

# Structural health monitoring by using a sparse coding-based deep learning algorithm with wireless sensor networks

Junqi Guo · Xiaobo Xie · Rongfang Bie ·  
Limin Sun

Received: 20 February 2014 / Accepted: 29 April 2014 / Published online: 21 August 2014  
© Springer-Verlag London 2014

**Abstract** Structural health monitoring has received remarkable attention due to the arising structural safety problems. Most of these structural health problems are accumulative damages such as slight changes in structural deformations which are very hard to be detected. In addition, the complexity of real structure and environmental noises make structural health monitoring more difficult. Existing methods largely use various types of sensors to collect useful parameters and then train a machine learning model to diagnose damage level and location, in which a large amount of training data are needed for the model training, while the labeled data are rare in the real world. To overcome this problem, sparse coding is employed in this paper to achieve structural health monitoring of a bridge equipped with a wireless sensor network, so that a large amount of unlabeled examples can be used to train a feature extractor based on the sparse coding algorithm. Features learned from sparse coding are then used to train a neural network classifier to distinguish different statuses of the bridge. Experimental results show the sparse coding-based deep learning algorithm achieves higher accuracy for

structural health monitoring under the same level of environmental noises, compared with some existing methods.

**Keywords** Structural health monitoring · Sparse coding · Wireless sensor network

## 1 Introduction

Structural deterioration is a growing problem both in China and around the world. For instance, bridges easily suffer from persistent traffic, wind loading, material aging, environmental corrosion, earthquakes and so on. All these factors can result in structural deficiencies and damages, which greatly shorten the lifetime of structures. Sometimes these imperceptible damages may cause serious security incidents such as collapse and subsidence, which may lead to significant casualties and loss of properties. For example, in August 2012, the Yangmingtan bridge in Harbin city collapsed with several cars falling down and three people died. Another serious accident occurred in Hunan province of China, the Tuojiang bridge under construction suddenly collapsed, causing 64 workers died. Many facts and experiences show that the continuously structural health monitoring is extremely necessary to avoid such incidents happen.

To solve this problem, great attention has recently been paid to structural health monitoring (SHM) techniques [1–5]. The early researches of SHM focus on real physical models trying to mimic the status of a real structure, which is called model-driven method [6, 7]. This method uses mathematical modeling and physical laws to represent the monitored structure. By analyzing and solving the model, the degree and location of the damage place can be accurately detected. However, when the complexity of the

---

J. Guo · X. Xie · R. Bie (✉)  
Beijing Normal University, Beijing, People's Republic of China  
e-mail: rfbie@bnu.edu.cn

J. Guo  
e-mail: guojunqi@bnu.edu.cn

X. Xie  
e-mail: to\_xiexiaobo@163.com

L. Sun  
Beijing Key Laboratory of IOT Information Security  
Technology, Institute of Information Engineering, CAS,  
Beijing, People's Republic of China  
e-mail: sunlimin@iie.ac.cn

monitored structure grows as well as the environmental factors are taken into consideration, building and solving such a complex model become much more difficult. Since the mid-1990s, model-driven method has been gradually replaced by a kind of new approach named data-driven method [8–15], in which wireless sensor networks (WSN) [16–19] accompanied with machine learning [20] are employed for better data collection and processing in SHM. For instance, Worden et al. [21, 22] take an experiment on laboratory structures by using novel detection algorithms such as outlier analysis and auto-associative neural network. Hoon Sohn [23] also applies time series analysis combined with auto-regressive and outlier analysis to identify different structural conditions of a fast patrol boat. Since physical data of engineering structures collected by wireless sensors are intelligently collected and analyzed by these data-driven methods, structural health problems hidden in raw data can be promptly detected and remaining lifetime of architectures may be predicted with less cost of time and labor.

The intuition behind data-driven approaches for SHM is simple: When there are some damages occurred in a structure, properties of the structure may change, which could be reflected in sensor data. By extracting features from these raw data, a classifier can be built to distinguish different statuses of the structure. Therefore, a SHM task is converted into a classification problem which can be solved by machine learning-based algorithms such as neural network [24–30] and support vector machine. There are many advantages of the machine learning-based approaches for structural health monitoring. First, it is very suitable for complex structures because their analysis on structure rely on the data collected by sensor, not the model self. Second, this kind of method can automatically learn the damage degree and location according to large amounts of data. However, performance of such methods heavily depends on the size of training data, while obtaining enough labeled data for training brings high cost of time and manpower which may limit performance of traditional supervised learning algorithms. Another factor that may affect classification performance is feature selection. For traditional supervised learning algorithms, suitable features should be selected from raw data according to engineering experience and professional knowledge, which makes feature selection a challenging task.

To overcome these problems, we seek to apply a sparse coding-based deep learning algorithm to build a feature extractor from unlabeled data. Since Geoffrey Hinton proposed a new method in which a deep “autoencoder” network is trained to learn low-dimensional codes from high-dimensional input vectors in 2006 [31], deep learning approaches have gained great interests. So far, deep learning has beaten many state-of-the-art algorithms in a

wide range of areas. Hinton et al. use deep neural networks (DNNs) for acoustic modeling in speech recognition and outperform GMM-HMMs model which has already been widely used in most current speech recognition systems [32]. In image classification field, Andrew Ng [33] builds a nine-layer network to learn a high-level feature detector from unlabeled images which outperforms most of other existing methods. Deep learning techniques have also been extensively studied in natural language processing (NLP) and achieved many breakthroughs [34–37]. Although deep learning has been widely studied in many applications, as far as we know, seldom literatures have introduced deep learning techniques into SHM.

In deep learning, sparse coding [37] provides an efficient way to find succinct representations of unlabeled data. It learns basis functions which capture high-level features in the data, making classification tasks much easier and more accurate. In this paper, we employ a sparse coding-based deep learning algorithm to achieve structural health monitoring of a bridge equipped with a wireless sensor network. The wireless sensor network system is responsible for data collecting. After data preprocessing stage, we perform sparse coding to learn valid feature representations from only unlabeled data. These feature representations will be taken as the input of neural network to classify different statuses of structure. The contribution of this paper is that deep learning techniques such as sparse coding are firstly introduced in SHM applications.

