This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

Earthquake Prediction based on Spatio-Temporal Data Mining: An LSTM Network Approach

Qianlong Wang, Yifan Guo, Lixing Yu, and Pan Li

Abstract—Earthquake prediction is a very important problem in seismology, the success of which can potentially save many human lives. Various kinds of technologies have been proposed to address this problem, such as mathematical analysis, machine learning algorithms like decision trees and support vector machines, and precursors signal study. Unfortunately, they usually do not have very good results due to the seemingly dynamic and unpredictable nature of earthquakes. In contrast, we notice that earthquakes are spatially and temporally correlated because of the crust movement. Therefore, earthquake prediction for a particular location should not be conducted only based on the history data in that location, but according to the history data in a larger area. In this paper, we employ a deep learning technique called long short-term memory (LSTM) networks to learn the spatio-temporal relationship among earthquakes in different locations and make predictions by taking advantage of that relationship. Simulation results show that the LSTM network with two-dimensional input developed in this paper is able to discover and exploit the spatio-temporal correlations among earthquakes to make better predictions than before.

Index Terms—Earthquake Prediction, Spatio-Temporal Data mining, LSTM.

1 Introduction

Earthquakes are one of the most destructive natural disasters. They usually occur without warning and do not allow much time for people to react. Therefore, earthquakes can cause serious injuries and loss of life and destroy tremendous buildings and infrastructure, leading to great economy loss. The prediction of earthquakes is obviously critical to the safety of our society, but it has been proven to be a very challenging issue in seismology [23].

Existing works on earthquake prediction can be mainly classified into four categories according to the employed methodologies, i.e., 1) mathematical analysis, 2) precursor signal investigation, 3) machine learning algorithms like decision trees and support vector machines (SVM), and 4) deep learning. The first type of work tries to formulate the earthquake prediction problem by using different mathematical tools [4], like the FDL (Fibonacci, Dual and Lucas) method, kinds of probability distribution or other mathematics proving and spatial connection theory [12]. In the second type of work, researchers study earthquake precursor signals to help with earthquake prediction. For example, electromagnetic signals [8], aerosol optical depth (AOD) [1], lithosphere-atmosphere-ionosphere [13] and cloud image [5], [25] have been explored. Even animals' abnormal behavior has been taken into account in this kind of study [9]. The third type of work mainly explores data mining and time series analysis methods, such as J48, adaboost, multi-objective info-fuzzy network (M-IFN), k-nearest neighbors (kNN), SVM, and artificial neural networks (ANNs) [14], [3], to predict the magnitude of the largest earthquake in the next year based on the previously recorded seismic events in the same region. In the fourth type of work, deep learning algorithms are utilized to predict both the magnitude and the time of major seismic events. Various kinds of neural networks have been adopted, such as multi-layer perceptron (MLP) [16], backward propagation (BP) neural network [15], feed forward neural network (FFNN) [21], recurrent neural network (RNN) [19], which can work under certain particular circumstances.

Although there have been a lot of works on earthquake prediction, very few of them can predict future seismic events accurately. The reason is that the occurrence of earthquakes involves processes of very high complexity and depends on a large number of factors that are difficult to analyze. There are obviously complex nonlinear correlations among earthquake occurrences, because of which traditional mathematical, statistical, and machine learning methods cannot analyze well in this process. Recently, deep learning methods like RNNs are shown to be able to capture the nonlinear correlations among data [27], [28]. Particularly, they are mostly used to analyze time-series data so as to make predictions. As a result, when previous works use deep learning to make predictions, they predict earthquakes in a particular location only based on the history time-series data in that location, and hence still cannot get good results. In contrast, we contend that the spatio-temporal correlations among history earthquake data have to be investigated in order to make more accurate predictions.

To this end, in this paper we investigate earthquake prediction from a spatio-temporal perspective. Specifically, we devise an earthquake prediction scheme by adjusting a long short-term memory (LSTM) network, which is an advanced RNN and has strong nonlinear learning capability even on the data containing long-term interval correlations that the RNN is not able to achieve. We consider as a whole the earthquakes in an area of interest (e.g., a country) to be an input element to the LSTM network, which is different from common deep learning approaches that only consider the data in one particular location as an input. Therefore, by having a time-series of such input elements, we can

This work was partially supported by the U.S. National Science Foundation under grants CNS-1602172 and CNS-1566479.

Q. Wang, Y. Guo, L. Yu, and P. Li are with Department of Electrical Engineering and Computer Science, Case Western Reserve University, Cleveland, OH 44106. Email: {qxw204, yxg383, lxy257, lipan}@case.edu.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

construct an LSTM network with two-dimensional input that can learn the correlations among earthquakes in different locations and at different time, and exploit it to make predictions. After building LSTM network, we find that it is difficult to well train the network due to its high complexity and the lack of training data. Then, we decompose the original LSTM into several smaller ones to reduce the complexity and the need for larger training data sets.

Our main contributions in this paper can be summarized as follows.

- We investigate the earthquake prediction problem from a spatio-temporal perspective.
- We construct an LSTM network with two-dimensional input, which can discover the spatio-temporal correlations among history earthquake data, and exploit it to make predictions on earthquakes in a large area of interest.
- We decompose the original large LSTM network into several smaller ones, which can lower the complexity and facilitate the network training.
- Simulation results show that the proposed LSTM approach can obtain good performances.

The rest of this paper is organized as follows. Section 2 introduces the most related work on earthquake prediction methods. Section 3 describes the proposed system model for earthquake prediction. Section 4 details the proposed LSTM based scheme, which is followed by simulation results and discussions in section 5. Finally, we conclude the paper in section 6.

2 Related Work

In this section, we introduce in detail the related works on earthquake prediction, which are classified into four categories as we mentioned above.

First, some works employ mathematical or statistical tools to make earthquake prediction. Kannan [12] predicts earthquake epicenters according to spatial connections theory, i.e., earthquakes occurring within a fault zone are related to one another. Particularly, predictions are made by taking advantages of Poisson range identifier function (PRI), Poisson distribution, etc. Boucouvalas et al. [4] improve the Fibonacci, Dual and Lucas (FDL) method and propose an scheme to predict earthquakes by using a trigger planetary aspect date prior to a strong earthquake as a seed for the unfolding of FDL time spiral. However, these works are only tested with very limited amount of data and do not provide good results (the success rate is low).

Second, some works predict earthquakes base on precursor signals studies. Hayakawa [8] and Jiang [11] take the electromagnetic signals as the precursor of significant earthquakes. Thomas et al. [25] and Fan et al. [5] have studied satellite images of clouds before earthquakes. Akhoondzadeh and Chehrebargh [1] claim that unusual aerosol optical depth (AOD) variations before earthquakes could be introduced as an earthquake precursor. Meanwhile, Korepanov [13] propose a earthquake precursor based on lithosphere-atmosphere-ionosphere coupling and relations. Florido et al. [6] discover precursory patterns for large earthquakes. Also, the new attributes, based on the well-known b- value, are also generated. In addition, Hayakawa et al. [9] study the abnormal behavior of animals about 10 days before earthquakes in order to make earthquake prediction. Unfortunately, it is difficult to draw conclusions on theses precursor signals due to very limited data. Besides, these precursor signals alone usually cannot lead to satisfactory prediction results.

