangwan HUANG (✉)[1], Wenzhong GUO[1], Lin SUN[2], Zhiwen YU[3]

1 College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China
2 Hangzhou Key Laboratory for IoT Technology & Application, Zhejiang University City College, Hangzhou 310015, China
3 School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China

Abstract Smart city driven by Big Data and Internet of Things (IoT) has become a most promising trend of the future. As one important function of smart city, event alert based on time series prediction is faced with the challenge of how to extract and represent discriminative features of sensing knowledge from the massive sequential data generated by IoT devices. In this paper, a framework based on sparse representation model (SRM) for time series prediction is proposed as an efficient approach to tackle this challenge. After dividing the over-complete dictionary into upper and lower parts, the main idea of SRM is to obtain the sparse representation of time series based on the upper part firstly, and then realize the prediction of future values based on the lower part. The choice of different dictionaries has a significant impact on the performance of SRM. This paper focuses on the study of dictionary construction strategy and summarizes eight variants of SRM. Experimental results demonstrate that SRM can deal with different types of time series prediction flexibly and effectively.

Keywords sparse representation, smart city, time series prediction, dictionary construction

1 Introduction

With the rapid development of Big Data and IoT, the concept of "smart city" has attracted much attention of researchers from academia, industry, and government in recent years. As one of the most cutting-edge issues, the smart city pays attention to perceive, correlate and excavate the key information comprehensively for intelligent management through urban computing, with the target of creating a better life for people and promoting the harmonious and sustainable growth of the city [1]. The applications of smart city have been embodied in various aspects including home automation, medical treatment, traffic management, environmental protection, public services and other fields [2–8].

However, the realization of the smart city still faces many great difficulties. One of them is how to extract and represent the sensing knowledge from massive IoT devices in order to realize the proactive event alert of smart city, as these devices record chronologically the changing states with a sampling interval. For example, the smart meter, as an important device of the smart grid, collects power consumption at intervals of 10 or 15 minutes [9]. Obviously, these observations can be viewed as time series, and the prediction of time series is an indispensable part in electric power system planning, as well as many other application scenarios.

How to predict time series quickly and effectively has been one of the research hotspots in the field of data mining for decades. Although a lot of researches have been done, the task of time series prediction is still faced with great challenges due to the characteristics of time series including big size, high dimension, continuous update and inevitable noise [10]. Faced with large amounts of complex time series and the needs of more accurate prediction in various fields, it is required to exploit a more effective model. In this paper, a systematic framework based on sparse representation model (SRM) for time series prediction is proposed as an elegant and efficient way to deal with the massive sequential data generated by IoT devices. In the last decade, there had been an increasing interest in sparse representation. The advantage of sparse representation is that the main information of a signal can be converted into a small number of components based on an over-complete dictionary. Therefore, sparse representation can not only reduce dimensionality by extracting key features but also eliminate noise by removing redundant information. The applications of sparse representation have been successfully extended from early signal processing and image analysis to other fields such as pattern recognition and machine learning [11]. At present, sparse representation has also been successfully applied to time series prediction. [9] focuses on extracting key factors affecting power load change by sparse representation and then fitting the data by ridge regression method based on extracted features. [12,13] all focus on sparse representation of dictionaries based on historical data to achieve one-step prediction, but the difference is whether to arrange historical data as the rows or columns of the dictionary. [14] extends one-step prediction to multi-step prediction and compares the performance of various sparse repre-

Received November 19, 2018; accepted October 14, 2019

E-mail: hfw@fzu.edu.cn

sentation algorithms. Nevertheless, the research on the dictionary construction strategy is not comprehensive enough. Motivated by this, this paper focuses on this aspect and eventually form a systematic framework based on sparse representation for time series prediction. The contributions of this paper are:

- According to the different strategies of dictionary construction, a systematic framework including eight variants of SRM is proposed to solve different types of time series prediction effectively.
- A novel strategy for dictionary construction considering cross-domain dependencies is proposed. This strategy can achieve the same accuracy improvement as dictionary learning without sacrificing computational complexity.
- The experimental comparison between SRM and many popular models shows the superior performance of SRM in both accuracy and complexity.

The rest of the paper is organized as follows: Section 2 reviews the related work of time series prediction. Some preliminary knowledge about sparse representation is introduced in Section 3. Section 4 focuses on the implementation of SRM. Experiments and results are presented in Section 5. Finally, in Section 6, the conclusions and future work are stated.

2 Relate work

The purpose of time series prediction is to forecast the next few values of a series that are most likely to occur in the future. It is one of the most extensively applied tasks of time series so that there is a lot of literature focused on it. In these studies, according to different research perspectives, the models of time series prediction can be divided into three categories: traditional statistical models, machine learning models, and hybrid integrated models.

2.1 Traditional statistical models

In the research of time series prediction, traditional statistical models have been developed very maturely. This category can be subdivided into two specific subcategories: linear statistical models and nonlinear statistical models. The idea of linear statistical models is based on the hypothesis that there is a linear correlation structure between historical data and the predicted value of time series. Popular linear statistical models include simple exponential smoothing (SES), auto-regressive integrated moving average (ARIMA), dynamic linear model (DLM) and a number of their variants [15]. In fact, these linear statistical models can be collectively referred to as state-space models which provide a unifying framework to define them. As the linear statistical models have the advantages of simple principle and easy implementation, they have been widely used in the fields of economic forecasting, agriculture and industrial control [15]. However, time series collected from real applications are generally nonlinear and non-stationary, which make linear statistical models less effective to complex real-world problems. Due to the limitation of linear statistical models, a lot of useful nonlinear statistical models have been proposed such as smooth transition auto regressive models (STAR) [16], and generalized auto-regressive conditional heteroskedasticity models (GARCH) [17].

The drawback of traditional statistical models is that they all make use of analytic equations with parameters to describe time series which must be satisfied with some stringent assumptions. Although they have good fitting prediction performance for time series that conform to their assumptions, these assumptions are often inconsistent with highly complex time series collected from the real world.

2.2 Machine learning models

With the rapid development of artificial intelligence, machine learning models have attracted more and more attention and have been strong competitors of traditional statistical models in time series prediction. Unlike traditional statistical models, machine learning models employ the historical data-driven approach to infer the complex dependencies between the past and the future of time series [18]. Compared with traditional statistical models, machine learning models tend to be more flexible and effective because they can obtain more accurate results by simulating human intelligence or natural phenomena. Common machine learning models include multi-layer perceptron (MLP), Bayesian neural networks (BNN), support vector regression (SVR), and random forest (RF) [19,20]. An empirical comparison of machine learning models for time series prediction has been studied in [19].

In recent years, deep learning has gained increasing popularity in the field of machine learning for time series prediction, as it had achieved superior performances on many benchmark datasets. The goal of deep learning is to stack multiple modules on top of each other to form deep networks in order to enable more expressive models that can learn more abstract representations of data. This enables deep learning to carry out unsupervised learning without prior knowledge and achieve better time series prediction performance [21]. The machine learning models based on deep learning have been successfully applied in the field of time series prediction related to the smart city. For instance, as one of the most popular types of recurrent neural network (RNN), long short term memory (LSTM) can make a more accurate prediction by using the long term dependence in the time series of power load [22]. Unfortunately, LSTM requires a lot of time to train network parameters. Therefore, many improved versions of this model are proposed, including gated recurrent unit (GRU) [23] and minimal gated unit (MGU) [24].

The success of the machine learning models lies in continuously adjusting and modifying the function structure and parameter estimation of the models with iterative learning to minimize the prediction error. However, they also have some defects, such as parameter sensitivity, local optimization, and over-fitting. Therefore, a variety of intelligent optimization algorithms are proposed and introduced into the machine learning models, such as Genetic Algorithm, Particle Swarm Optimization, and Artificial Bee Colony Optimization [25,26].

2.3 Hybrid integrated models

In general, no single model for time series prediction is optimal in every situation because each model has its unique advantages and disadvantages. In this context, hybrid integrated models worked on a higher level by ensemble methods have become a newly developing trend in the field of time series prediction recently. Hybrid integrated models focus on getting the utmost out of the strengths of individual models to improve overall prediction performance. In other words, hybrid integrated models can not only avoid selecting the wrong model by integrating all candidate models but also obtain more accurate results by making multiple predictions from different starting points [27]. A hybrid integrated model can be built from two or more models of the same or different classes. For instance, a hybrid integrated model of ARIMA and artificial neural network (ANNs) takes advantage of traditional statistical models and machine learning models in linear and nonlinear aspects respectively [28]. Recently, another hybrid integrated model based on convolutional neural network (CNN) and LSTM has been proposed for urban traffic flow prediction by capturing the spatial and temporal features from real taxis trace data [29].

Although many empirical studies show that the prediction performance of the hybrid integrated model is higher than that of an individual model, it is inevitable that the complexity of the ensemble method increases greatly. The balance between prediction performance and computational complexity is a key issue to consider. In addition, how to select a more effective way of combining still remains to be further studied.