The rest of this paper is organized as follows: in Sect. 2, we will discuss data collection, data preprocessing, feature extraction and the theory of sparse coding in detail. In Sect. 3, experiments setting and results will be given to demonstrate the efficiency of our approach. Finally, a short conclusion will be drawn in Sect. 4.

## 2 System design

Basically, the architecture of our system design consists of three main layers. The first layer is responsible for data preparation, including data collection and data preprocessing. The second layer is the key layer in which feature extraction is performed, including sparse coding module and modal analysis module. The last layer is the classification layer, in which we adopt neural network as the classification algorithm. Figure 1 describes our system design.

### 2.1 Data collection

To validate the proposed algorithm in this paper, we choose a three-span bridge to monitor its health status. As is shown in Fig. 2, wireless sensors are allocated in joints

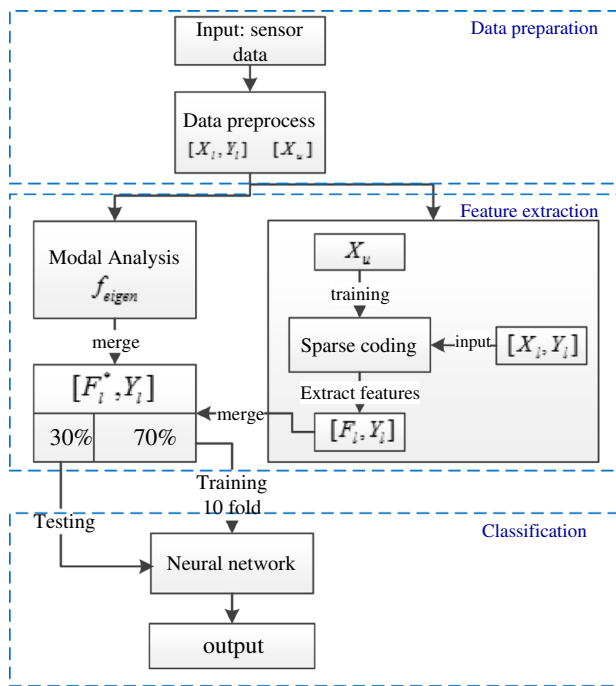


Fig. 1 System design for structural health monitoring

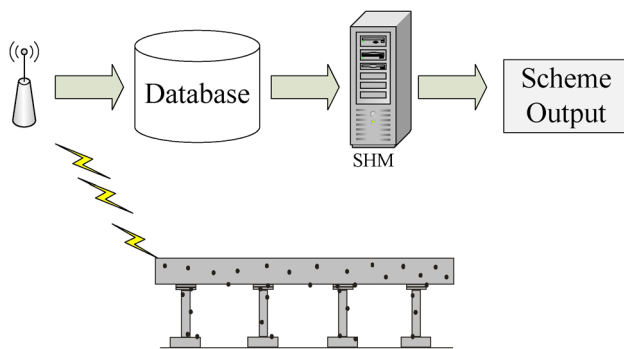


Fig. 2 Structural health monitoring of a bridge

and some key parts of the bridge. These sensors will measure acceleration data in a constant interval and store them.

Data collected by each sensor can be denoted as a vector  $D_i = (d_1, d_2, \dots, d_i, \dots)$ . Our algorithm runs the data in database every day, so that a report about the current health situation can be given based on the analytical results.

### 2.2 Data preprocessing

Generally, data collected by each sensor can be represented by an unlimited-dimensional vector or a time series. In data preprocessing stage, these time series are cut into small pieces by using a time frame, as shown in Fig. 3. Suppose there are  $r$  sensors attached in the bridge, the time frame

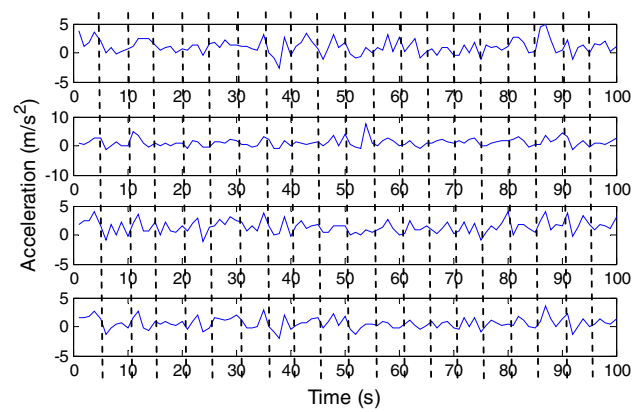


Fig. 3 Data preprocessing

size is  $t$ , and sampling frequency is  $f$ . The number of data pieces in one time frame is  $r$ . If we concatenate these pieces of data into a vector  $x = (p_1, p_2, \dots, p_r) \in \mathbb{R}^{r \times t \times f}$ , along with a class label  $y \in \{1, \dots, C\}$ , where  $C$  denotes the number of category, then one training example  $\{x, y\}$  is formed. Repeatedly do the same procedure described above, we obtain labeled training set of  $m$  examples  $\{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \dots, (x_l^{(m)}, y^{(m)})\}$ . When the health status of the bridge is unknown, we can also construct a set of  $k$  unlabeled examples  $x_u^{(1)}, x_u^{(2)}, \dots, x_u^{(k)} \in \mathbb{R}^{r \times t \times f}$ .

### 2.3 Modal analysis

The second layer is the feature extraction layer. Basically, this layer consists of two steps. In the first step, a sparse coding algorithm is applied to learn high-level features from raw acceleration data. We will talk about this topic later in the next subsection. While in the second step, we use built-in solver function  $fe\_eig$  provided by SDT—to obtain modal frequencies as the complementary features.

The  $fe\_eig$  function returns the eigendata—including both mode shapes and natural frequencies—in a structured matrix with fields  $.def$  for shapes and  $.DOF$  to code the DOFs (Degree of freedom) of each row in  $.def$  and  $.data$  giving the modal frequencies in Hz. This function can be call with the following form:

$$Eigopt = [SolutionMethod nm \dots];$$

$$Def = fe\_eig(model, eigopt);$$

the function parameter  $model$  is a matlab structure which holds the bridge we construct, and  $eigopt$  is an array which describes related parameters with this function. We choose Lanczos solver as the solution method because it is more suitable for complex models. Parameter  $nm$  represents the number of mode shapes we need. In addition, we also need to define load and boundary conditions, materials and section properties and sensors before using this function.