Third, machine learning has been employed as an important method to make earthquake prediction. Last et al. [14] compare several data mining and time series analysis methods, which include J48, AdaBoost, information network (IN), multi-objective info-fuzzy network (M-IFN), k-nearest neighbors (k-NN) and SVM, for predicting the magnitude of the largest coming seismic event based on previously recorded seismic events in the same region. Besides, the prediction features based on the Gutenberg-Richter Ratio as well as some new seismic indicators are proved to be much more useful than those traditionally used in the earthquake prediction literature, i.e., the average number of earthquakes in each region. Asencio-Cort et al. [2] study the sensitivity of the existing seismicity indicators reported in the literature by changing the input attributes and their parameterization. We notice that most machine learning methods make earthquake prediction based on seismicity indicators, where only time-domain but not space domain correlations are studied. Moreover, traditional machine learning methods expose their limitations on mining data with complex nonlinear correlations.

Fourth, recently deep learning methods have been applied to earthquake prediction. Narayanakumar and Raja [18] evaluate the performance of BP neural network techniques in predicting earthquakes. They gather data with event time, latitude, longitude, depth and magnitude to convert them into input for the neural network. The results show that the BP neural network method can provide better prediction accuracy for earthquakes of magnitudes 3 to 5 than previous works, but still cannot have good results for earthquakes of magnitude 5 to 8 due to the lack of sufficient data. Moustra et al. [17] first make earthquake predictions by using time series magnitude data, they then use seismic electric signals (SES) to further improve their results. Li and Liu [15] develop particle swarm optimization (PSO) algorithm to optimize the parameter of a BP neural network. Particularly, they improve the PSO algorithm by adding nonlinear decreasing inertia weight to enhance searching prediction. Saba et al. [21] predict earthquakes combining the Bat algorithm and a feed-forward neural network (FFNN). Mahmoudi et al. [16] use an MLP network to predict the magnitudes of earthquakes. With online training, which is superior to batch training for large data sets, as their training method, MLP has good prediction performance. We notice that most of these neural network methods use various kinds of features as input to predict the time and/or magnitudes of earthquakes, but few of them consider the spatial relations among earthquakes. Moreover, the spatio-temporal correlations among earthquakes are not studied.

3 System Model

In this study, we propose to make prediction on earthquakes by taking advantages of the spatio-temporal correlations among them. The intuition is that 1) earth is connected, and hence the seismic activities in one location will naturally lead to seismic activities in other locations, and 2) the seismic activities tend to have certain patterns in the time domain.

In particular, we divide an area of interest into several sub-regions to facilitate spatio-temporal earthquake data mining. Our objective is to predict earthquakes in each sub-region. We denote the system status at time t by a "multi-hot" vector $\mathbf{x}_t$, each element of which is equal to 1 or 0, representing there are either earthquakes in the corresponding sub-region in time slot t (being hot) or not. Define M as the total number of sub-regions. Then, $\mathbf{x}_t$ is a vector of dimension $M \times 1$ as shown in Fig. 1, where the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

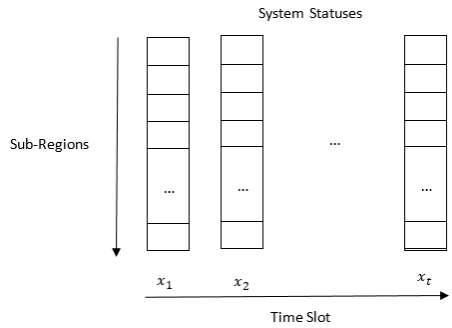


Fig. 1. System statuses represented by $M \times 1$ vectors.

elements equal to 1 are called "hot" elements. Therefore, our goal is to predict the next system status $\mathbf{x}_{t+1}$ based on previous system statuses, i.e., $\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots$, etc.

We notice that how to divide the area of interest is an important issue here. Without loss of generality, in this study we choose to divide the area evenly into rectangular sub-regions. Particularly, we consider the area of interest to be a rectangular area with the four vectors denoted by V_{nw}, V_{sw}, V_{ne}, V_{se}, respectively. The latitude and longitude of a point are represented by $La(\cdot)$ and $Lo(\cdot)$ respectively. Thus, we denote the whole area into M, which is equal to $m_h \times m_v$, sub-regions. The vertical and horizontal edges of each sub-region are as follows:

$$SR_{ve} = |La(V_{nw}) - La(V_{sw})|/m_v \tag{1}$$

$$SR_{he} = |Lo(V_{nw}) - Lo(V_{ne})]/m_h \tag{2}$$

where, SR_{ve} means the length of vertical edge for each sub-region and SR_{he} means the length of the horizontal edge.

Thus, when we build the system model and represent the real earthquake data by the multi-hot vectors, we first define the length of a time slot, i.e., a week, a month, etc., and then check which sub-region each earthquake in this time slot happens in. This can easily be done by finding the index k of the sub-region (numbered from top to down and from left to right): for an earthquake at location e, the sub-region index k is

$$k = m_v i + j + 1 \tag{3}$$

where i is the coordinate on the horizontal axis and j is the coordinate on the vertical axis for each earthquake. Thus, i and j can be calculated by

$$i = \lfloor \frac{|Lo(e) - Lo(V_{nw})|}{SR_{he}} \rfloor, \tag{4}$$

$$j = \lfloor \frac{|La(e) - La(V_{nw})|}{SR_{ve}} \rfloor. \tag{5}$$

We describe our system modeling process in detail in Algorithm 1.

Algorithm 1 System Modeling from the Spatio-Temporal Perspective

Input: The gathered raw information of earthquake events $E = \{e_1, e_2, \ldots\}$. For each event e, time(t), latitude(La), longitude(Lo), magnitude(Ma) information is given.

Initialization. Select the study area of rectangle shape with four vertices expressed as V_{nw}, V_{sw}, V_{ne}, V_{se}.
Spacial Segmentation:
Divide the area into sub-regions. The shape of sub-region generated with $SR_{ve} = |La(V_{nw}) - La(V_{sw})|/m_h$, $SR_{he} = |Lo(V_{nw}) - Lo(V_{ne})]/m_v$.
Allocate each event to belonged sub-region.
for e in E **do**
    $i = \lfloor \frac{|La(e) - La(V_{sn})|}{SR_{ve}} \rfloor$
    $j = \lfloor \frac{|Lo(e) - Lo(V_{sn})|}{SR_{he}} \rfloor$
    The sub-region with index of $k = i + 3j + 1 \leftarrow$ the given event e
end for
Temporal Segmentation:
Generate events frequency of each sub-region in each time interval Δt and generate the mult-hot input vector. For each sub-region, apply function below.
for each time interval Δt **do**
    if e exists **then**
        Frequency in current time interval $\leftarrow$ The number of the existing e.
    else
        Frequency in current time interval $\leftarrow 0$
    end if
end for
for each time interval Δt **do**
    for each sub-region **do**
        if CurrentFrequency $!= 0$ **then**
            CurrentFrequency $\leftarrow 1$
        else
            CurrentFrequency $\leftarrow 0$
        end if
    end for
end for

Output: Multi-Hot feature vector. Each Multi-Hot vector points out the sub-regions with earthquake occurred in the current time interval.

4 Earthquake Prediction using an LSTM Network with Two-Dimensional Input

In this section, we describe in detail our proposed earthquake prediction algorithm using an LSTM network with two-dimensional input. The main idea of our algorithm is to develop an LSTM network with two-dimensional input to predict the next system status based on a number of most recent system statuses. This is achieved by learning the correlations among earthquakes in different locations at different times. In the following, we will first introduce the fundamentals of the LSTM architecture and then explain how we devise an LSTM with two-dimensional input to develop our earthquake prediction algorithm.