3 Preliminary

Over the past few years, sparse representation originated from the theory of compressed sensing (CS) has received more and more attention from researchers [30]. From the mathematical perspective, the target of sparse representation is to reconstruct the original signal by selecting the sparsest linear combination of basis vectors from an over-complete dictionary. More emphatically, there are multiple combinations of signal decomposition because the number of basis vectors in an over-complete dictionary is greater than the dimension of the signal, which provides some advantages such as greater flexibility, more compact representation, and increased stability to noise [31].

The sparse representation of time series can be defined as follows: given a time series $T = [t_1, t_2, \dots, t_n]^{\mathrm{T}} \in R^n$ and an over-complete dictionary $D = [d_1, d_2, \dots, d_m] \in R^{n\times m}$ $(m \gg n)$, each column of D can be called a basis vector or an "**atom**". If T can be reconstructed by a linear superposition of m atoms, it should be expressed as:

$$T = \sum_{i=1}^{m} d_i x_i, \tag{1}$$

where x_i is the weighting coefficient of atom d_i. For the convenience of description, Eq. (1) can be rewritten by adopting a matrix-vector notation:

$$T = DX, \tag{2}$$

where the vector $X \in R^m$ can be considered as a representation of time series T based on the over-complete dictionary D. Obviously, Eq. (2) is an underdetermined linear system having the infinite number of solutions because D is a rectangular matrix having more columns than rows. Only if the majority of coefficients in vector X are zero, it can be considered as a sparse representation of T. The schematic diagram of sparse representation is shown in Fig. 1.

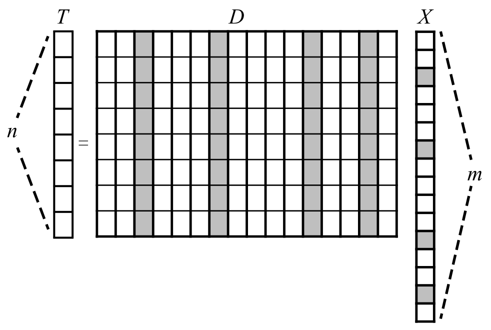

Fig. 1 Sparse representation of time series T based on the over-complete dictionary D

The sparsest representation of T can be obtained by solving Eq. (2) with the l_0-norm minimization constraint as follows:

$$\min_X \|X\|_0, \quad s.t.\ T = DX, \tag{3}$$

where $\|.\|_0$ is used to measure the sparse degree by calculating the number of the nonzero elements in vector X. On account for the noise present in the real data, the constraint is often relaxed using a quadratic penalty function. Therefore, the following error-tolerant version can be used to provide the approximate solution of Eq. (3):

$$\min_X \|X\|_0, \quad s.t.\ \|T - DX\|_2^2 \leqslant \delta, \tag{4}$$

where δ is a small positive constant that can be considered as the reconstruction error. If the objective and the constraint are flipped, Eq. (4) can be transformed to:

$$\min_X \|T - DX\|_2^2, \quad s.t.\ \|X\|_0 \leqslant \varepsilon, \tag{5}$$

The sparse factor ε is introduced to express the maximum number of nonzero entries in vector X. In addition, because the reconstruction error or the sparse factor is difficult to be determined in advance, they can be optimized simultaneously according to the Lagrange multiplier theorem. Equations (4) and (5) can be equivalently converted to an unconstrained minimization problem:

$$\min_X \frac{1}{2}\|T - DX\|_2^2 + \lambda\|X\|_0. \tag{6}$$

Equation (6) aims to minimize the reconstruction error as well as the sparse solution, and λ is a constant used to make a tradeoff between them. It is difficult to determine exactly the inherent correlation between the parameter λ and the sparse degree of vector X [32]. In general, the larger λ means the more sparse solution. The process of solving Eqs. (3)–(6) is usually referred to as **sparse coding**.

4 SRM for time series prediction

In this section, the framework based on SRM for time series prediction is detailed, and its implementation process is similar to the signal reconstruction based on sparse representation. The process of signal reconstruction can be divided into the following three steps: 1) construct an appropriate over-complete dictionary D; 2) obtain the sparse representation α of signal based on dictionary; 3) $D \times \alpha$ is used to reconstruct the signal. In a sense, the second step can be considered as a coding process, and the third step corresponds to the decoding process. Considering that the future values in the time series have some correlation with the historical values, we believe that if the historical values can be well reconstructed based on a dictionary, then the future values can be well predicted by using the dictionary associated with it. Inspired by this, we split the dictionary into two parts, the upper part for training and the lower part for prediction. It's important to emphasize that if the dictionary is not split into these two parts, the sparse representation algorithm is only able to reconstruct the historical signal, but unable to predict the future signal. By flexibly adjusting the number of rows in the lower part of the dictionary, SRM can be used not only for one-step prediction but also for multi-step prediction. In addition, the dictionary of SRM can be augmented with columns, for example by adding atoms that reflect cross-domain influence factors, to achieve multivariable time series prediction. To sum up, the greatest advantage of SRM is that it can flexibly implement different types of time series predictions by using different dictionaries.

The general problem of time series prediction is described as follows: given a time series $T_1 = [t_1, t_2, \ldots, t_n]^{\mathrm{T}}$, the goal is to predict the values of $T_2 = [t_{n+1}, t_{n+2}, \ldots, t_{n+h}]^{\mathrm{T}}$, where h is the desired prediction horizon. In order to accomplish this goal, from the perspective of series connection, the over-complete dictionary $D \in R^{(n+h)\times m}$ $(m \gg n+h)$ can be divided into two parts, a decomposition part $D_1$and a prediction part D_2. Firstly, based on the decomposition part D_1, the sparse representation of the time series T_1 can be obtained by using the sparse coding technique. Then, the sparse representation of T_1 is projected onto the prediction part of the dictionary to compute the desired forecast for T_2. The implementation of SRM is discussed below in more details, including three stages.

Stage 1: Dictionary construction The dictionary $D \in R^{(n+h)\times m}$ is constructed according to an appropriate strategy, and then it is divided into two parts according to the length of historical data and prediction horizon. The first part of D, called the decomposition part $D_1 \in R^{n\times m}$, consists of the first n rows of the dictionary. The second part of D, called the prediction part $D_2 \in R^{h\times m}$, consists of the remaining h rows of the dictionary.

Stage 2: Sparse coding Firstly, the time series T_1 is normalized to $T_1' = (t_1', t_2', \ldots, t_n')^{\mathrm{T}}$ by using the l_2-norm, and its norm $l_2(T_1)$ is saved. The relevant formulas are defined as follows:

$$t_i' = t_i / l_2(T_1) \quad i = 1, 2, \ldots, n, \tag{7}$$

$$l_2(T_1) = \sqrt{\sum_{i=1}^{n} t_i^2}. \tag{8}$$

The purpose of the normalization step is to focus on the shape of raw data. Secondly, the optimization problem defined as Eq. (4) is solved by using the sparse coding technique to obtain the sparse representation X of time series T_1' in term of the decomposition part D_1.

Stage 3: Prediction The values of $T_2' = (t_{n+1}', t_{n+2}', \ldots, t_{n+h}')^{\mathrm{T}}$ can be calculated by the product of the prediction part D_2 and the sparse representation X. The prediction horizon h can be a constant greater than or equal to 1. If h is equal to 1, one-step prediction can be achieved, while if h is greater than 1, it means multi-step prediction. Finally, to undo the normalization effect, the values of T_2' have to be multiplied by $l_2(T_1)$ to calculate the final values of $T_2 = (t_{n+1}, t_{n+2}, \ldots, t_{n+h})^{\mathrm{T}}$.

$$t_{n+i} = t_{n+i}' \times l_2(T_1), \quad i = 1, 2, \ldots, h. \tag{9}$$

The framework of SRM for time series prediction is shown in Fig. 2. The following is a detailed description of the implementation of stage 1 and stage 2.

4.1 Dictionary construction

The construction of over-complete dictionary D is very important for the success of SRM because an appropriate dictionary can obtain a higher prediction performance. Generally speaking, there are four strategies for dictionary construction of SRM: 1) basic dictionary; 2) learned dictionary; 3) basic dictionary considering cross-domain dependencies; 4) learned dictionary considering cross-domain dependencies. These four strategies can be selected based on different application scenarios and different complexity requirements.