When a structure get damaged, some properties of this system will also change. Most of the time, these changes will be directly reflected in the modal frequencies of this structure. In other word, the modal frequencies are strong indicators for the health statuses of structures. In this paper, we perform the modal analysis to obtain the modal frequencies for each status of the structure, and then these frequencies are merged into the feature vector as a part of features.

### 2.4 Sparse coding

Another module in the second layer is the sparse coding module. As we have mentioned in the previous section, since the labeled data are rare, a method which can make full use of large amounts of unlabeled data and automatically capture features from input data is preferred. Sparse coding, which is an unsupervised feature learning algorithm, feeds all the requirements above. Sparse coding was first proposed by Olshausen and Field [38], which originally used as an unsupervised computational model of low-level sensory processing in human beings. Here is the architecture of a sparse coding:

Sparse coding consists of three layers: an input layer, a hidden layer and an output layer. Each neuron has weights connected to all neurons in the next layer. Given an unlabeled training example set  $\{x^{(1)}, x^{(2)}, x^{(3)}, \dots\}$ , where  $x^{(i)} \in \mathbb{R}^n$ , sparse coding employs the backpropagation algorithm, setting the target values to be equal to the inputs, which means that the sparse coding tries to learn an identity function  $h_{w,b}(x) \approx x$ . If we add a constraint on the network to limit the number of hidden units, the network is forced to learn a compress representation of the input. We can also reconstruct the input data as similar as possible by using the learned features. In practical, we do not limit the number of hidden units; instead, we impose a sparsity constraint on the hidden units to limit the number of “active” units. Informally, if the output of a neuron is close to 1, we consider it as being “active,” otherwise, it is “inactive.” What we want is to constrain the neurons in hidden layer to be inactive in most of the time (Fig. 4).

Suppose that  $a_j^{(2)}(x)$  denotes the activation of hidden unit  $j$ , given a specific input  $x$ . In forward propagation process, the activation of hidden layer can be denoted as:  $a^{(2)} = \text{sigmoid}(W^{(1)}x)$ .  $W^{(1)}$  is the weight between input layer and hidden layer. So the average activation of hidden unit  $j$  can be given by:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^m [a_j^{(2)}(x^{(i)})] \tag{1}$$

We would like to let  $\hat{\rho}_j$  be close to a small value  $\rho$ , which is called the sparsity parameter. To achieve this, we add an extra term to the objective function that penalizes  $\hat{\rho}_j$

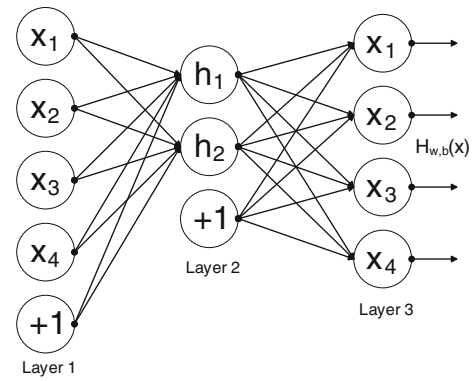


Fig. 4 Architecture of sparse coding

deviating significantly from  $\rho$ . We choose KL divergence as our penalty term:

$$\sum_{j=1}^{L_2} \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \tag{2}$$

Recall that the cost function of neural network can be defined as follows:

$$C(W, b) = \left[ \frac{1}{m} \sum_{i=1}^m \left( \frac{1}{2} \|h_{w,b}(x^{(i)}) - y^{(i)}\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{m_l-1} \sum_{i=1}^{m_l} \sum_{j=1}^{m_{l+1}} (w_{ij}^{(l)})^2 \tag{3}$$

So the cost function for sparse coding can be modified as below:

$$C_{sparse}(W, b) = C(W, b) + \beta \sum_{j=1}^{L_2} KL(\rho || \hat{\rho}_j) \tag{4}$$

where  $KL(\rho || \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}$ .

In order to find the optimal parameters of sparse coding, we need to minimize  $C_{sparse}(W, b)$  as a function of  $W$  and  $b$ . Batch gradient descent is a proper choice. Each iteration of gradient descent updates the parameters  $W, b$ :

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \beta \frac{\partial}{\partial w_{ij}^{(l)}} C(W, b) \tag{5}$$

$$b_i^{(l)} = b_i^{(l)} - \beta \frac{\partial}{\partial b_i^{(l)}} C(W, b) \tag{6}$$

The backpropagation algorithm can compute the partial derivatives efficiently. However, for sparse coding training, it will be slightly different from the backpropagation algorithm. We need to compute a forward pass on all training examples to compute the average activation  $\hat{\rho}_j$ . Then, a second forward pass will be conducted to do backpropagation on training examples. Here is the sparse coding algorithm:

**Algorithm:** Backpropagation for sparse coding

**Input :** an training example  $(x, y)$

1: Random initialize  $W$  and  $b$

2: Perform forward propagation to compute  $\hat{\rho}_j$

3: **for** unit  $i$  in output layer **do**

$$4: \delta_i^{(n_i)} = \frac{\partial}{\partial z_i^{(n_i)}} \frac{1}{2} \|h_{w,b}(x) - y\|^2 = -(y_i - a_i^{n_i}) \cdot f'(x)$$

5: **end for**

6: **for**  $l = n_i - 1, \dots, 2$  **do**

7: **for** each neuron  $i$  in layer  $l$  **do**

$$8: \delta_i^{(l)} = \frac{\partial C}{\partial a_i^{(l)}} \cdot \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} \\ 9: = \left( \sum_{j=1}^{m_{l+1}} \delta_j^{(l+1)} w_{ij}^{(l)} + \beta \left( -\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j} \right) \right) \cdot f'(z_i^{(l)})$$

10: **end for**

11: **end for**

12: compute the partial derivatives.