4.1 The Basic LSTM Architecture

Long short-term memory (LSTM) is a redesigned architecture based on the traditional RNN. It was proposed by Sepp Hochreiter and Jrgen Schmidhuber in 1997 [10]. Notice that in theory RNN can well handle data with time dependency between each other, but in practice it struggles with the long-term temporal dependency problem. By having memory cells that record their states in a traditional RNN, an LSTM is able to learn relations among data during a long time interval.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

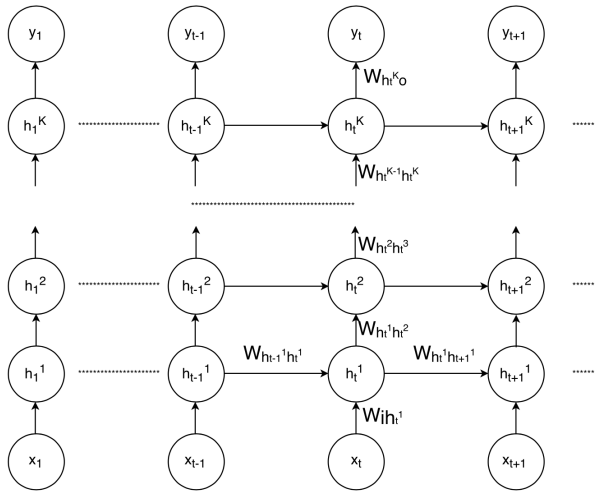


Fig. 2. The typical RNN architecture with K hidden layers. $\mathbf{h}_t^k$ represents the state of the kth hidden layer in time slot t. Solid lines mean the connections by weight.

4.1.1 The Typical RNN Architecture

Fig. 2 shows the architecture of a general RNN with n hidden layers. Compared with a normal ANN, an RNN tries to take advantage of information in the past time. Particularly, in an RNN, the output not only depends on the current input, but also depends on previous inputs. Denote the input vector at time t by $\mathbf{x}_t$. Then, the RNN network updates the hidden layer states $\mathbf{h}_t^1, \ldots, \mathbf{h}_t^K$ and computes the output $\mathbf{y}_t$ based on the input $\mathbf{x}_t$ and the hidden layer statuses at the past time instance. $\mathbf{h}_t^k$ denotes the kth hidden layer's state at time t, which is essentially a vector with the number of the elements representing the number of nodes at the kth hidden layer. As shown in Fig. 2, we can see that the past input information would be propagated horizontally in each layer through weight matrices and nonlinear functions, and hence can be used for prediction.

Specifically, an RNN works as follows. We usually set the initial input at time $t = 0$ to $\mathbf{0}$. Therefore, at time t, the hidden layer states are updated according to the follow equations:

$$\begin{aligned} \mathbf{h}_t^1 &= \mathcal{F}(\mathbf{W}_{ih_t^1}\mathbf{x}_t + \mathbf{W}_{h_{t-1}^1 h_t^1}\mathbf{h}_{t-1}^1 + \mathbf{b}_t^1) \\ \mathbf{h}_t^k &= \mathcal{F}(\mathbf{W}_{h_t^{k-1} h_t^k}\mathbf{h}_t^{k-1} + \mathbf{W}_{\mathbf{h}_{t-1}^k h_t^k}\mathbf{h}_{t-1}^k + \mathbf{b}_t^k) \end{aligned}$$

where $2 \leq k \leq K$. Here, $\mathcal{F}$ is a nonlinear hidden layer function that, for example, can be set as a sigmoid function. $\mathbf{W}_{ih_t^1}$ denotes the weight matrix connecting the input to the first hidden layer at time t, $\mathbf{W}_{\mathbf{h}_{t-1}^k h_t^k}$ denotes the recurrent connection matrix between the kth hidden layers at time $t-1$ and at time t, $\mathbf{W}_{h_t^{k-1} h_t^k}$ denotes the weight matrix connecting the $(k-1)$th and kth hidden layer at time t and $\mathbf{b}$ denotes the bias vector. In particular, suppose that we have $\mathbf{x}_t \in R^{M\times 1}$ and the number of nodes at the kth hidden layer is N^k. Then, we can know that the dimensions of the matrices $W_{ih_t^1}, W_{h_{t-1}^k h_t^k}, W_{h_t^{k-1} h_t^k}$ are $N^1 \times M, N^k \times N^k, N^k \times N^{k-1}$ respectively, and the dimension of vector $\mathbf{b}_t^k$ is $N^k \times 1$. Note that these parameters will be optimized during the training process.

Besides, the output at time t denote by $\mathbf{y}_t$ can be calculated as:

$$\mathbf{y}_t = \mathbf{W}_{h_t^K o}\mathbf{h}_t^K + \mathbf{d}_t$$

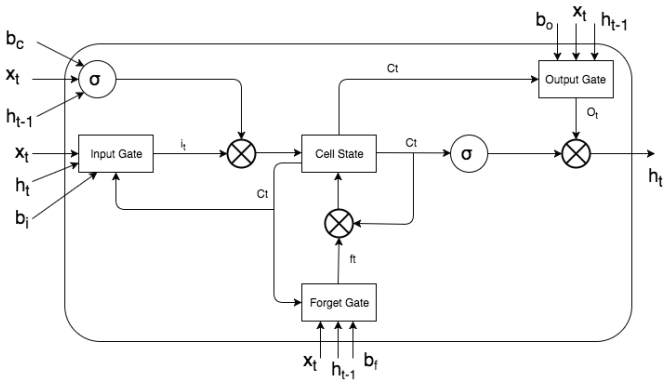


Fig. 3. The typical LSTM Single Memory Cell.

Here, $\mathbf{y}_t \in R^{M\times 1}$, $\mathbf{W}_{h_t^K o} \in R^{M\times N^K}$ is the weight matrix between the Kth hidden layer and the output, and $\mathbf{d}_t \in R^{M\times 1}$ is the bias vector for $\mathbf{y}_t$, respectively. Similar to the other parameters mentioned above, these parameters will be optimized during the training process.

4.1.2 The Typical LSTM Architecture

As mentioned before, RNNs are incapable of handling long-term time dependency in practice, LSTMs are explicitly designed to avoid the long-term dependency problem. In particular, LSTMs have the same chain like structure, but they use a different way to inplement function $\mathcal{F}$ in order to store information. It is to build a memory cell instead, which could be considered as a black box that, for example, at the first layer, takes the previous state $\mathbf{h}_{t-1}$ and current system input $\mathbf{x}_t$ as inputs and compute internally to decide what to keep in memory and output the hidden state $\mathbf{h}_t$. Fig. 3 shows the typical architecture of a single LSTM memory cell [7]. We can see that the cell state runs straight down the entire path with only some linear interactions, which makes it very easy for information to be propagated in time.

To describe the memory cell in an LSTM in more detail, we have the following equations:

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \\ \mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \phi(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \\ \mathbf{h}_t &= \mathbf{o}_t \circ \phi(\mathbf{c}_t) \end{aligned}$$

Here, i, f, o and c denotes the input gate, forget gate, output gate, and cell state, respectively. These gates are all of the same dimension as the hidden vector $\mathbf{h}$ ($N^k \times 1$ in the kth cell). σ is a sigmoid function, and ϕ is a nonlinear function which maps the input to $[-1, 1]$.$\mathbf{W}_{ic}$, $\mathbf{W}_{fc}$ and $\mathbf{W}_{oc}$ are the peephole connection matrices, which connect cell state to input gate, forget gate, and output gate, respectively. Similarly, $\mathbf{W}_{ix}, \mathbf{W}_{fx}, \mathbf{W}_{ox}$ and $\mathbf{W}_{cx}$ are the weight matrices connecting between input vector $\mathbf{x}_t$ and input gate, forget gate, output gate and cell state, respectively. Besides, since the gates and the input vector $\mathbf{x}_t$ have the dimension of $N \times 1$ and $M \times 1$ respectively, we can have that the dimensions of matrices $\mathbf{W}_{ih}, \mathbf{W}_{ic}, \mathbf{W}_{fh}, \mathbf{W}_{fc}, \mathbf{W}_{ch}, \mathbf{W}_{oh}, \mathbf{W}_{oc}$ are all the same, which is $N \times N$, and the dimensions of matrices $\mathbf{W}_{ix}, \mathbf{W}_{fx}, \mathbf{W}_{cx}, \mathbf{W}_{ox}$ are $N \times M$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