4.1.1 Basic dictionary

The basic dictionary can be created in two different ways. One way is based on analytic approach and the other way is based on data realization [33]. The main idea of analytic approach is to generate an implicit dictionary with tight frame by formulating a mathematical model. In other words, dictionaries of this type are usually achieved simply and quickly by utilizing a group of fixed transformations which can transform time series from the time domain into other domains in order to extract the main features of the series. Common transformations applied to the low-dimensional signals include Fourier and Wavelet. For example, each element of the dictionary based on discrete fourier transform (DFT) can be defined as follows:

$$d_{pq} = \sqrt{\frac{1}{N}} e^{-j\frac{2\pi}{N}(p-1)(q-1)}, \quad p, q = 1, 2, \ldots, N, \tag{10}$$

where $N = n + h$ is the dimension of the atom. It is obvious that the DFT dictionary is a symmetric square matrix. Different from DFT, discrete cosine transform (DCT) replaces the complex exponential functions with the cosine functions, which can be defined as follows:

$$d_{pq} = \sqrt{\frac{1}{N}} f(q) \cos\left(\frac{(2p-1)(q-1)\pi}{2N}\right), \quad p, q = 1, 2, \ldots, N, \tag{11}$$

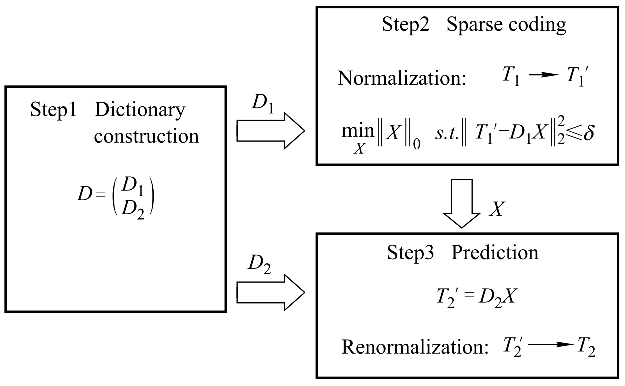
Step1 Dictionary construction
$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}$
D_1
D_2
Step2 Sparse coding
Normalization: $T_1 \to T_1'$
$\min_X \|X\|_0 \;\; s.t. \|T_1' - D_1 X\|_2^2 \leq \delta$
X
Step3 Prediction
$T_2' = D_2 X$
Renormalization: $T_2' \to T_2$

Fig. 2 The framework of SRM for time series prediction

where $f(q) = 1$ when $q = 1$ and $f(q) = \sqrt{2}$ when $q = 2, 3, \ldots, N$. Both DFT and DCT dictionary are orthonormal basis because the columns of the dictionary are unit vectors and orthogonal to each other. However, since DCT can guarantee that every element in the dictionary is non-complex, it is more widely used in practical applications. In addition, considering that much complex time series does not have the characteristic of periodicity, discrete wavelet transform (DWT) makes use of the multi-resolution analysis technique by decomposing the signal through variable-sized windows, which can map the signals into a joint time-scale domain [34]. There are numerous wavelet families that could be used, among which the fastest and easiest to implement is Haar wavelet. It had been proved that the Haar wavelet is an orthonormal basis with compact support [35]. In order to guarantee the over-completeness of the dictionary, the above orthogonal basis can be merged with each other or with the standard orthogonal basis known as *Dirac* dictionary. *Dirac* dictionary is essentially the identity matrix which is the collection of waveforms that all the points are zero except $d_{pq} = 1$ when $p = q$ $(p, q = 1, 2, \ldots, N)$. More recently, some new transformations have been developed to handle the multi-dimensional signals, such as Curvelets, Contourlets, Bandelets, Shearlets, etc [33]. In summary, the highly structured dictionary based on analytic approach can significantly reduce memory requirements, as it only needs to describe the algorithm rather than the full matrix [36].

In contrast, the other way based on data realization infers the dictionary from the original data, which can be called an explicit dictionary. For example, given a time series consisting of historical data $Y = (y_1, y_2, \ldots, y_l)^{\mathrm{T}}$, an over-complete dictionary $D \in R^{(n+h)\times m}$ can be obtained by using sliding window technology to segment historical data [13,14]. The size of the sliding window can be predefined. If the window size is equal to 1, it means that only one element is skipped between two adjacent atoms, and the implementation process is shown below:

$$D = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & \cdots & y_m \\ y_2 & y_3 & \cdots & y_{m+1} \\ \vdots & \vdots & & \vdots \\ y_n & y_{n+1} & \cdots & y_{m+n-1} \\ y_{n+1} & y_{n+2} & \cdots & y_{m+n} \\ \vdots & \vdots & & \vdots \\ y_{n+h} & y_{n+h+1} & \cdots & y_{m+n+h-1} \end{bmatrix}, \tag{12}$$

Note that each column of the decomposition part $D_1 \in R^{n\times m}$ should also be normalized using l_2-norm and the corresponding column of the prediction part $D_2 \in R^{h\times m}$ should be multiplied by the same normalization factor [14]. In many cases, the dictionary based on data realization has the advantage of flexibility and adaptability for the specific signal. However, generating an unstructured dictionary requires more expense which depends on the size of the over-complete dictionary.

4.1.2 Learned dictionary

The basic dictionary has been able to help SRM to achieve one-step prediction and multi-step prediction of time series [13,14]. However, there is a possibility that the basic dictionary after learning may be able to achieve a better result. The target of dictionary learning is to make it more effective for the approximation of the signals by updating the initial dictionary step by step. In order to realize dictionary learning, the remaining historical data can generate a training set $Z = [Z_1, Z_2, \ldots, Z_r] \in R^{(n+h)\times r}$, the training sample Z_i is also generated by sliding window technology. The optimized target function of dictionary learning is can be defined as:

$$\min_{D,X_i} \frac{1}{r} \sum_{i=1}^{r} \|Z_i - DX_i\|, \quad s.t. \|X_i\|_0 \leqslant \varepsilon, \tag{13}$$

where r indicates the number of training samples. The algorithms used to realize the above optimization problem usually implement the alternating update operation of sparse vector and atoms through multiple iterations. The most popular algorithm for dictionary learning is the k-Singular Value Decomposition (k-SVD) algorithm which is a generalization of the k-means algorithm [37]. The k-SVD algorithm requires k iterations in total, and each of iteration consists of two steps: 1) For each training sample Z_i, the sparse representation X_i is obtained according to the current dictionary; 2) The atoms in the dictionary are updated according to the non-zero entries in the sparse representation. Note that only one atom is updated at a time and only those signals in Z whose sparse representations use the current atom are used to participate in the optimization process. This ensures that a dictionary is found under strict sparse constraints after k iterations, which can minimize the reconstruction error for each sample of the training set. Both the atom and its corresponding coefficient row are updated simultaneously via singular value decomposition with the result of accelerated convergence.

It should be noted that dictionary learning can effectively reduce the reconstruction errors and lead to better prediction performance at the cost of increased computational complexity.

4.1.3 Basic dictionary considering cross-domain dependencies

Since dictionary learning leads to increased computational complexity, other approaches need to be considered to improve the prediction performance of SRM without sacrificing computational complexity. In this paper, a novel dictionary considering cross-domain dependencies is proposed for SRM from the perspective of parallel connection, which takes the form $D = [U, V]$. More specifically, U is a basic dictionary and V is an extended dictionary consisting of a set of atoms which represent the cross-domain influence factors related to time series. The motivation for considering cross-domain dependencies is that the results of time series prediction based on historical data alone are always unsatisfactory in many cases. For instance, accurate prediction of power load in smart grid is still a thorny problem because there are many load variables that affect the system highly uncertain with nonlinear fluctuations. These variables include weather, temperature, season, day of the week, hour of the day, and so on [9,38]. According to our previous work, the operation of adding influence factors of time series to the untrained basic dictionary can further improve the performance of SRM.

4.1.4 Learned dictionary considering cross-domain dependencies

The only difference between this strategy and the third one is that it considers cross-domain dependencies based on the learned dictionary rather than the basic dictionary. So the dictionary D should be rewritten as $[U', V]$. The implementation of V is the same as the third strategy, and U' is the basic dictionary after learning. This strategy is expected to combine the benefits of dictionary learning and cross-domain dependencies, but it is also prone to over-fitting.

To sum up, the following three questions need to be considered when constructing the dictionary. Firstly, whether the basic dictionary is generated by means of analytic approach or data realization? Secondly, whether the basic dictionary needs to be learned? Finally, whether the cross-domain influence factors related to time series need to be considered? Different choices of these three questions can form eight variants of SRM, as shown in Table 1.

Table 1 Eight variants of SRM

Abbr.	Generation of Basic Dictionary		Dictionary Learning		Cross-Domain Dependencies	
	AA	DR	No	Yes	No	Yes
SRM-A	√		√		√	
SRM-D		√	√		√	
SRM-AL	√			√	√	
SRM-DL		√		√	√	
SRM-AC	√		√			√
SRM-DC		√	√			√
SRM-ALC	√			√		√
SRM-DLC		√		√		√

Note: AA and DR correspond to the abbreviations of analytic approach and data realization respectively

4.2 Sparse coding

As described in Fig. 2, after constructing the appropriate dictionary, the second stage of SRM is sparse coding. Due to the fact that sparse representation with l_0-norm minimization is an NP-hard problem [39], it is difficult to find the global optimal solution in a reasonable time. Many suboptimal methods have been developed to alleviate this difficulty. From the perspective of sparse penalty, the methods of sparse coding can be roughly divided into the following three categories: 1) l_0-norm minimization; 2) l_1-norm minimization; 3) l_p-norm ($p \neq 0$ and 1) minimization. At present, many different strategies have been proposed to solve different norm minimization [11]. Due to limited space, three of these strategies are briefly described below.