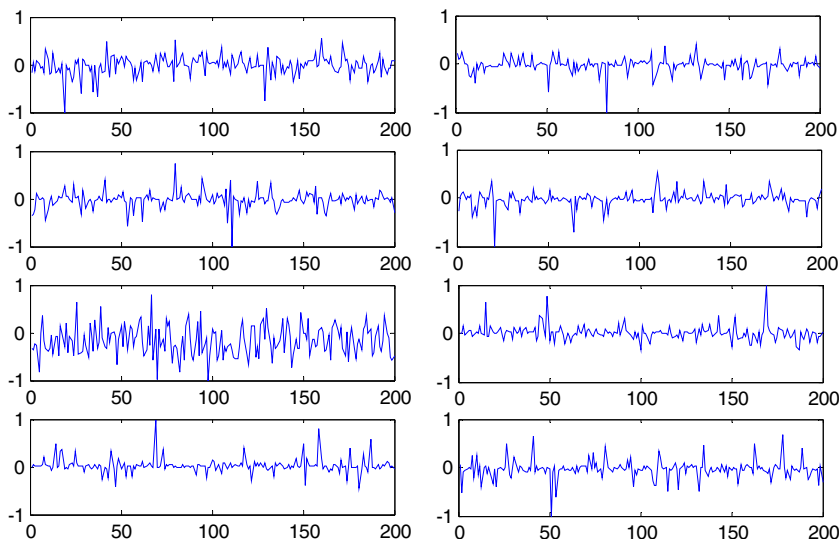
**Output :**

$$13: \frac{\partial}{\partial w_{ij}^{(l)}} C(W, b) = \delta_j^{(l+1)} \alpha_i^{(l)}$$

$$14: \frac{\partial}{\partial b_i^{(l)}} C(W, b) = \delta_i^{(l+1)}$$

Once we have trained a sparse coding, it can learn high-level features from unlabeled data. To try to understand what it has learned, we visualize the features captured by hidden units. Notice that  $W^{(1)} \in \mathbb{R}^{m \times n}$  denotes the weight between input layer and hidden layer, and  $i$ th row of  $W^{(1)}$  represent the parameters for  $i$ th hidden unit. Take  $i$ th row of  $W^{(1)}$  as the input of sparse coding, the activation of  $i$ th hidden unit will be maximal. In Fig. 5, we randomly

**Fig. 5** Features learned by eight hidden units from input data



choose 8 units from 156 hidden units to illustrate features learned from input data. This figure shows some basic patterns or features learned by hidden units.

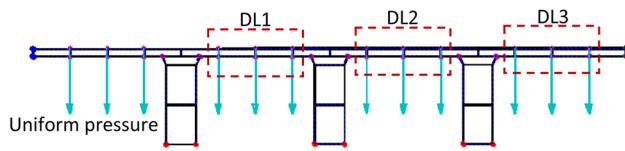
After building a sparse coding to extract features using unlabeled data, a classifier can be constructed to make predictions for the statuses of structures. So far, there are many classification algorithms available; here, we choose a neural network to build a classifier in consideration of its high performance and stability. Given a set of training examples, sparse coding takes these data as input and extract features  $\{f_i^{(1)}, f_i^{(2)}, \dots, f_i^{(m)}\}$ . Then, these features along with the corresponding class labels will be used to train a neural network. Similarly, for testing examples, we also use sparse coding to extract features and employ neural network to predict the class label of testing examples.

### 3 Experiment

#### 3.1 Presentation of structure

The structure considered is a three-span bridge which is presented in Fig. 6. This bridge is constructed using the Structural Dynamics Toolbox (SDT [39]) under Matlab with 150 nodes and 192 elements. The surface of the bridge is set to be 80 meters long and 8 meters wide, and it has two lanes in opposite direction. The material of the bridge is steel. The Young’s modulus and shear modulus of these materials are assumed to be 210 GPa and 80 GPa, respectively [40]. Table 1 lists some main material properties used in our experiment. However, if the bridge has been damaged or corroded, both of these two parameters will decline according to the degree of damage or corrosion. How these two material factors change will be





**Fig. 6** Three-span bridge with different damage locations

**Table 1** Material properties of steel

Symbol	Value	Unit	Physical quantity
E	210000000000	Pa	Young's modulus
Nu	0.285		Poisson's ratio
Rho	7800	Kg/m <sup>3</sup>	Density
G	81712062256	Pa	Shear modulus
Eta	0		Loss factor
Alpha	0	/°C	Thermal expansion coef
T0	20	°C	Reference temperature

explained in detail later on. The system is excited by a uniform pressure acting on whole surface of the bridge. The bridge's motion is restricted to in plane vibrations. As shown in Fig. 6, the left and right edges of the bridge surface are fixed as well as the bottom of three piers.

### 3.2 Experiment setting

In order to monitor the status of the bridge in real time, about 36 triaxial accelerometers are allocated on the upper and bottom surfaces of the bridge, as well as joint places between bridge surface and piers. The sample frequency of these sensors is set to be 5 Hz. Each of the sensors records 5 acceleration data every 1 s at the sensor's location. So during the period of monitoring, the data record by one sensor can be regarded as a one-dimensional vector and data collected by total 36 sensors form a matrix. These sensor data are our raw data and can be used in feature extraction latter.

To quantify the damage degree of a bridge and differentiate different statuses of a bridge conveniently, we define four kinds of scenarios: a healthy scenario and three damage scenarios. The differences between these scenarios are Young's modulus and shear modulus of material at some predefined locations of the bridge, which are DL1, DL2 and DL3 (see Fig. 6). For convenience, we also use d1, d2 and d3 to denote three damage scenarios. In the healthy scenario, the Young's modulus and shear modulus are declared in Table 1. However, when steel becomes corrupt over time, both Young's modulus and shear modulus will decrease definitely. Tables 2 and 3 describe the Young's modulus and shear modulus reduction at locations

**Table 2** Young's modulus reduction at locations DL1, DL2 and DL3 for three damage scenarios considered

Damage case	DL1 (%)	DL2 (%)	DL3 (%)
d1	3	8.5	5
d2	6	15	10
d3	9	30	20

**Table 3** Shear modulus reduction at locations DL1, DL2 and DL3 for three damage scenarios considered

Damage case	DL1 (%)	DL2 (%)	DL3 (%)
d1	2.5	7.5	5
d2	5	15	10
d3	10	30	20

DL1, DL2 and DL3 for three damage scenarios, respectively.