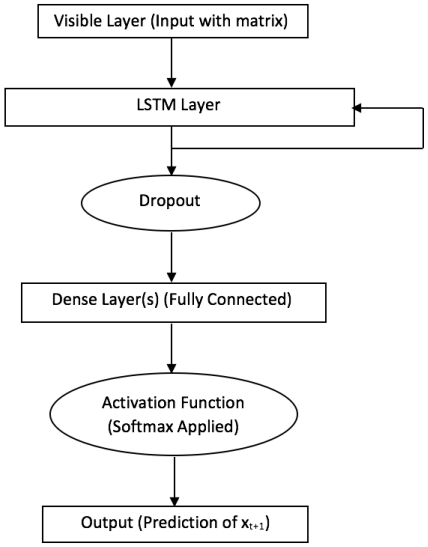


Fig. 4. The flow diagram of our system

4.2 The Proposed LSTM Network with Two-Dimensional Input

4.2.1 System Architecture

We notice that when previous works employ neural networks to predict earthquakes, they mainly consider the particular location and make predictions based on the historical earthquake data at this location, i.e., the input $\mathbf{x}_t$ has only one dimension and is only about one location. In so doing, they essentially make predictions based on the temporal correlations among historical data. In contrast, we consider that $\mathbf{x}_t$ is a vector representing earthquake data at time t at several locations, i.e., the M sub-regions as mentioned in our system model. The input to our LSTM network is a series, say L, of $\mathbf{x}_t$'s, i.e., a matrix of dimension $M \times L$. Therefore, we can make predictions for earthquakes in a large area not only based on temporal data dependencies, but also based on spatial data correlations.

The main idea of our system is shown by the flow chart in Fig. 4. Specifically, the input matrix first goes through the LSTM layer. Then, dropout process is applied to the output of the LSTM network, the result of which goes to the dense layer, i.e. a fully connected neural network. Finally, we apply an activation function, which is set to softmax function, and obtain the prediction result $\mathbf{x}_{t+1}$.

The architecture of our system is presented in Fig. 5. Notice that as we mentioned above, in our system $\mathbf{X}_t$ is a matrix of dimension $M \times L$. As in Fig. 6, in the training process, the target of prediction based on input matrix $\mathbf{X}_t$ at time t is $\mathbf{x}_{t+1}$, and in the prediction phase, $\mathbf{x}_{t+1}$ is what needs to be predicted at time t. In our architecture, $\mathbf{h}_t^L$ is an output of the LSTM layer at time t, which is constructed by memory cells depicted in Fig. 3. In particular, the details of our LSTM layer are shown in Fig. 7, where there are L memory cells, one for each time slot. The output of the jth memory cell at time t, i.e., $\mathbf{h}_{t-j}$ and $\mathbf{c}_j$, is part of the input of the next, i.e., the $(j-1)$th, memory cell. Besides, the output of the LSTM layer goes to a dense layer whose output is denoted by $\mathbf{h}_t^D$. In the following, we describe in detail what happens after the LSTM layer.

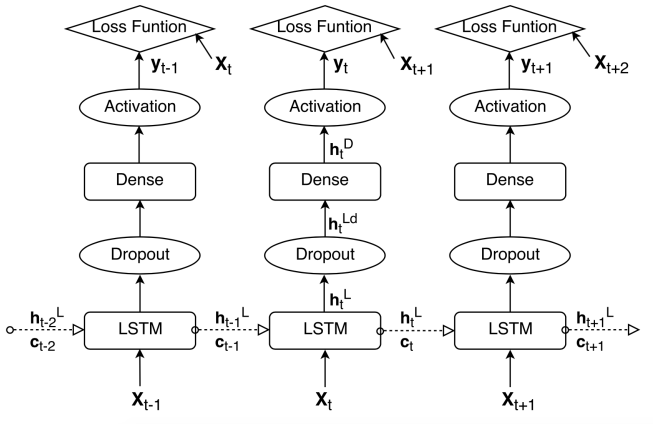


Fig. 5. Our system architecture. Dense means a fully connected neural network.

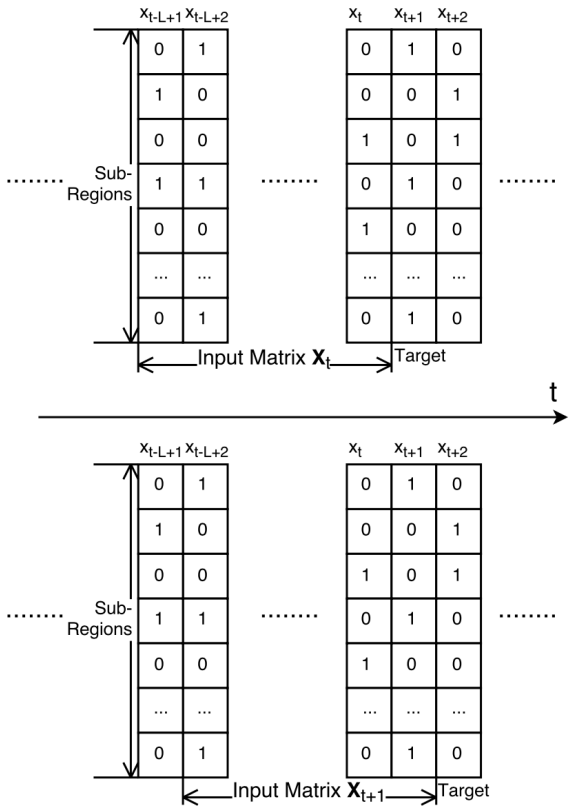


Fig. 6. The input matrix in our system.

4.2.2 Dropout

To prevent our system from being overfitted, we apply a method called dropout to the output of the LSTM layer. System overfitting can lead to very high performance in training but very low in testing. This is because when overfitting occurs, the system focuses too much on historical data, which makes it too rigid to give satisfactory result on new input. Many works have proved that adding dropout in the system can efficiently prevent a neural network from being overfitted [24]. In particular, by having dropout in the system, a certain number of randomly selected nodes are temporarily turned off in each training, along with all its conjoint connections. Therefore, in our case, we apply dropout between

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

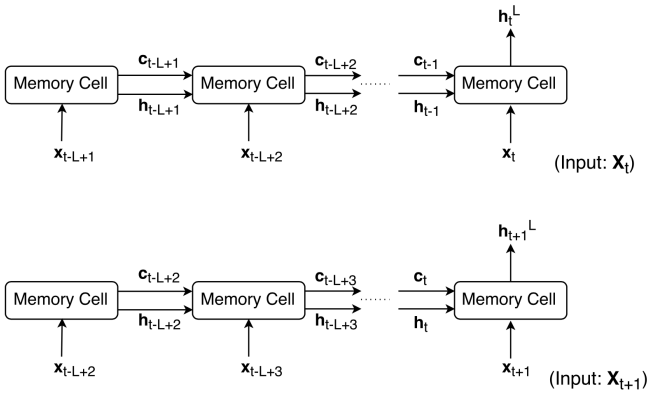


Fig. 7. The zoomed-in LSTM layer architecture with input $\mathbf{X}_t$ and input $\mathbf{X}_{t+1}$, respectively. There are L memory cells in the LSTM layer due to the fact that we need to look back L system statuses to make predictions.