4.2.1 Greedy strategy for l_0-norm minimization

It is well known that the most common strategy for solving an NP-hard problem is the greedy strategy. Therefore, a lot of literature focused on how to utilize the greedy strategy to obtain the approximate solution of sparse representation with l_0-norm minimization. The greedy strategy is a method of sequential selection of basis vectors, which chooses the best local optimal solution step by step until the termination criterion of iteration is satisfied. One of the representative methods which adopted the greedy strategy is matching pursuit (MP) algorithm [40]. The principle of MP is to seek the best atom from the dictionary $D \in R^{n\times m}$ to approximate the signal $T \in R^n$ by minimizing representation residual in each iteration. More specifically, for the kth iteration, the best matching atom $d_{g(k)}$ and the corresponding representation residual R_k can be formulated as:

$$d_{g(k)} = \arg\max |\langle R_{k-1}, d_i\rangle|, \quad i = 1, 2, \ldots, m, \tag{14}$$

$$R_k = R_{k-1} - \langle R_{k-1}, d_{g(k)}\rangle d_{g(k)}, \quad R_0 = T, \tag{15}$$

where $g(k)$ is the label index of the selected column.

One drawback of MP is although asymptotic convergence is guaranteed, more iterations are necessary if the selected atom $d_{g(k)}$ is not orthogonal to the residual R_k. A modification of MP is called as Orthogonal Matching Pursuit (OMP) algorithm which modifies the formula of the residual R_k to ensure the full orthogonality between the residual and the selected atoms at every step [41]. To be specific, after selecting a new atom $d_{g(k)}$ using Eq. (14), OMP utilizes the standard least squares technique to calculate the best approximation over all the selected atoms and then calculates the residual R_k as follows:

$$D_k = D_{k-1} \cup d_{g(k)}, \quad D_0 = \varphi, \tag{16}$$

$$\widetilde{X} = \arg\min \|T - D_k X\|_2^2, \tag{17}$$

$$R_k = T - D_k\widetilde{X}. \tag{18}$$

Obviously, the update process of OMP is more expensive than MP, but the benefit is that fewer iterations are required because OMP does not select the same atom twice due to orthogonality.

Recently, greedy iterative algorithms mainly focus on how to improve the reconstruction performance or reduce the complexity, such as compressive sampling MP (CoSaMP) [42], stagewise OMP (StOMP) [43], subspace pursuit [44] and forward-backward pursuit (FBP) [45].

4.2.2 Convex relaxation strategy for l_1-norm minimization

Considering the l_0-norm is a non-convex and highly discontinuous function, convex relaxation strategy aims to replace it with l_1-norm. It had been proved that if the representation X is sparse enough, the l_0-norm minimization is equal to the l_1-norm minimization [46]. In other words, the number of non-zero entries in the representation X can be approximated by the absolute sum of them. According to the above idea, Eqs. (3)–(6) can be transformed to solve the following problems:

$$\min_X \|X\|_1, \quad s.t.\ T = DX. \tag{19}$$

$$\min_X \|X\|_1, \quad s.t.\ \|T - DX\|_2^2 \leqslant \delta, \tag{20}$$

$$\min_X \|T - DX\|_2^2, \quad s.t.\ \|X\|_1 \leqslant \tau, \tag{21}$$

$$\min_X \frac{1}{2}\|T - DX\|_2^2 + \lambda\|X\|_1, \tag{22}$$

where $\|.\|_1$ and τ are respectively defined as the sum of the absolute values of non-zero entries in vector X and the maximum value of the sum. The advantage of convex relaxation is that the l_1-norm minimization is a convex optimization problem which has a unique solution and can be solved in polynomial time. Due to the equivalence of the l_1-norm minimization with Linear Programming (LP), one of the typical approaches for Eq. (19) is basis pursuit (BP) which advocates the simplex method or interior-point method of LP to obtain an optimal solution. As a variation of BP, basis pursuit denoising (BPDN) can be adapted to noisy data by solving an optimization problem like Eq. (22) using quadratic programming techniques [47]. In addition, there are extensive methods for the l_1-norm minimization including least angle regression (LAR) [48], gradient projection (GP) [49], iterative shrinkage thresholding algorithm (ISTA) [50], and a number of their variants.

4.2.3 Nonconvex strategy for l_p-norm minimization

Although l_1-norm minimization has been widely used and been accepted as a very effective strategy for solving sparse representation, there is still the risk of inconsistent selection because the l_1-norm imposes a heavier penalty on larger coefficients of the representation X, while the l_0-norm imposes the same penalty on all nonzero coefficients [51]. Recently, more and more studies have focused on solving the sparse representation problem with the l_p-norm minimization, especially $0 < p < 1$ [52]. The objective function of the l_p-norm minimization can be rewritten as follows:

$$\min_X \|X\|_p^p, \quad s.t.\ \|T - DX\|_2^2 \leqslant \delta, \tag{23}$$

or

$$\min_X \frac{1}{2}\|T - DX\|_2^2 + \lambda\|X\|_p^p, \tag{24}$$

where $\|X\|_p^p = \sum_{i=1}^m |x_i|^p$.

The l_p-norm minimization ($0 < p < 1$) is a nonconvex and nonsmooth optimization problem which is difficult to be solved effectively [53]. Moreover, it is also a problem to choose which p will produce the best result. In recent years, the representativeness of $l_{1/2}$ minimization is highlighted in the study of l_p-norm minimization [54], and some effective methods have been proposed to solve this problem such as iterative reweighted method (IRM) [55], iterative reweighted least squares (IRLS) [56], and iterative thresholding method (ITM) [57].

5 Experiments and results

In this section, two aspects of experiments are carried out on two publicly available datasets for time series prediction. The experiments in the first aspect mainly focus on eight variants of SRM performed on the first dataset, while the experiments in the second aspect compare the performances of SRM and other popular models for time series prediction based on both the first and the second datasets.

5.1 Datasets

Here is a brief overview of the two datasets. The first dataset that we analyze is the time series of electricity consumption from the European network of intelligent technologies (EUNITE) competition organized by the slovak eastern power company in August 2001 [58]. The EUNITE competition dataset not only provides electricity load recorded every half hour from January 1997 to January 1999 but also contains cross-domain influence factors including daily mean temperature and holiday information. To further validate the importance of cross-domain influences, we added the weekend information to provide additional contextual information to support the prediction task. The specification of the EUNITE competition dataset used in this paper is described in Table 2. The second dataset is a large-scale time series dataset provided by the M3 competition which consists of 3003 time series covering different sampling period of month, quarter, and year [59]. In order to facilitate the comparison with other related work [60,61], the 1045 monthly time series with the length between 99 and 144 are selected from the M3 competition data for the experiment. They come from different categories and have different data dimensions. The specification of the M3 competition dataset used in this paper is described in Table 3.

Table 2 The specification of the EUNITE competition dataset used in this paper

Attributes	Time interval	Dimension
Maximum daily load	1997.1.1–1999.1.31	761
Daily mean temperature	1997.1.1–1999.1.31	761
Holiday information	1997.1.1–1999.1.31	761
Weekend information	1997.1.1–1999.1.31	761

Table 3 The specification of the M3 competition dataset used in this paper

Category	Time interval	Quantity	Dimension
Demographic	Monthly	90	132–138
Financial	Monthly	123	99–144
Industry	Monthly	333	122–144
Macro	Monthly	300	107–144
Micro	Monthly	197	126
Other	Monthly	2	120

5.2 Experimental results of different variants of SRM on the EUNITE competition dataset

The key to the success of SRM lies in the construction of the over-complete dictionary. Since this paper mainly focuses on the performance comparison of eight variants of SRM generated under different dictionary construction strategies, the following experiments uniformly adopt OMP algorithm written by Matlab to solve the sparse coding, and the sparse factor is selected from the given interval for optimization. The size of the over-complete dictionary constructed by different variants of SRM is different from each other. On the whole, dictionaries of the variations that require learning are smaller than those that do not need because some historical data needs to be reserved as training sets. From the perspective of fairness principle, the historical data used to complete the entire dictionary construction process is the same regardless of whether the variation requires dictionary learning. The mean absolute percentage error (MAPE) is utilized as an accuracy criterion, which can be defined as:

$$\text{MAPE} = \frac{1}{h}\sum_{i=1}^{h}\frac{|t_i - \bar{t}_i|}{t_i} \times 100\%, \tag{25}$$

where h is the prediction horizon, t_i and $\bar{t}_i$ are the actual and the predicted value of the ith moment.