The condition of a bridge in real world can easily be subjected to environmental factors, such as wind, humidity, temperature and even slight disturbance. Changes in environmental factors will lead to the changes of acceleration data collected by sensors. In order to make the experiment as realistic as possible, we simply assume all the environmental noises obey Gaussian distribution with zero mean and  $\sigma$  standard deviation. The reason for this assumption is that we cannot list all environmental factors and we also cannot tell which factors are dominant ones, so the safest way to analyze is that assuming all the factors obey Gaussian distribution. Based on this assumption, for each sample data, environmental noise is added to the measure data in the following form:

$$\alpha_i(t) = \alpha_i(t) + N(0, \sigma)(t) \quad (7)$$

where  $\alpha_i(t)$  is the acceleration measured at sensor  $i$  and time  $t$  and  $N(0, \sigma)$  is a Gaussian random variable with zero mean and  $\sigma$  standard deviation. In our simulation, we choose two different values of  $\sigma$ , namely 1 and 0.5, to represent two different noise levels, respectively.

### 3.3 Feature extraction

The feature extraction consists of two key steps. In the first step, the sparse coding algorithm is applied to learn high-level features from raw acceleration data. While in the second step, we use built-in solver function *fe\_eig* provided by SDT—to obtain modal frequencies as complementary features.

So far, we have defined four kinds of scenarios and two noise levels for the bridge showed in Fig. 6. For each scenario under certain noise level, we do a simulation for a period of time. There are 36 sensors in total, for each

sensor, the sample frequency is 5 Hz. If time frame size is set to 5 s, then we can obtain 900 data from 36 sensors in a time frame. Actually, this is what we have mentioned in data preprocessing step. According to the method in data preprocessing, these 900 data can form into a vector  $x = (p_1, p_2, \dots, p_r) \in \mathbb{R}^{1 \times 900}$ . In the experiment, the classification label is  $y \in \{1, 2, 3, 4\}$ , which denotes four different scenarios: healthy scenario and three damage scenario d1, d2 and d3. Here, we use  $(x_l^t, y_l^t)$  to represent a labeled example in one time frame. By cutting acceleration data from all sensors during simulation, we can obtain a set of training and testing examples:

$$[X_l, Y_l] = \{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \dots, (x_l^{(t)}, y^{(t)})\} \tag{8}$$

apart from the labeled training and testing data  $[X_l, Y_l]$ , we also need unlabeled training data to train a sparse coding feature extractor. In our experiment, we build several bridges which are similar to the bridge we show in Fig. 6. For each bridge, same scenarios and noise levels are define as we have discussed above. But there is a slight difference this time, we leave out all the labels  $Y$  to get a set of unlabeled data  $X_u = \{x_u^1, x_u^2, \dots, x_u^t\}$ .

The input of sparse coding model training is a set of unlabeled data  $X_u$ , then the sparse coding training algorithm will be performed to learn model parameters  $W$  and  $b$  (weights and bias among neurons). Our sparse coding model consists of three layers, with 900 neurons in the input layer, 156 neurons in the hidden layer and 900 neurons in the output layer. Once sparse coding has been trained, we feed it with labeled data  $X_l$  and perform the feedforward algorithm by using  $W$  and  $b$  to extract features. For each example  $x_l \in X_l$ , the output of hidden layer is the corresponding feature vector  $f_l \in \mathbb{R}^{1 \times 156}$ . By using sparse coding model, we successfully extract features  $F_l \in \mathbb{R}^{m \times 156}$  (suppose there are  $m$  training and testing examples) from labeled data  $X_l \in \mathbb{R}^{m \times 900}$ . For each data  $(x_l, y_l)$ , we use sparse coding to obtain a feature vector  $f_l$  from  $x_l$ , so the class label for  $f_l$  is  $y_l$ , which is taken from the same  $x_l$ . Feature matrix  $F_l$  and class label  $Y_l$  will be further used in the training and testing phase of classification.

The second step of feature extraction is calculating modal frequencies of the bridge. In the experiment, we adopt built-in functions *fe\_eig* in SDT to obtain modal frequencies. In addition, we also need to define load and boundary conditions, materials, section properties and sensors before using this function. For each of four scenarios described above (including one healthy scenario and three damage scenarios), we obtain 20 modal frequencies in total. Table 4 shows modal frequencies under four different scenarios.

As Table 4 shows, the modal frequencies of the bridge under different health statuses are also different. More specifically, the differences in low frequencies are not

**Table 4** Modal frequencies under different scenarios

Health (Hz)	d1 (Hz)	d2 (Hz)	d3 (Hz)
13.529619	13.52883	13.52796	13.52591
13.577989	13.57679	13.57553	13.57278
13.663792	13.66153	13.65922	13.65438
13.796059	13.7941	13.79204	13.78755
13.989928	13.98504	13.98007	13.9698
14.270259	14.26374	14.25718	14.24385
14.678254	14.66927	14.6602	14.64171
15.284126	15.27227	15.26038	15.23649
16.212584	16.19736	16.18205	16.15112
17.698144	17.67868	17.6592	17.62018
20.219783	20.20174	20.18364	20.14725
24.893324	24.86077	24.82812	24.7624
33.887029	33.88702	33.88702	33.88702
34.999965	34.96146	34.9227	34.8443
66.089337	66.01031	65.92995	65.7642
68.100046	68.10002	68.09999	68.09993
102.9711	102.9077	99.04077	90.67022
106.44299	102.9711	101.7434	96.53754
106.52954	104.194	102.8209	98.59637
106.67012	104.8181	102.9722	101.4132

obvious, but there are great differences in high frequencies. With the damage situation of the bridge getting worse, the high modal frequencies tend to decline, as the last rows of Table 4 shows. Thus, we add the modal frequencies to feature vectors and get a new feature matrix  $F_l^* \in \mathbb{R}^{m \times 176}$ .

### 3.4 Training and testing

In previous section, we discuss two steps in feature extraction. So in this section, we will talk about the training and testing of classification. In the experiment settings, we define four category of bridge conditions, and in each situation, we perform data preprocessing and feature extraction to get feature matrix  $F_l^*$  and class label  $Y_l$ . By using these data, we can train a classification model so that we could predict which category of condition the bridge belongs to when a new example comes. We build a three-layer neural network for data classification, with 176 units in input layer, 85 units in hidden layer and 4 units in output layer. All the data are divided into two parts: 70 % for training and 30 % for testing. We also use tenfold cross-validation to find the optimal parameters of classification model. Finally, to evaluate and compare the performance of our algorithm with others, we also implement four algorithms for comparison. They are neural network without sparse coding, logistic regression (LR) [41], softmax regression (SR) [42] and decision tree (DT) [43].