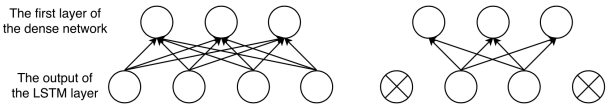


Fig. 8. Dropout Structure. The left side shows a normal system without dropout, while the right side is the system with dropout applied.

the LSTM layer and the dense network, which is shown in Fig. 8. Since some of the nodes in the output of the LSTM layer have been turned off, the system becomes insensitive to some extent, and hence can avoid from being "too smart", i.e., overfitted.

4.2.3 Dense Network

After the LSTM layer, we have the output of LSTM goes to a dense network, which is essentially a fully connected neural network. In this fully connected neural network, at each layer, each neuron gets connected to all the neurons at the previous layer. By going through the dense network, the output of LSTM is multiplied by a matrix and added a bias to. The reason for having a dense network here is that the output of the LSTM contains the feature information we need to make prediction, but it is still not exactly what we need. The dense network is so trying to learn the function between feature data and the prediction result. In our system, we set up two layers in the dense network. The processing in the fully connected network can represented below:

$$\mathbf{h_t^D} = \mathbf{W_D W_P h_t^L} + \mathbf{b}$$

where $\mathbf{W_P}$ and $\mathbf{h_t^L}$ are the weight matrix between the output of the LSTM layer and the dense network, and the the output of the LSTM network, respectively, after the dropout, $\mathbf{W_D}$ denotes the weight matrix in the dense network, $\mathbf{h_t^D}$ is the output of the dense network, and $\mathbf{b}$ is the bias.

4.2.4 Activation Function

To obtain the final output of the system, we choose softmax as the activation function and apply it to the output of the dense network. Particularly, the activation function maps the output vector into a vector of elements between 0 and 1, each of which represents earthquake probability in a sub-region and the sum of which equals to 1. The softmax function can be calculated as:

$$\mathbf{y_t^m} = \frac{e^{\mathbf{z^m}}}{\sum_{i=1}^{M} e^{\mathbf{z^i}}}, \text{for } m = 1, \ldots, M$$

Here, we use $\mathbf{z}$ to represent the output of the dense network $\mathbf{h_t^D}$ for simplicity. $\mathbf{z^m}$ and $\mathbf{y_t^m}$ represent the mth element in vector $\mathbf{z}$, and that in the output $\mathbf{y_t}$, respectively. Note that the result is a vector of probabilities between 0 and 1 but not binary results that we need for crime prediction yet. To map the probabilities into 0s or 1s, we obtain an optimal probability threshold in the training process that minimizes the sum of the absolute value of the differences between the predicted label values and the real label values, which are either 0s or 1s.

Moreover, to have the system learn to hit the target value in the training process, we need to define a loss function. Since our problem is essentially a classification into variant labels, we employ cross-entropy as the loss function, which is commonly used and tested to be appropriate [22]. Particularly, cross-entropy can be calculated as:

$$\xi(\mathbf{x_{t+1}}, \mathbf{y_t}) = -\sum_{i=1}^{M} \mathbf{x_{t+1}^i} \log \mathbf{y_t^i},$$

where $\mathbf{x_{t+1}^i}$ and $\mathbf{y_t^i}$ denote the ith element in $\mathbf{x_{t+1}}$ and in $\mathbf{y_t}$, respectively.

To train our system, our goal is to minimize the loss function. We use the gradient descent method due to its efficiency. In particular, we utilize RMSprop to minimize the loss function, which has been experimentally proved to be an effective algorithm for RNNs [26].

Algorithm 2 The training Process of the Proposed LSTM

Input: $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_t}$

1: Enter the LSTM Layer, and calculate function $\mathbf{h_t} = \mathbf{o_t} \circ \phi(\mathbf{c_t})$.
2: Apply Dropout.
3: Go through the dense network. Calculate $\mathbf{h_t^D} = \mathbf{W_D W_P h_t^L} + \mathbf{b}$.
4: Apply softmax as the activation function.
5: Calculate cross-entropy function as the loss function.
6: Employ the gradient descent method to minimize the loss function, and hence optimize the system parameters.

We summarize the training process of our proposed LSTM network in Algorithm 2.

4.2.5 Improving System Performance by Decomposition

So far we have introduced how our proposed LSTM works. However there are two more problems: first, by considering a large area consisting of many sub- regions, we may have a very large system with many variables, which require a large amount of training data to fully train the system, and second, by considering the sub-regions all together, are make earthquake predictions by taking advantages of the spatio-temporal correlations among earthquake data in these sub-regions, while in fact same sub-regions may not be very closely related in practice and hence will hinder the correct prediction. The first problem makes the system computationally very expensive, and he second problem leads to less accurate predictions. In the following, we propose to improve the system efficiency and accuracy by decomposition.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

Specifically, we divide all the sub-regions into groups, which collectively and exclusively cover the whole area of interest. We train the groups separately and make earthquake predictions for the sub-regions in the groups respectively. It is obvious that how to form the groups is a very important problem. We choose to put the sub-regions within the same fault zone into the same group. In so doing, the disturbance from not-so-related sub-regions can be mitigated, the amount of training data and the computational complexity can be significantly reduced.

5 Performance Evaluation

In this section, we evaluate the performance of the proposed system through two case studies. In the first case, we study the system performance when we use one-dimensional input as before in our system; in the second case, we explore the system performance when two-dimensional input is used as we propose in this study.

5.1 Case Study I: The Proposed LSTM Network with One-dimensional Input

As mentioned before, in this case, we make earthquake predictions by using the proposed LSTM network with only one-dimensional input, i.e., by exploiting the temporal correlations only.

5.1.1 Data preprocessing

The data that we use is gathered from the USGS (US Geological Survey) website. In particular, we use Conterminous U.S earthquake data from 2006 to 2016 with magnitudes greater than 2.5 in our simulations. We set one time slot to one month. In each time slot, the input is the number of earthquakes that happened in this time slot in a certain sub-region. We have 120 data items when one time slot is one month. As usual, we divide the data into two parts: training data and testing data. Particularly, the first 2/3 of data would be used for training and the rest would be used for testing.

5.1.2 LSTM Network Settings

In this case, we build our LSTM network with one-dimensional input only in the time domain. The output of the LSTM layer has 4 neurons. The activation function is set by default to the sigmoid function and makes a single value prediction. The "look back window" of the system is set to 1 and 10, which is the number of most recent data that we consider as input to predict the next time slot variables.

5.1.3 Simulation Results and Discussions

Fig. 9 and Fig. 10 show the prediction results for the total number of earthquakes when the look back window is equal to 1 and 10, respectively. The blue line shows the real earthquake frequency distribution, while the green line and red line denote the training results and testing results respectively. Here, we adopt the mean squared deviation (MSD) as our loss function. We get MSDs of 40.37 and 42.50 for look back windows of 1 and 10, respectively, which represent highly inaccurate predictions.

Besides, we also employ our proposed LSTM network with one-dimensional input to predict whether there are earthquakes or not. The selected area of interest is in mainland China, particularly, between 75 E and 119 E longitudes and 23 N and 45 N latitudes, as shown in Fig. 11. We equally divide this area into nine smaller

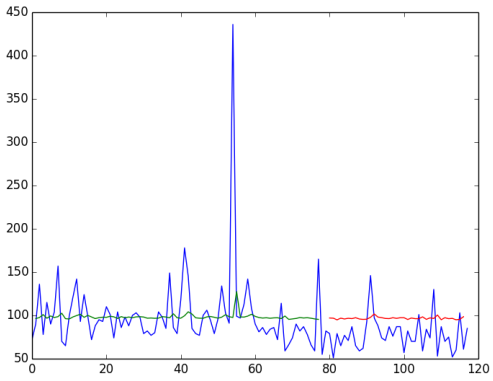


Fig. 9. Prediction results when the look back window is 1. The horizontal axis represents time slots and the vertical axis represents the number of earthquakes that have happened in the corresponding time slot.