5.2.1 Experiments of SRM-A on the EUNITE competition dataset

According to the description in Table 1, SRM-A employs an analytic approach to construct a basic dictionary without learning and considering cross-domain factors. In the first, we merge DCT, DFT and DWT dictionaries with *Dirac* dictionary respectively to obtain three over-complete dictionaries. Secondly, we utilize these three different dictionaries for time series prediction according to the workflow of SRM shown in Fig. 2. For a more objective comparison, the experiment of SRM-A makes use of the data from January 1, 1997 to the day before the predicted month as the training set to predict the maximum daily load of the next month. Depending on the predicted month, 13 experiments are performed from January 1998 to January 1999 respectively. The experimental results are shown in Table 4. For a more intuitive display of the results, we visualize the results as Fig. 3. It can be concluded from Fig. 3 that SRM-A based on DCT is a good choice for this dataset because it has maintained stable performance and small prediction error in all experiments. Because of the obvious periodicity of power load data, DWT does not have much advantage in this dataset.

Table 4 MAPE (%) of SRM-A on the EUNITE competition dataset

Training data	Test data	DCT	DFT	DWT
97.01–97.12	98.01	**3.60**	4.00	4.11
97.01–98.01	98.02	**1.57**	3.79	3.79
97.01–98.02	98.03	**3.69**	3.70	3.70
97.01–98.03	98.04	**7.01**	7.07	7.05
97.01–98.04	98.05	**5.40**	14.44	9.62
97.01–98.05	98.06	**3.34**	13.14	5.74
97.01–98.06	98.07	5.01	9.11	**3.06**
97.01–98.07	98.08	**3.03**	17.09	**3.03**
97.01–98.08	98.09	**3.55**	5.74	6.09
97.01–98.09	98.10	4.95	**3.21**	7.16
97.01–98.10	98.11	3.68	**3.07**	5.08
97.01–98.11	98.12	3.89	**3.21**	4.23
97.01–98.12	99.01	**2.79**	4.75	2.94

5.2.2 Experiments of SRM-D on the EUNITE competition dataset

In terms of dictionary construction, the only difference between SRM-D and SRM-A is that it follows the way of data realization to create an over-complete dictionary. It should be noted that the dictionary construction of SRM-D is related to the length of historical data. According to the requirements of the competition, the experiments will take the data of 1997 and 1998 as historical data to predict the maximum daily load in January 1999. Since the dictionary $D \in R^{(n+h)\times m}$ only requires more columns than rows, there are many different sizes of dictionaries produced based on 730 points of two years. Considering that prediction horizon $h = 31$, after deducting the data of n days before the predicted month used for sparse coding, the number of columns m can be calculated according to the following formula:

$$m = 730 - n - (n + 31) + 1. \qquad (26)$$

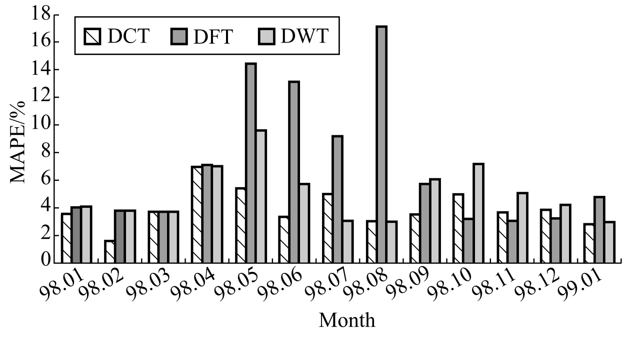

Fig. 3 Prediction performance comparison of SRM-A for different basic dictionaries on the EUNITE competitive dataset

For a full comparison, we arrange five experiments with n = 90, 120, 150, 180, and 210. According to the value of n and Eq. (12), five dictionaries of different sizes are created, including 121×520, 151×460, 181×400, 211×340, and 241×280. After constructing the dictionary, the sparse representation X can be obtained based on the data of n days previously reserved and then the prediction is made. The experimental results of SRM-D are shown in the second column of Table 5.

Table 5 MAPE (%) of four variants of SRM-D based on the EUNITE competition dataset, where D stands for the dictionary based on data realization

Size of D	SRM-D	SRM-DL	SRM-DC	SRM-DLC
121×520	2.69	2.18	1.72	1.74
151×460	2.69	1.94	**1.61**	1.43
181×400	2.02	**1.68**	1.94	**1.42**
211×340	2.22	1.90	1.98	1.62
241×280	**1.96**	1.69	1.62	1.49

According to the last row of Table 4, the MAPE of the same prediction problem solved by SRM-A based on DCT is **2.79%**, which is higher than the results of SRM-D based on different dictionaries. The best result of SRM-D is **1.96%** when the size of the dictionary is set to 241×280. The fact that SRM-D is superior to SRM-A for this dataset further demonstrates that explicit dictionaries are more flexible and adaptable than implicit dictionaries.

5.2.3 Experiments of SRM-AL and SRM-DL on the EUNITE competition dataset

SRM-AL adds the process of dictionary learning based on SRM-A. In view of the previous experiments on SRM-A, we directly choose DCT merged with *Dirac* dictionary to form the basic dictionary. We conduct five experiments to predict the maximum daily load in January 1999 respectively, and the size of the training set gradually increases from 20 to 100. The generation of the training sample is related to the size of the training set. If the training set has a total of r samples, each training sample $Z_i = \{t_i, t_{i+1}, \ldots, t_{730-r+i}\}$ $(i = 1, 2, \ldots, r)$. The method of dictionary learning is performed using the k-SVD toolbox V10 and the experimental results of SRM-AL are shown in the first row of Table 6. It is obvious that SRM-AL performs better than SRM-A (MAPE=**2.79%**) in the vast majority of cases. Especially when the number of time series involved in dictionary learning r=100, SRM-AL can achieve the best prediction performance (MAPE=**1.41%**).

Similar to SRM-AL, SRM-DL adds dictionary learning based on SRM-D. We maintain the dictionaries of SRM-DL consistent with those of SRM-D in order to facilitate performance comparisons. Five dictionaries of different sizes have different training sets. For dictionary $D \in R^{(n+h)\times m}$, the number of training samples $r = n$, each training sample $Z_i = \{t_{m+i}, t_{m+i+1}, \ldots, t_{m+i+n+h-1}\}$ $(i = 1, 2, \ldots, r)$. The experimental results of SRM-DL are shown in the third column of Table 5. By comparing the MAPE values before and after the dictionary learning, it can be concluded that this strategy can also effectively improve the prediction performance of SRM-D. The best performance of SRM-DL is **1.68%**, which is worse than that of SRM-AL (MAPE=**1.41%**), indicating that dictionary learning is more helpful for implicit dictionary constructed by analytic approach.

5.2.4 Experiments of considering cross-domain dependencies on the EUNITE competition dataset

Due to the fact that this dataset provides three cross-domain influence factors for the time series of power loads, it is convenient to compare the prediction performances of different variations with or without influence factors. The cross-domain influence factors include daily mean temperature, dates of the holidays, and day of the weeks. We mark holidays and weekends as 1, and the rest of the working days as 0. After normalizing all cross-domain influence factors, they are merged with different dictionaries of different varieties. We performed the contrast experiments on all the above models to verify the importance of cross-domain dependencies. For the problem of power load forecasting in January 1999, the MAPE of SRM-A based on DCT is **2.79%**, while the MAPE of SRM-AC considering cross-domain dependencies is **2.26%**. The experimental results of SRM-DC are shown in the fourth column of Table 5. Obviously, considering cross-domain dependencies can also be effective in improving the prediction performance of SRM-A and SRM-D. The same conclusion applies to SRM-AL and SRM-DL. The experimental results of SRM-DLC and SRM-ALC are shown in the last column of Table 5 and the last row of Table 6. For better comparison, we divide the eight variants into four groups based on whether cross-domain dependencies are considered. The prediction performance comparisons of the best results are shown in Fig. 4. It can be concluded from Fig. 4 that when the dataset provides cross-domain influence factors that affect the changing trend of time series, it should be recommended to add atoms of influence factors to the over-complete dictionary in order to further improve the prediction performance.

Table 6 MAPE (%) of SRM-AL and SRM-ALC on the EUNITE competition dataset, where r represents the number of time series involved in dictionary training

Abbr.	r=20	r=40	r=60	r=80	r=100
SRM-AL	3.72	2.29	1.52	1.76	**1.41**
SRM-ALC	1.87	1.69	1.44	1.41	**1.40**

Finally, to better compare the overall performance of the eight variations of SRM, Table 7 lists their optimal MAPE values and corresponding running time. We can draw the following conclusions from Table 7:

1) For SRM-A, dictionary learning can improve prediction performance more effectively than considering cross-domain dependencies, but at the cost of high computational complexity.

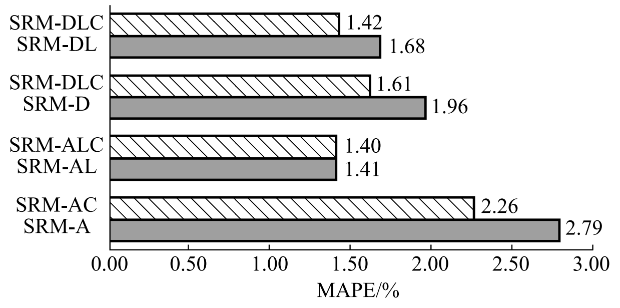

Fig. 4 Prediction performance comparison of eight variants of SRM on the EUNITE competition dataset

2) For SRM-D, a similar effect can be achieved if either dictionary learning or cross-domain dependencies are considered, while the combination of the two can further improve the predictive performance.