### 3.5 Classification accuracy

Classification accuracy is a popular metric to evaluate the performance of classification algorithms. “Accuracy” reflects the percentage of examples that algorithms guess correctly in the total testing examples. In the experiment, we select 10 different sizes of training set to testify our approach. For each training set, we use 70 % of data for training and the other for testing. We first run our experiment under the condition that contains environmental noise of  $N(0, 1)$ . Figure 7 shows accuracy of all the algorithms with the increase of the number of examples.

As is illustrated in Fig. 7, the test accuracy of all the algorithms goes up as the training set size increases. The accuracy of our algorithm increases dramatically when the training set size below 300 and then goes up steadily as the training set size grows, reaching an accuracy about 98 %. Neural network without sparse coding achieves 96 % and softmax regression approaches to 94 %. The other two methods only achieve an accuracy below 90 %.

Figure 8 compares accuracy of our algorithm with other ones when environmental noise  $N(0, 0.5)$  is added. Because of the impact of noise, the accuracy of neural network drops to about 93 %. However, our algorithm still achieves a relative high accuracy, i.e., 98 %, which indicates that sparse coding can tolerate much more noise than other algorithms. The accuracies of the other four algorithms go down to 85 % or less. Note that the vibration magnitude of a bridge is small; so when the standard deviation of noise is smaller, the noise signal will be more similar to the original signal, which may have a strong interference on the raw data. Thus, compared with the first scenario, traditional algorithms suffer more performance degradation. However, sparse coding shows a better noise tolerance performance than other algorithms. Compared with input data which exist correlations among them, the noise signals are always disorder or irregular, sparse coding can capture these correlations from input data and filter some random noises at the same time; this could explain why sparse coding can tolerate more noises than others methods.

### 3.6 Recall rate

Recall rate is another important metric to evaluate the performance of classification algorithms. We calculate the recall rate under two different environmental noise conditions and compare our algorithm with the other four algorithms. Figure 9 shows the recall rate with noise level  $N(0, 1)$ , while Fig. 10 shows the recall rate with noise level  $N(0, 0.5)$ .

In both of these two figures, our method has achieved acceptable rates, compared with the other approaches. In Fig. 9, although neural network and softmax regression

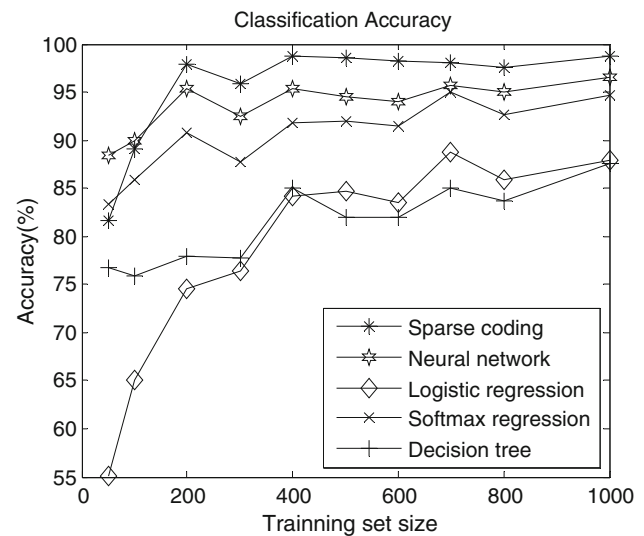


Fig. 7 Classification accuracy with noise level  $\sim N(0, 1)$

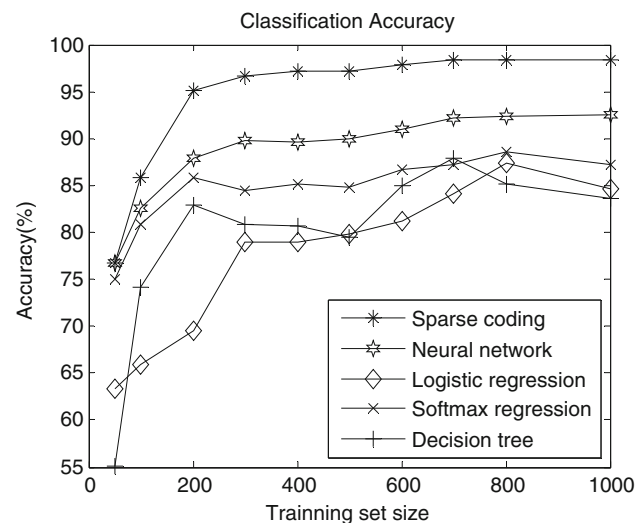


Fig. 8 Classification accuracy with noise level  $\sim N(0, 0.5)$

perform relatively well, the recall rate of our algorithm gets a little bit higher. In Fig. 10, all the algorithms suffer from the stronger noise and the corresponding recall rate drops obviously, except our method. The comparisons between all the approaches under two different noise conditions also show us that the proposed algorithm can achieve a better performance when environmental condition changes.