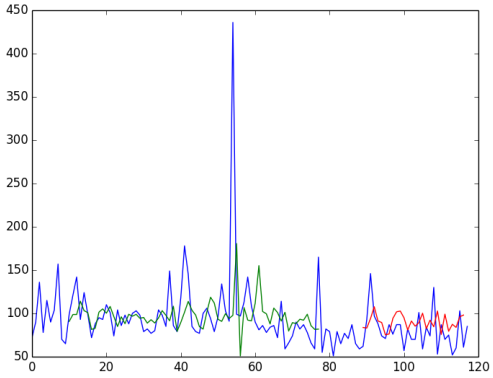


Fig. 10. Prediction results when the look back window is 10. The horizontal axis represents time slots and the vertical axis represents the number of earthquakes that have happened in the corresponding time slot.

sub-regions, and aim to predict whether there are earthquakes with magnitudes greater than 4.5 in each of the sub-regions with the data collected from 1966 to 2016. Besides, in our LSTM network, the LSTM layer has an output of 128 neurons, the dense network has 256 and 64 neurons in the first layer and second layer, respectively, and the output layer has 9 neurons. The activation function is set to the softmax function. Our results show that the overall prediction accuracy is 63.50%, with true positive accuracy of 46.83% and true negative accuracy of 79.6%.

5.2 Case Study II: The Proposed LSTM Network with Two-dimensional Input

In this case, by changing the input to two-dimensions, we take advantage of spatio-temporal correlations to make earthquake predictions.

5.2.1 Data Preprocessing

The same as that in the previous case, we gather data from USGS website, and use earthquake data in mainland China for our simulations. As shown in Fig. 11, our area of interest is still between 75 E and 119 E longitudes and 23 N and 45 N latitudes, which is equally divided into nine smaller sub-regions. We are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

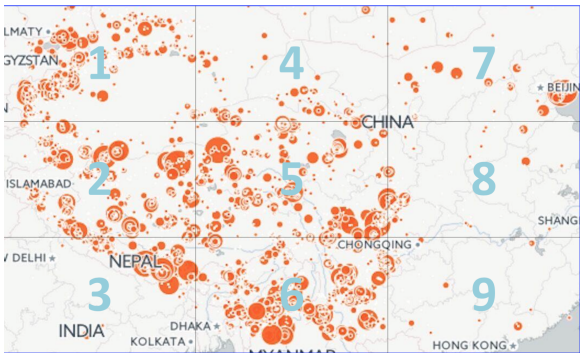


Fig. 11. Earthquakes with magnitudes greater than 4.5 from 1966 to 2016. The way we divide the area into 9 sub-regions is shown in the figure, where sub-regions have been numbered from 1 to 9.

time	latitude	longitude	depth	mag	magType
1966-12-16T20:52:18.000Z	29.535	80.823	20	5.7	mw
1967-01-30T21:05:30.000Z	26.107	96.115	25	5.6	mw
1967-02-20T15:18:41.000Z	33.617	75.333	20	5.5	mw
1967-03-14T06:58:05.000Z	28.463	94.248	16.9	5.7	mw
1967-03-27T08:58:24.000Z	38.474	116.608	29.7	6.1	mw
1967-05-27T19:05:48.000Z	36.034	77.714	20	5.9	mw
1967-08-15T09:21:03.000Z	31.079	93.639	20	5.7	mw
1967-08-30T04:22:04.000Z	31.633	100.262	10	6.4	mw
1967-08-30T11:08:49.000Z	31.587	100.255	10	5.8	mw
1967-09-15T10:32:44.000Z	27.338	91.91	13.7	5.8	mw
1968-12-22T09:06:37.000Z	36.306	101.772	25	5.6	mw
1969-02-11T22:08:54.000Z	41.389	79.307	13.3	6.1	mw
1969-10-17T01:25:14.000Z	23.041	94.636	135	6.3	mw
1970-01-04T17:00:41.000Z	24.185	102.543	11.3	7.1	mw
...	...	...	...	...	...
2016-09-22T16:50:33.440Z	30.1048	99.7368	23.45	4.7	mb
2016-09-22T17:23:15.960Z	30.1768	99.6502	21.01	5.3	mb
2016-09-26T15:48:53.970Z	30.1734	99.6087	23.32	4.9	mb
2016-09-28T21:36:31.970Z	29.6438	101.9644	10	4.5	mb

Fig. 12. Raw data of earthquakes with magnitudes greater than 4.5 from 1966 to 2016.

interested in the earthquakes in this area with magnitudes greater than 4.5 from 1966 to 2016. Some raw data is shown in Fig. 12. We define a time slot to be one month. Thus, we have 600 data items, as shown in Fig. 13. Here, the number in each data item means the frequency of earthquake events belong to the certain sub-region.

As explained in our system model, we represent the original input by a multi-hot vector, in which an element is set to 1 if the corresponding sub-region has earthquakes happened and 0 otherwise. Besides, we choose the look back window to be 12. Therefore, our input matrix is of 12×9. Moreover, 70% of our data is used for model training, and the remaining 30% is used

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	1
7	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
11	0	3	0	0	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
585	0	5	0	3	0	1	0	0	2
586	12	1	1	8	0	0	0	0	1
587	1	4	1	6	0	0	0	0	6
588	4	1	1	3	0	0	0	0	0
589	1	4	0	6	0	0	0	0	1
590	0	4	0	2	0	2	0	0	1
591	0	1	0	0	0	0	0	0	5
592	0	5	0	1	0	1	0	0	3
593	1	6	0	2	0	0	0	0	4
594	1	1	1	9	0	0	0	0	2
595	0	2	0	1	0	0	0	0	1
596	1	8	1	4	0	0	0	0	4
597	0	2	1	1	0	0	0	0	1
598	0	1	0	2	0	1	0	0	5
599	8	1	0	0	0	2	0	0	1
600	0	1	0	8	0	0	0	0	0

Fig. 13. Data of earthquake frequencies in all the 9 sub-regions.

	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0
5	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	0	0	0
12	0	0	0	0	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...
585	1	0	1	0	1	1	0	0	1
586	1	1	1	0	1	1	0	1	0
587	1	1	1	0	1	1	1	1	0
588	1	1	1	0	0	1	1	1	0
589	1	1	1	0	1	1	0	0	0
590	1	0	1	1	1	1	0	0	1
591	1	0	1	1	0	0	0	0	0
592	1	0	1	1	1	1	0	0	0
593	1	1	1	0	1	1	0	0	0
594	1	1	0	1	1	1	0	1	0
595	1	0	0	0	1	1	0	0	0
596	1	1	1	0	1	1	0	0	0
597	1	0	1	0	0	1	0	1	0
598	1	0	0	1	1	1	0	0	1
599	1	1	0	1	0	1	0	0	0
600	0	0	1	0	0	1	1	0	0

Fig. 14. An input matrix with multi-hot vectors.

for the testing. Fig. 14 has shown some of our generated multi-hot vectors.