Table 7 The overall performance of eight variants of SRM on the EUNITE competition dataset

	SRM-A	SRM-AC	SRM-AL	SRM-ALC
MAPE/%	2.79	2.26	1.41	1.40
Time/s	0.05	0.05	1.01	1.01
	SRM-D	SRM-DC	SRM-DL	SRM-DLC
MAPE/%	1.96	1.61	1.68	1.42
Time/s	0.27	0.27	0.47	0.47

5.3 Contrast experiments between SRM and other time series prediction models

In order to show the strength of SRM for time series prediction, the contrast experiments are carried out on two different datasets.

5.3.1 Experimental results of the EUNITE competition dataset

In accordance with the requirements of the competition in that year, the experiments will take maximum daily loads of 1997 and 1998 as historical data to predict that of January 1999. According to the performance comparison of the eight variants of SRM shown in Table 7, we choose SRM-DC, SRM-DLC, and SRM-ALC to compare with some popular models of time series prediction, including MLP, SVR, RF, ARIMA, LSTM, GRU, and MGU.

The MLP method using a three-layer neural network is constructed exploiting the nnet function of the R statistical package. The number of the hidden nodes is set to $2n+1$ following the practical guidelines, where n is the number of the input nodes, that is, the number of cross-domain influence factors. The activation functions of the hidden layer and output layer adopt logistic function and linear function respectively. Since the initial weights are generated randomly, the experiment is performed 10 times to obtain the optimal result. The SVR method using ε-regression is constructed exploiting the SVM function of the e1071 R statistical package. The kernel used is the radial basic one, which is mainly due to its good performance and fewer required parameters. The insensitive loss ε is set to 0.1 and the cost of constraints violation is fixed to the maximum of the target output values, which is 1. The RF method using ensemble learning is constructed with default parameters exploiting the RF function of the R statistical package. Similar to MLP, the performance of random forest varies greatly due to the initial parameters. The experiment is also carried out 10 times to obtain the optimal result. The ARIMA method is constructed exploiting the forecast function of the R statistical package. The parameters in ARIMA are fitted via maximum likelihood and the model orders (p, d, q) are finally selected as (2, 1, 3), where p denotes the order of autoregressive, d is express differencing, and q is the specific order of the moving average. LSTM, GRU, and MGU are implemented in Python, using Keras library as the backend. We adopt the adaptive learning rate optimization algorithm Adam to calculate the gradient over mini-batches of the training data. Since different parameter settings can lead to different results, the optimization is performed by assigning to each hyperparameter a value selected from a given interval. As the prediction horizon h=31, the time step of the network is also set to 31. We train the network for 3000 epochs with the learning rate $\lambda = 10^{-3}$ and the optimized configuration is finally set as: the number of the hidden units N_h=30, the size of the batch B_S=15.

The experimental results in Table 8 show that the accuracy of SRM-ALC, SRM-DLC, and SRM-DC ranks first, second and fourth respectively without sacrificing running time. Among them, although SRM-DC is not as accurate as SRM-ALC and SRM-DLC, it has the shortest running time due to the lack of dictionary learning. To be clear, LSTM is also very suitable for time series prediction because it can capture both long-term and short-term time dependencies between data. However, as LSTM adopts a relatively complex gate structure, a large number of connection weights need to be trained, resulting in high computational complexity of the model. As the simplified versions of LSTM, GRU, and MGU reduce the running time, but their accuracy is worse than that of LSTM. In summary, SRM only needs to learn the basic dictionary or supplement the cross-domain influence factors to achieve time series prediction quickly and efficiently. Considering the tradeoff of accuracy and complexity, SRM has advantages over other models. Figure 5 shows more detail between the real value of the raw time series and the predicted value of all models. Although all the models reflect the periodic change of the raw time series, the prediction performance is quite different. More obviously, ARIMA and RF do not show a slow upward trend similar to the raw values, while the other models can well fit them. LSTM does not fit well in the first half, but the second half is excellent. SRM-ALC and SRM-DLC have the best fitting effect in the whole prediction horizon.

Table 8 The prediction performance of different models on the EUNITE competition dataset

Model	MAPE/%	Time/s	Rank
ARIMA	4.08	42.5	10
RF	3.44	0.53	9
SVR	3.09	0.11	8
MLP	2.75	0.08	7
MGU	2.20	3986	6
GRU	1.91	2521	5
SRM-DC	1.61	0.27	4
LSTM	1.53	4949	3
SRM-DLC	1.42	0.47	2
SRM-ALC	**1.40**	1.01	1

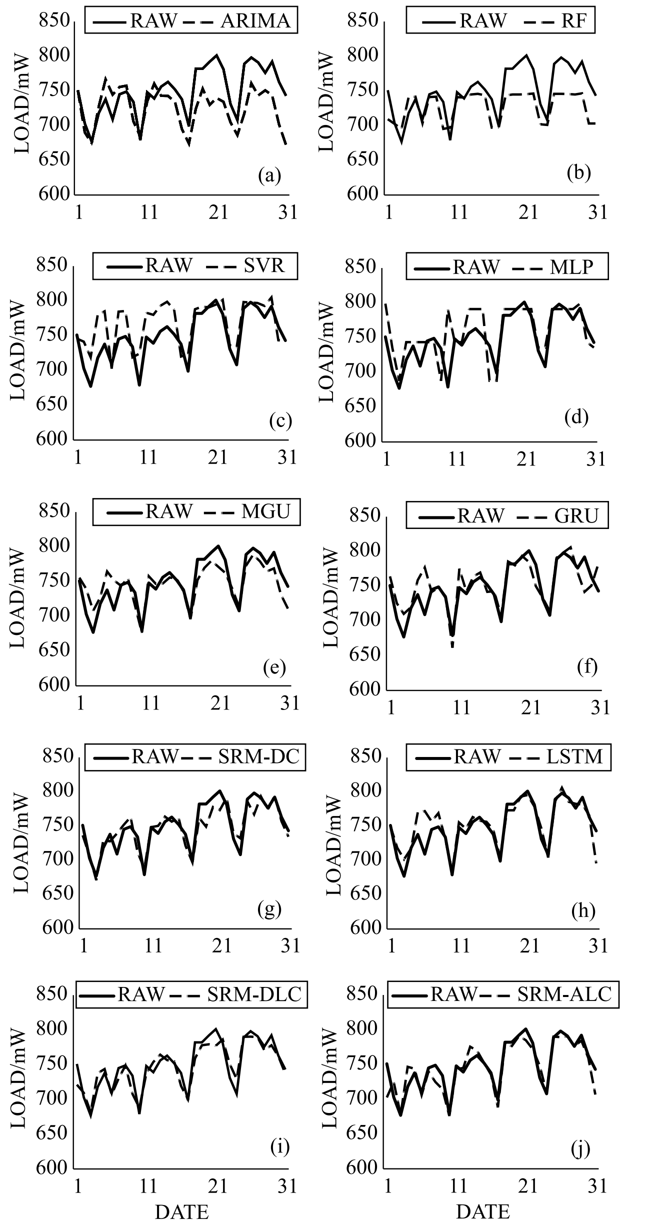

Fig. 5 Prediction performance comparison of different models on the EUNITE competition dataset. (a) The predicted load of ARIMA vs. the raw load; (b) The predicted load of RF vs. the raw load; (c) The predicted load of SVR vs. the raw load; (d) The predicted load of MLP vs. the raw load; (e) The predicted load of MGU vs. the raw load; (f) The predicted load of GRU vs. the raw load; (g) The predicted load of SRM-DC vs. the raw load; (h) The predicted load of LSTM vs. the raw load; (i) The predicted load of SRM-DLC vs. the raw load; (j) The predicted load of SRM-ALC vs. the raw load

5.3.2 Experimental results of the M3 competition dataset

For the M3 competition dataset, it does not provide cross-domain influences factors. It is characterized by a large-scale dataset consisting of 1045 time series with the length between 99 and 144. All models are trained with the first n-18 observations, where n is the length of the series, and then 18 predictions are generated and compared with the actual values to evaluate their performance. According to the computational complexity comparison of eight variants of SRM shown in Table 7, as the simplest one, SRM-A has the advantages of the parameter-free adjustment and the lowest running time, which makes it particularly suitable for this dataset. The work performed in [60,61] utilized the exact same dataset to compare the performance of three categories of time series prediction models, including traditional statistical models, machine learning models, and hybrid integrated models. So it is convenient to compare the results of SRM-A with the above models on this basis. Considering that the work performed in [60,61] adopted symmetric mean absolute percentage error (SMAPE) as an accuracy criterion, SRM-A is evaluated with the same performance metric for reasons of fairness. SMAPE is called the adjusted MAPE and is defined as:

$$\text{SMAPE} = \frac{1}{h}\sum_{i=1}^{h}\frac{|t_i - \bar{t}_i|}{(|t_i + \bar{t}_i|)/2} \times 100\%. \qquad (27)$$