### 3.7 F1-score

Precision and recall rate are two most commonly metrics to reflect different aspects of performance of classification algorithms. However, there is a trade-off between them. By investigating each of them separately, we are hardly to tell which algorithm is better. Fortunately, F1-score combines



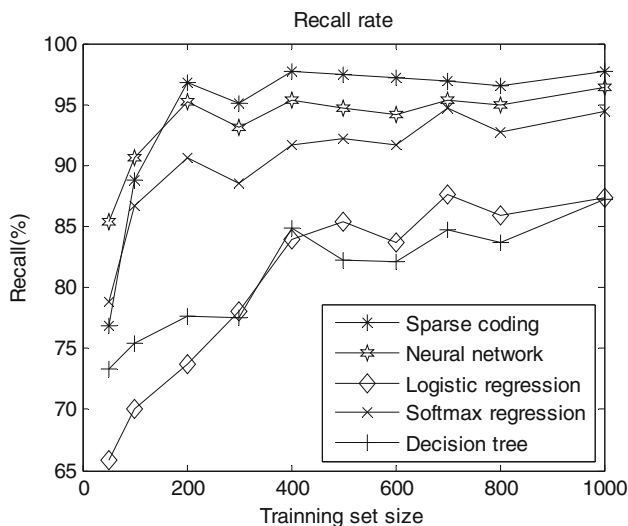


Fig. 9 Recall rate with noise level  $\sim N(0, 1)$

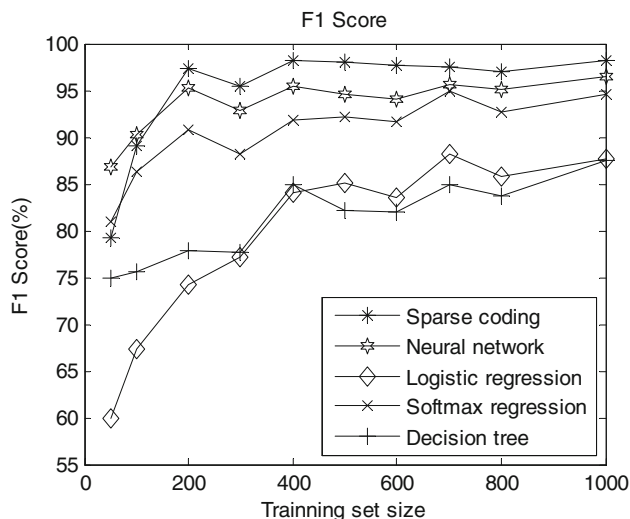


Fig. 11 F1-score with noise level  $\sim N(0, 1)$

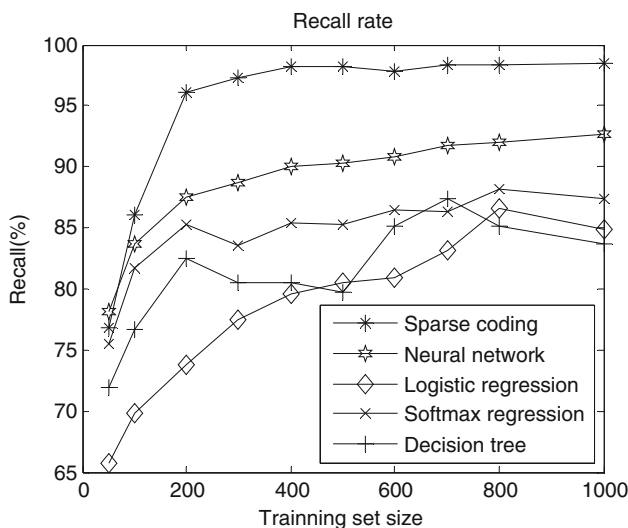


Fig. 10 Recall rate with noise level  $\sim N(0, 0.5)$

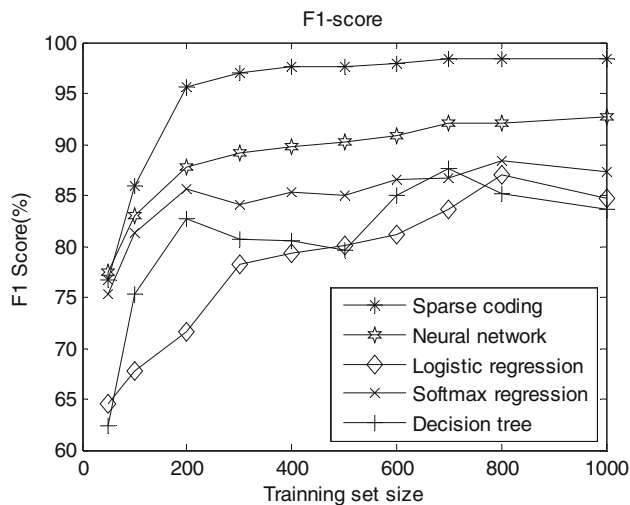


Fig. 12 F1-score with noise level  $\sim N(0, 0.5)$

both of them and gives us a comprehensive understanding of the performance of algorithms. Here is the definition of F1-score:

$$F_1 - score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{9}$$

As the same as accuracy and recall rate analysis, we also calculate this metric under two environmental noise levels. Figures 11 and 12 show the F1-score with all the five classification algorithms.

Similar to recall rate, when the noise level is low, all algorithms can achieve a good performance, as Fig. 11 shows. The F1-score of neural network and softmax regression are 96 and 94 % roughly, while our method can achieve a F1-score about 98 %. In Fig. 12, the F1-score of

neural network and softmax regression decreases to 93 and 87 %, respectively. Similarly, the F1-score of logistic regression and decision tree is a little bit lower, about 85 %. However, our method can still hold a relatively high F1-score, despite the effect of a stronger noise level. Compare with the two figures, it is obvious that our method has a better performance than others in a real situation application.

#### 4 Conclusion

In this paper, we apply deep learning techniques combined with a wireless sensor network for SHM and propose a new method using sparse coding to learn a feature

representation to enhance classification performance. From the simulation, the learned features from sparse coding not only improve the performance of classification but also make our approach more tolerant to environmental noises. Performance comparison also demonstrates the efficiency and robustness of our algorithm.

## 5 Future work

Although our method performs very well in the simulation experiments, the actual performance still need to be testify in a real scenario. In the feature work, we will choose a real bridge and apply our method proposed in the paper to check its availability. In addition, the real health monitoring will be much more complicated and some unpredictable problems still need to be further discussed.

**Acknowledgments** This research is sponsored by National Natural Science Foundation of China (61171014,61272475, 61371185) and the Fundamental Research Funds for the Central Universities (2013NT5, 2012LYB46) and by SRF for ROCS, SEM.