5.2.2 LSTM Network Settings

We use the same settings as those used for obtaining the second set of simulation results in Section 5.1.3, i.e., predicting whether there are earthquakes or not in mainland China with one-dimensional input. Specifically, with two-dimensional input, our LSTM neural network has 128 neurons in the output of the LSTM layer, 256 and 64 neurons in the first layer and second layer of the dense network, respectively, and 9 neurons in the output layer. As we mentioned before, the softmax function is selected as our activation function. In addition, we employ RMSprop as our optimizer with the learning rate being 0.01.

5.2.3 Simulation Results and Discussions

By conducting simulation results with the proposed LSTM network with two-dimensional input, we find that the prediction accuracy on the testing data is 74.81%, with the true positive accuracy of 68.56% and true negative accuracy of 81.31% as shown in Fig. 15. We can easily see that the prediction performance so far is obviously much better than that in Case I with one-dimension input. Particularly, the true positive accuracy is much higher, i.e., 68.56% compared with 46.83%. Consequently, we can know that mining spatio-temporal data correlations does provide better prediction results than only mining temporal data correlations.

Moreover, notice that the above results are obtained when the area of interest is studied as a whole, and all the 9 sub-regions are considered to be closely correlated. Next we employ our decomposition method to further improve performance.

Specifically, the sub-regions 1, 2, 5, 6 cover Tibet, Sichuan, Xinjiang, Gansu and Ningxia provinces, which are within the same fault zone [20]. When we consider these four sub-regions as a group, the prediction accuracy is 88.57%. This indicates that our system can well learn spatio-temporal correlations among the earthquakes in these four sub-regions and make accurate predictions. Besides, the 3rd sub-region includes Nepal, which is also a region with intense earthquake activities. However, because of Himalayas mountains, Nepal is located in a different fault zone from all the other sub-regions within Mainland China. So it may have loose spatio-temporal correlations data for the other sub-regions. This has been confirmed by the fact that the overall prediction accuracy of group of 2, 3, 5 sub-regions is 52.46%, and that of the group of 3, 5, 6 is 56.25%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

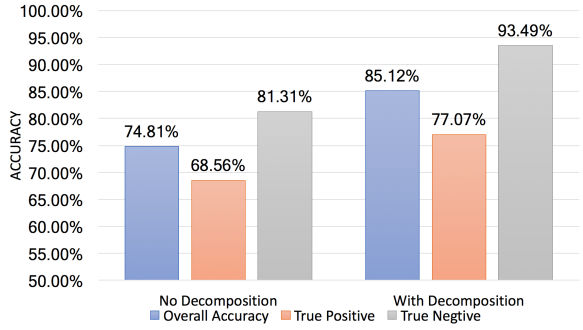


Fig. 15. Results comparison between without and with decomposition, with 3×3 sub-regions and one-month time slots.

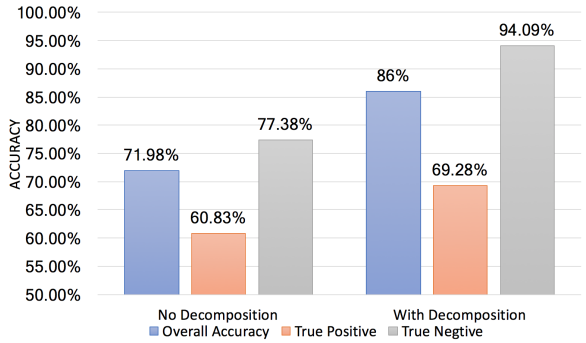


Fig. 16. Results comparison between without and with decomposition, with 3×3 sub-regions and two-weeks time slots.

After the analysis above, our final grouping plan is as follows. Group 1 consists of the 1st, 2nd, 5th and 6th sub-regions with prediction accuracy of 88.57%, group 2 includes the 4th, 7th, 8th and 9th sub-regions with prediction accuracy of 87.57%, and group 3 contains the 3rd sub-region with prediction accuracy of 61.60%. Combining the results together, we have that our overall prediction accuracy is 85.12% with true positive accuracy of 77.07% and true negative accuracy of 93.49%, which is also shown in Fig. 15. From the figure, we can clearly see the performance improvement in terms of prediction accuracy, true positive accuracy, and true negative accuracy, by applying our decomposition method.

On the other hand, we compare a previous earthquake prediction scheme with ours. Specifically, Moustra et al. [17] make earthquake prediction by using a multi-layer perceptron (MLP), which is a kind of traditional ANN. We run this method on our two-dimensional input data and the prediction accuracy is 66.99%, which is much lower than our result without decomposition, i.e., 74.81%, and that with decomposition, i.e., 85.12%.

Furthermore, we evaluate the performance of our system with input of different time slot sizes and different numbers of sub-regions. Note that previous results are obtained when each time slot is one month. Then, we conduct simulations by reducing each time slot to two weeks. With the same system settings, our system leads to prediction results shown in Fig. 16. We can find that without the decomposition method, the system gives lower prediction accuracy than that when we set each time slot to one month as shown in Fig. 15. This is because the input data becomes sparser when the time slot is reduced to two weeks only, which makes it more difficult for the system to find the correlations among earthquake occurrences. Nevertheless, when applying the proposed decomposition method, we can still achieve comparable results with those with one-month time slots in Fig. 15. Specifically, the overall accuracy becomes 86%, and the true positive accuracy and the true negative accuracy increase from 60.83% to 69.28% and from 77.38% to 94.09%, respectively. These results show that our proposed LSTM system with the decomposition method can work well with different temporal prediction granularities.

Besides, we also attempt to increase the number of sub-regions to make the spatial prediction more accurate. Similarly, we equally divide the whole area of interest into 5×5 sub-regions instead of the previous 3×3 sub-regions. To make fair comparisons, we still set each time slot to one month. Without applying the proposed decomposition method, the overall accuracy increases to 82.47%, which is better than 74.81% that is obtained when the whole area

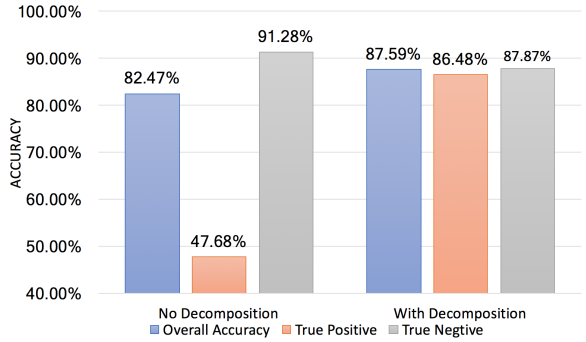


Fig. 17. Results comparison between without and with decomposition, with 5×5 sub-regions and one-month time slots.

is divided into 3×3 sub-regions. The complete results are shown in Fig. 17. We can see that although the overall accuracy seems good, the true positive accuracy has dropped from 68.56% to 47.68%, which is too low to correctly predict earthquakes. The reason is that similar to reducing the time slot size, the data becomes much sparser when the number of sub-regions increases from 9 to 25. Particularly, in the case of 25 sub-regions, the input vector becomes much longer, which makes mining the correlations among earthquake occurrences much more difficult. To address this issue, we apply our decomposition method. Here, the grouping plan is still based on the fault zone distribution, which is similar to what we use when there are 3×3 sub-regions. From Fig. 17, we can find that the overall accuracy increases from 82.47% to 87.59% with the decomposition method applied. More noticeably, the true positive accuracy significantly increases from 47.68% to 86.48%, which is the best result we get so far, and in the meantime, the true negative accuracy remains as high as 87.87%. Thus, with higher spatial prediction granularity, our LSTM system with the decomposition method can still effectively exploit the correlations among earthquakes to make accurate earthquake predictions.