Table 9 shows the SMAPE of different horizons by different models. The first four models belong to traditional statistical models. Naïve 2 is a kind of random walk model adjusted for seasonality, which is often used as a benchmark of the M3 competition. The second and third models are ARIMA and automatic exponential smoothing (AES), both of which are very popular statistical forecasting models. The fourth model is the Theta method that achieved the best overall prediction performance in the original M3-competition. The selection of machine learning models participating in the contrast experiment is based on the conclusion that MLP and BNN are superior to other machine learning models for one-step prediction [61]. There are usually three methods to extend the horizon of machine learning models from one-step prediction to multi-step prediction, including iterative method, direct method, and multi-output method [62]. The iterative method involves recursively using the one-step model where the prediction of the prior time step is used as an input for making a prediction on the next time step. That means the new prediction depends on the accuracy of the previous ones such that performance can quickly degrade as the forecasting horizon increases. On the contrary, the direct method produces a separate model for each forecast time step. Finally, the multi-output method is to develop one model that is capable of predicting the entire forecast horizon at one time. Obviously, the direct method is not suitable for large-scale datasets due to its complexity. Thus, only the other two methods are applied to MLP and BNN for multi-step prediction. Finally, the ninth model is a hybrid integrated model that combines three exponential smoothing models: SES, Holt, and Damped, based on the principle of averaging multiple forecast results to achieve better accuracy. A detailed introduction of the above models can be found in [60].

Table 9 SMAPE (%) of different horizons by different models on the M3 competition dataset

Model	h=1	h=5	h=9	h=18	Mean
Results from [60]					
Naïve 2	10.75	10.38	13.24	16.49	12.72
ARIMA	7.78	9.84	11.27	16.03	11.23
AES	8.06	9.94	11.09	15.45	11.14
Theta	8.20	9.68	10.73	14.86	10.87
MLP Iterative	7.98	10.75	12.03	17.20	11.99
BNN Iterative	7.92	10.54	11.73	16.82	11.75
MLP Multi	7.87	10.88	12.25	18.38	12.35
BNN Multi	7.95	10.74	12.38	18.25	12.33
Comb S-H-D	8.18	9.70	10.87	14.99	10.94
Our results					
SRM-A	**3.66**	**7.66**	**9.76**	**14.19**	**8.82**

As can be seen from Table 9, the top three with the best average prediction performance are SRM-A, Theta and Comb S-H-D, while machine learning models perform poorly. The main reason is that the traditional statistical model has a stable prediction performance in predicting a large number of time series one by one. However, the prediction performance of machine learning model fluctuates greatly, resulting in the decrease of the average value. In the case of the overall poor performance of machine learning models, SRM-A fully achieves the best performance of different prediction horizons. This fully demonstrates the effectiveness of SRM for time series prediction.

6 Conclusions and future work

In smart city, huge amounts of sensing information are captured every second by diverse IoT devices. These information change with time and have some regulations, which can be predicted with the help of data mining technology in order to realize the event alert. In this paper, a systematic framework based on sparse representation model for time series prediction (SRM) is proposed as a novel approach to face the challenge. SRM divides the dictionary into two parts from the perspective of the series connection. The decomposition part is used to obtain the sparse representation of historical data, while the prediction part combined with sparse representation can be used for the forecast. To enable SRM to be flexibly applied to various types of time series prediction, this paper focuses on the study of dictionary construction strategy and summarizes eight variants of SRM. The difference between the eight variants lies in the choice of three questions, including: how the basic dictionary is generated, whether dictionary learning is required, and whether cross-domain dependencies are considered. By comparing these eight variants in experiments, we got the following insights:

1) For the prediction of single time series, SRM-D with explicit dictionary constructed by data realization is more flexible and adaptable than SRM-A with implicit dictionary constructed by analytic approach. However, the way of data realization to generate an over-complete dictionary depends on the length of the historical time series, while the analytic approach is more universal and convenient.

2) Dictionary learning can effectively improve the prediction performance of SRM, but it will increase the computational complexity.

3) The prediction performance of SRM can be effectively improved by taking cross-domain influence factors into account, with the benefit of no additional computational complexity.

4) As the simplest of eight variants of SRM, SRM-A has the advantages of the parameter-free adjustment and the lowest computational complexity, which makes it particularly suitable for the large-scale dataset.

For future work, we are interested in two aspects. On the one hand, the integration of analytic approach and data realization can be studied to generate an over-complete dictionary so as to take the advantages of both implicit dictionary and explicit dictionary. On the other hand, Kernel Sparse Representation Model (KSRM) for time series prediction with nonlinear features is a primary focus, because KSRM has the opportunity to learn more discriminative sparse representation than SRM and greatly boosts the performances of nonlinear time series prediction.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 61772136, 61672159), the Technology Innovation Platform Project of Fujian Province (2014H2005), the Research Project for Young and Middle-aged Teachers of Fujian Province (JT180045), the Fujian Collaborative Innovation Center for Big Data Application in Governments, the Fujian Engineering Research Center of Big Data Analysis and Processing.

References

1. Zheng Y. Urban computing: enabling urban intelligence with big data. Frontiers of Computer Science, 2017, 11(1): 1–3
2. Su K, Li J, Fu H. Smart city and the applications. In: Proceedings of International Conference on Electronics, Communications and Control. 2011, 1028–1031
3. Yi F, Yu Z, Chen H, Du H, Guo B. Cyber-physical-social collaborative sensing: from single space to cross-space. Frontiers of Computer Science, 2018, 12(4): 609–622
4. Yu Z, Yi F, Lv Q, Guo B. Identifying on-site users for social events: mobility, content, and social relationship. IEEE Transactions on Mobile Computing, 2018, 17(9): 2055–2068
5. Yu Z, Wang H, Guo B, Gu T, Mei T. Supporting serendipitous social interaction using human mobility prediction. IEEE Transactions on Human-Machine Systems, 2015, 45(6): 811–818
6. Yu Z, Yu Z, Chen Y. Multi-hop mobility prediction. Mobile Networks and Applications, 2016, 21(2): 367–374
7. Yu Z, Xu H, Yang Z, Guo B. Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints. IEEE Transactions on Human-Machine Systems, 2015, 46(1): 151–158
8. Wang L, Yu Z, Guo B, Ku T, Yi F. Moving destination prediction using sparse dataset: a mobility gradient descent approach. ACM Transactions on Knowledge Discovery from Data, 2017, 11(3): 1–33
9. Yu C N, Mirowski P, Ho T K. A sparse coding approach to household electricity demand forecasting in smart grids. IEEE Transactions on Smart Grid, 2016, 8(2): 738–748
10. Fu T C. A review on time series data mining. Engineering Applications of Artificial Intelligence, 2011, 24(1): 164–181
11. Zhang Z, Xu Y, Yang J, Li X, Zhang D. A survey of sparse representation: algorithms and applications. IEEE Access, 2015, 3: 490–530
12. Fakhr M W. Online nonstationary time series prediction using sparse coding with dictionary update. In: Proceedings of IEEE International Conference on Information and Communication Technology Research. 2015, 112–115
13. Rosas-Romero R, Díaz-Torres A, Etcheverry G. Forecasting of stock return prices with sparse representation of financial time series over redundant dictionaries. Expert Systems with Applications, 2016, 57: 37–48
14. Helmi A, Fakhr M W, Atiya A F. Multi-step ahead time series forecasting via sparse coding and dictionary based techniques. Applied Soft Computing, 2018, 69: 464–474
15. De Gooijer J G, Hyndman R J. 25 years of time series forecasting. International Journal of Forecasting, 2006, 22(3): 443–473
16. Dijk D, Teräsvirta T, Franses P H. Smooth transition autoregressive models—a survey of recent developments. Econometric Reviews, 2002, 21(1): 1–47
17. Anand N C, Scoglio C, Natarajan B. GARCH—non-linear time series model for traffic modeling and prediction. In: Proceedings of IEEE Network Operations and Management Symposium. 2008, 694–697
18. Bontempi G, Taieb S, Borgne Y L. Business Intelligence. Berlin, Springer, 2013
19. Ahmed N K, Atiya A F, Gayar N E, Shishiny H E. An empirical comparison of machine learning models for time series forecasting. Econometric Reviews, 2010, 29(5–6): 594–621
20. Kumar M, Thenmozhi M. Forecasting stock index movement: a comparison of support vector machines and random forest. In: Proceedings of the 9th Capital Markets Conference on Indian Institute of Capital Markets. 2006, 1–16
21. Längkvist M, Karlsson L, Loutfi A. A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, 2014, 42: 11–24
22. Zheng J, Xu C, Zhang Z, Li X. Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network. In: Proceedings of the 51st IEEE Annual Conference on Information Sciences and Systems. 2017, 1–6
23. Fu R, Zhang Z, Li L. Using LSTM and GRU neural network methods for traffic flow prediction. In: Proceedings of the 31st IEEE Youth Academic Annual Conference of Chinese Association of Automation. 2016, 324–328
24. Zhou G B, Wu J, Zhang C L, Zhou Z H. Minimal gated unit for recurrent neural networks. International Journal of Automation and Computing, 2016, 13(3): 226–234
25. Guo W, Li J, Chen G, Niu Y, Chen C. A PSO-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks. IEEE Transactions on Parallel and Distributed Systems, 2014, 26(12): 3236–3249
26. Ye D, Chen Z. A new approach to minimum attribute reduction based on discrete artificial bee colony. Soft Computing, 2015, 19(7): 1893–1903
27. Qiu X, Zhang L, Ren Y, Suganthan P N, Amaratunga G. Ensemble deep learning for regression and time series forecasting. In: Proceedings of IEEE Symposium on Computational Intelligence in Ensemble Learning. 2014, 1–6
28. Zhang G P. Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 2003, 50: 159–175
29. Duan Z, Yang Y, Zhang K, Ni Y, Bajgain S. Improved deep hybrid networks for urban traffic flow prediction using trajectory data. IEEE Access, 2018, 6: 31820–31827
30. Donoho D L. Compressed sensing. IEEE Transactions on Information Theory, 2006, 52(4): 1289–1306
31. Lewicki M S, Sejnowski T J. Learning overcomplete representations. Neural Computation, 2000, 12(2): 337–365
32. Wang S, Guo W. Sparse multigraph embedding for multimodal feature representation. IEEE Transactions on Multimedia, 2017, 19(7): 1454–1466
33. Rubinstein R, Bruckstein A M, Elad M. Dictionaries for sparse representation modeling. Proceedings of the IEEE, 2010, 98(6): 1045–1057
34. Batal I, Hauskrecht M. A supervised time series feature extraction technique using DCT and DWT. In: Proceedings of IEEE International Conference on Machine Learning and Applications. 2009, 735–739
35. Stanković R S, Falkowski B J. The Haar wavelet transform: its status and achievements. Computers & Electrical Engineering, 2003, 29(1): 25–44
36. Qayyum A, Malik A S, Naufal M, Saad M, Mazher M, Abdullah F, Abdullah T A R B T. Designing of overcomplete dictionaries based on DCT and DWT. In: Proceedings of IEEE Student Symposium in Biomedical Engineering & Sciences. 2015, 134–139
37. Aharon M, Elad M, Bruckstein A. K-SVD: an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing, 2006, 54(11): 4311–4322
38. Ertugrul Ö F. Forecasting electricity load by a novel recurrent extreme learning machines approach. International Journal of Electrical Power & Energy Systems, 2016, 78: 429–435
39. Natarajan B K. Sparse approximate solutions to linear systems. SIAM Journal on Computing, 1995, 24(2): 227–234
40. Mallat S G, Zhang Z. Matching pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing, 1993, 41(12): 3397–3415
41. Pati Y C, Rezaiifar R, Krishnaprasad P S. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: Proceedings of the 27th IEEE Asilomar Conference on Signals, Systems and Computers. 1993, 40–44
42. Needell D, Tropp J A. CoSaMP: iterative signal recovery from incomplete and inaccurate samples. Applied and Computational Harmonic Analysis, 2009, 26(3): 301–321
43. Donoho D L, Tsaig Y, Drori I, Starck J L. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. IEEE Transactions on Information Theory, 2012, 58(2): 1094–1121
44. Dai W, Milenkovic O. Subspace pursuit for compressive sensing signal reconstruction. IEEE Transactions on Information Theory, 2009, 55(5): 2230–2249