## References

- Li A, Ding Y, Wang H, Guo T (2012) Analysis and assessment of bridge health monitoring mass data—progress in research/development of “Structural Health Monitoring”. *Sci China Technol Sci* 55(8):2212–2224
- Ye XW et al (2012) Statistical analysis of stress spectra for fatigue life assessment of steel bridges with structural health monitoring data. *Eng Struct* 45:166–176
- Huang Y et al (2014) Robust Bayesian compressive sensing for signals in structural health monitoring. *Comput Aided Civil Infrastruct Eng* 29(3):160–179
- McCague C et al (2014) Novel sensor design using photonic crystal fibres for monitoring the onset of corrosion in reinforced concrete structures. *J Lightwave Technol* 32(5):891–896
- Mujica LE et al (2014) A structural damage detection indicator based on principal component analysis and statistical hypothesis testing. *Smart Mater Struct* 23(2):25014–25025
- Ofstun, SC, Wilmering TJ (2004) Model-driven development of integrated health management architectures. *Aerospace conference, 2004. proceedings. 2004 IEEE. vol 6. IEEE*
- Biswas G, Sankaran M (2007) A hierarchical model-based approach to systems health management. *Aerospace conference, 2007 IEEE. IEEE*
- Tian Zhigang, Zuo Ming J (2010) Health condition prediction of gears using a recurrent neural network approach. *IEEE Trans Reliab* 59(4):700–705
- Fukushima Kunihiko (1980) Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern* 36(4):193–202
- Allen DW, et al (2001) Damage detection in building joints by statistical analysis. *IMAC-XIX: a conference on structural dynamics, vol 2*
- Guidorzi R et al (2014) Structural monitoring of a tower by means of MEMS-based sensing and enhanced autoregressive models. *Eur J Control* 20(1):4–13
- Ji S, Sun Y, Shen J (2014) A method of data recovery based on compressive sensing in wireless structural health monitoring. *Math Probl Eng* 2014:546478. doi:10.1155/2014/546478
- Torres-Arredondo, MA et al (2014) Data-driven multivariate algorithms for damage detection and identification: evaluation and comparison. *Struct Health Monit* 13.1:19–32
- Sung SH et al (2014) A multi-scale sensing and diagnosis system combining accelerometers and gyroscopes for bridge health monitoring. *Smart Mater Struct* 23(1):015005
- Rahmatalla Salam et al (2014) Finite element modal analysis and vibration-waveforms in health inspection of old bridges. *Finite Elem Anal Des* 78:40–46
- Antunes PC et al (2014) Dynamic structural health monitoring of a civil engineering structure with a POF accelerometer. *Sensor Rev* 34.1:36-41
- Boukabache H et al (2011) Sensors/actuators network development for aeronautics structure health monitoring. *Sensors, 2011 IEEE. IEEE*
- Junqi G, Hongyang Z, Yunchuan S et al (2013) Square-root unscented Kalman filtering based localization and tracking in the internet of things. *Personal Ubiquitous Comput*. doi:10.1007/s00779-013-0713-8
- Deraemaeker Arnaud, Preumont André (2006) Vibration based damage detection using large array sensors and spatial filters. *Mech Syst Signal Process* 20(7):1615–1630
- Bishop CM, Nasrabadi NM (2006) *Pattern recognition and machine learning, vol 1*. Springer, New York
- Worden Keith, Manson Graeme, Allman David (2003) Experimental validation of a structural health monitoring methodology: part I. Novelty detection on a laboratory structure. *J Sound Vib* 259(2):323–343
- Manson Graeme, Worden Keith, Allman David (2003) Experimental validation of a structural health monitoring methodology: part II. Novelty detection on a Gnat aircraft. *J Sound Vib* 259(2):345–363
- Sohn H et al (2001) Structural health monitoring using statistical pattern recognition techniques. *J Dyn Syst Meas Control* 123(4):706–711
- Yoon H et al (2013) Algorithm learning based neural network integrating feature selection and classification. *Expert Syst Appl* 40(1):231–241
- Xiaobo X, Junqi G, Hongyang Z et al (2013) Neural-network based structural health monitoring with wireless sensor networks. 9th international conference on natural computation and 10th international conference on fuzzy systems and knowledge discovery (ICNC'13-FSKD'13)
- Malhi Arnaz, Yan Ruqiang, Gao Robert X (2011) Prognosis of defect propagation based on recurrent neural networks. *IEEE Trans Instrum Meas* 60(3):703–711
- Na S, Lee HK (2013) Neural network approach for damaged area location prediction of a composite plate using electromechanical impedance technique. *Compos Sci Technol* 88:62–68
- Dackermann U et al (2013) Identification of member connectivity and mass changes on a two-storey framed structure using frequency response functions and artificial neural networks. *J Sound Vib* 332(16):3636–3653
- Yan LJ et al (2013) Substructure vibration NARX neural network approach for statistical damage inference. *J Eng Mech Asce* 139(6):737–747
- Kao CY, Loh CH (2013) Monitoring of long-term static deformation data of Fei-Tsui arch dam using artificial neural network-based approaches. *Struct Control Health Monit* 20(3):282–303
- Hinton Geoffrey E, Salakhutdinov Ruslan R (2006) Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507

32. Hinton G et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *Signal Process Mag IEEE* 29(6):82–97
33. Le QV (2013) Building high-level features using large scale unsupervised learning. acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on. IEEE
34. Turian J, Lev R, Yoshua B (2010) Word representations: a simple and general method for semi-supervised learning. Proceedings of the 48th annual meeting of the association for computational linguistics. Association for Computational Linguistics
35. Socher R et al (2011) Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. NIPS 24
36. Socher R et al (2011) Parsing natural scenes and natural language with recursive neural networks. In: Proceedings of the 28th international conference on machine learning (ICML-11)
37. Lee H et al (2007) Efficient sparse coding algorithms. *Adv Neural Inf Process Syst* 19:801
38. Olshausen Bruno A (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381(6583):607–609
39. SDTools, Structural Dynamics Toolbox. <http://www.sdtools.com>
40. Young's modulus, Wikipedia. [http://en.wikipedia.org/wiki/Young's\\_modulus](http://en.wikipedia.org/wiki/Young's_modulus)
41. Hosmer Jr, DW, Lemeshow S, Sturdivant RX (2013) Applied logistic regression. Wiley. com
42. Dunne RA, Campbell NA (1997) On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In: Proceedings of the 8th Australian conference on the neural networks, Melbourne, 181, vol 185
43. Magerman DM (1995) Statistical decision-tree models for parsing. In: Proceedings of the 33rd annual meeting on association for computational linguistics. Association for Computational Linguistics