45. Karahanoglu N B, Erdogan H. Compressed sensing signal recovery via forward–backward pursuit. Digital Signal Processing, 2013, 23(5): 1539–1548
46. Donoho D L. For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution. Communications on Pure and Applied Mathematics, 2006, 59(6): 797–829
47. Chen S S, Donoho D L, Saunders M A. Atomic decomposition by basis pursuit. SIAM Review, 2001, 43(1): 129–159
48. Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression. The Annals of Statistics, 2004, 32(2): 407–499
49. Figueiredo M A T, Nowak R D, Wright S J. Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. IEEE Journal of Selected Topics in Signal Processing, 2007, 1(4): 586–597
50. Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences, 2009, 2(1): 183–202
51. Qiu K, Dogandzic A. Variance-component based sparse signal reconstruction and model selection. IEEE Transactions on Signal Processing, 2010, 58(6): 2935–2952
52. Guo S, Wang Z, Ruan Q. Enhancing sparsity via l_p $(0 < p < 1)$ minimization for robust face recognition. Neurocomputing, 2013, 99: 592–602
53. Chartrand R. Exact reconstruction of sparse signals via nonconvex minimization. IEEE Signal Processing Letters, 2007, 14(10): 707–710
54. Zeng J, Xu Z, Zhang B, Hong W, Wu Y. Accelerated $L_{1/2}$ regularization based SAR imaging via BCR and reduced Newton skills. Signal Processing, 2013, 93(7): 1831–1844
55. Foucart S, Lai M J. Sparsest solutions of underdetermined linear systems via l_q-minimization for $0 < q \leqslant 1$. Applied and Computational Harmonic Analysis, 2009, 26(3): 395–407
56. Daubechies I, DeVore R, Fornasier M, Güntürk C S. Iteratively reweighted least squares minimization for sparse recovery. Communications on Pure and Applied Mathematics, 2010, 63(1): 1–38
57. Xu Z, Chang X, Xu F, Zhang H. $L_{1/2}$ regularization: a thresholding representation theory and a fast solver. IEEE Transactions on Neural Networks and Learning Systems, 2012, 23(7): 1013–1027
58. Chen B J, Chang M, Lin C J. Load forecasting using support vector machines: a study on EUNITE competition 2001. IEEE Transactions on Power Systems, 2004, 19(4): 1821–1830
59. Makridakis S, Hibon M. The M3-Competition: results, conclusions and implications. International Journal of Forecasting, 2000, 16(4): 451–476
60. Makridakis S, Spiliotis E, Assimakopoulos V. The accuracy of machine learning (ML) forecasting methods versus statistical ones: extending the results of the M3-Competition. Working Paper, University of Nicosia, Institute for the Future, Greece, 2017
61. Makridakis S, Spiliotis E, Assimakopoulos V. Statistical and machine learning forecasting methods: concerns and ways forward. PloS One, 2018, 13(3): e0194889
62. Taieb S B, Atiya A F. A bias and variance analysis for multistep-ahead time series forecasting. IEEE Transactions on Neural Networks and Learning Systems, 2015, 27(1): 62–76

Zhiyong Yu is an associate professor at College of Mathematics and Computer Science, Fuzhou University, China, also affiliated with Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, and Key Laboratory of Spatial Data Mining and Information Sharing, Ministry of Education, China. He received his PhD from Northwestern Polytechnical University, China in 2011. He was a visiting student at Kyoto University, Japan from 2007 to 2009 and a visiting researcher at TELECOM SudParis, France from 2012 to 2013. His current research interests include pervasive computing, mobile social networks, and crowd sensing.

Xiangping Zheng received the BS degree in computer science and technology from Xiamen University of Technology, China in 2015. Currently, he is a postgraduate student in College of Mathematics and Computer Science, Fuzhou University, China. His current research interests include data mining and machine learning.

Fangwan Huang is a senior lecturer at College of Mathematics and Computer Science, Fuzhou University, China. She received the BS and MS degrees in computer science from Fuzhou University, China in 2002 and 2005. Currently, she is a PhD candidate in College of Physics and Information Engineering, Fuzhou University, China. Her research interests include computational intelligence, big data analysis, and so on.

Wenzhong Guo received the BS and MS degrees in computer science and the PhD degree in communication and information system from Fuzhou University, China in 2000, 2003, and 2010, respectively. He completed the postdoctoral fellow at Institute of computer Science, National University of Defense and Technology, China in 2013, and senior visiting scholar at Faculty of Engineering, Information and System, University of Tsukuba, Japan in 2013. He is a professor and dean of College of Mathematics and Computer Science, Fuzhou University. He is also a member of ACM and IEEE. His current research interests include VLSI physical design, wireless sensor networks, big data, image processing, and so on.

Lin Sun received the BS degree in communication engineering in 2001 and MS degree in computer science in 2004 from East China University of Science and Technology, China. He received PhD degree in computer science in 2010 from Zhejiang University, China. Now he is the association professor of Zhejiang University City College, China. His research interests include pattern recognition and pervasive computing.

Zhiwen Yu is a professor and the vice-dean of the School of Computer Science, Northwestern Polytechnical University, China. He received the PhD degree in computer science from Northwestern Polytechnical University, China in 2006. He was a Alexander Von Humboldt fellow with Mannheim University, Germany, and a research fellow with Kyoto University, Japan. His research interests include ubiquitous computing and social network analysis. He is a senior member of the IEEE.