To sum up, the above results clearly demonstrate the robustness and effectiveness of our new LSTM system. Last but not the least, comparing with previous earthquake prediction methods, which are mostly based on various seismic indicators, our system has little overhead on obtaining the input data. Particularly, even in areas where there are no seismic sensors and monitors, we are still able to use our system for earthquakes prediction.

6 Conclusions

In this paper, we have proposed a new earthquake prediction system from the spatio-temporal perspective. Specifically, we have designed an LSTM network with two-dimensional input, which can discover the spatio-temporal correlations among earthquake occurrences and take advantage of the correlations to make accurate earthquake predictions. The proposed decomposition method for improving the effectiveness and efficiency of our LSTM network has been shown to be able to significantly improve the system performance. Simulation results also demonstrate that our system can make accurate predictions with different temporal and spatial prediction granularities.

References

[1] M. Akhoondzadeh and F. J. Chehrebargh. Feasibility of anomaly occurrence in aerosols time series obtained from modis satellite images during hazardous earthquakes. *Advances in Space Research*, 2016.

[2] G. Asencio-Cortés, F. Martínez-Álvarez, A. Morales-Esteban, and J. Reyes. A sensitivity study of seismicity indicators in supervised learning to improve earthquake prediction. *Knowledge-Based Systems*, 101:15–30, 2016.

[3] G. Asencio-Cortés, F. Martínez-Álvarez, A. Morales-Esteban, J. Reyes, and A. Troncoso. Improving earthquake prediction with principal component analysis: application to chile. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 393–404. Springer, 2015.

[4] A. Boucouvalas, M. Gkasios, N. Tselikas, and G. Drakatos. Modified-fibonacci-dual-lucas method for earthquake prediction. In *Third International Conference on Remote Sensing and Geoinformation of the Environment*, pages 95351A–95351A. International Society for Optics and Photonics, 2015.

[5] J. Fan, Z. Chen, L. Yan, J. Gong, and D. Wang. Research on earthquake prediction from infrared cloud images. In *Ninth International Symposium on Multispectral Image Processing and Pattern Recognition (MIPPR2015)*, pages 98150E–98150E. International Society for Optics and Photonics, 2015.

[6] E. Florido, F. Martínez-Álvarez, A. Morales-Esteban, J. Reyes, and J. Aznarte-Mellado. Detecting precursory patterns to enhance earthquake prediction in chile. *Computers & Geosciences*, 76:112–120, 2015.

[7] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[8] M. Hayakawa. Earthquake prediction with electromagnetic phenomena. In *THE IRAGO CONFERENCE 2015: 360 Degree Outlook on Critical Scientific and Technological Challenges for a Sustainable Society*, volume 1709, page 020002. AIP Publishing, 2016.

[9] M. Hayakawa, H. Yamauchi, N. Ohtani, M. Ohta, S. Tosa, T. Asano, A. Schekotov, J. Izutsu, S. M. Potirakis, and K. Eftaxias. On the precursory abnormal animal behavior and electromagnetic effects for the kobe earthquake (m˜ 6) on april 12, 2013. *Open Journal of Earthquake Research*, 5(03):165, 2016.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] M. Jiang. Easily magnetic anomalies earthquake prediction. In *MATEC Web of Conferences*, volume 63, page 01020. EDP Sciences, 2016.

[12] S. Kannan. Innovative mathematical model for earthquake prediction. *Engineering Failure Analysis*, 41:89–95, 2014.

[13] V. Korepanov. Possibility to detect earthquake precursors using cubesats. *Acta Astronautica*, 128:203–209, 2016.

[14] M. Last, N. Rabinowitz, and G. Leonard. Predicting the maximum earthquake magnitude from seismic data in israel and its neighboring countries. *PloS one*, 11(1):e0146101, 2016.

[15] C. Li and X. Liu. An improved pso-bp neural network and its application to earthquake prediction. In *Control and Decision Conference (CCDC), 2016 Chinese*, pages 3434–3438. IEEE, 2016.

[16] J. Mahmoudi, M. A. Arjomand, M. Rezaei, and M. H. Mohammadi. Predicting the earthquake magnitude using the multilayer perceptron neural network with two hidden layers. *Civil Engineering Journal*, 2(1):1–12, 2016.

[17] M. Moustra, M. Avraamides, and C. Christodoulou. Artificial neural networks for earthquake prediction using time series magnitude data or seismic electric signals. *Expert systems with applications*, 38(12):15032–15039, 2011.

[18] S. Narayanakumar and K. Raja. A bp artificial neural network model for earthquake magnitude prediction in himalayas, india. *Circuits and Systems*, 7(11):3456, 2016.

[19] A. Panakkat and H. Adeli. Recurrent neural network for approximate earthquake time and location prediction using multiple seismicity indicators. *Computer-Aided Civil and Infrastructure Engineering*, 24(4):280–292, 2009.

[20] G. Pararas-Carayannis. The earthquake of may 12, 2008 in the sichuan province of china. *Website: http://www. drgeorgepc. com/Earthquake2008ChinaSichuan. html*, 2008.

[21] S. Saba, F. Ahsan, and S. Mohsin. Bat-ann based earthquake prediction for pakistan region. *Soft Computing*, pages 1–9, 2016.

[22] J. Shore and R. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on information theory*, 26(1):26–37, 1980.

[23] G. A. Sobolev. Methodology, results, and problems of forecasting earthquakes. *Herald of the Russian Academy of Sciences*, 85(2):107–111, 2015.

[24] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

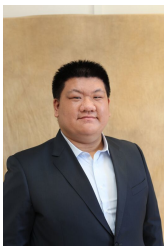
[25] J. Thomas, F. Masci, and J. Love. On a report that the 2012 m6: 0 earthquake in italy was predicted after seeing an unusual cloud formation. *Natural Hazards and Earth System Sciences (NHESS)*, 2015.

[26] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[27] X. Wang, L. Gao, and S. Mao. Phasefi: Phase fingerprinting for indoor localization with a deep learning approach. In *Global Communications Conference (GLOBECOM), 2015 IEEE*, pages 1–6. IEEE, 2015.

[28] X. Wang, L. Gao, S. Mao, and S. Pandey. Csi-based fingerprinting for indoor localization: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 2016.

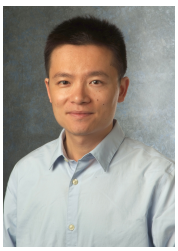
Qianlong Wang received the B.E. degree in Electrical Engineering from Wuhan University, China, in 2013, and the M.E. degree in Electrical and Computer Engineering from Stevens Institute of Technology, Hoboken, NJ, in 2015, respectively. He is currently working toward the PhD degree in the Department of Electrical Engineering and Computer Science, Case Western Reserve University. His research interests include big data, and deep learning. He is a student member of the IEEE.



Yifan Guo received the B.S. degree in Information and Computing Sciences from Beijing University of Posts and Telecommunications, China, in 2013, and the M.S. degree in Computer Science from Northwestern University, Evanston, in 2015, respectively. He is currently working toward the PhD degree in the Department of Electrical Engineering and Computer Science, Case Western Reserve University. His research interests include artificial intelligence, machine learning and big data. He is a student member of the IEEE.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2699169, IEEE Transactions on Emerging Topics in Computing

Lixing Yu received the B.S. degree in Electrical Engineering from Beijing Institute of Technology, China, in 2012, and the M.S. degree in Electrical Engineering from the George Washington University, Washington DC, in 2014, respectively. He is currently working toward the PhD degree in the Department of Electrical Engineering and Computer Science, Case Western Reserve University. His research interests include deep learning, and data mining. He is a student member of the IEEE.



Pan Li (S'06-M'09) received the B.E. degree in Electrical Engineering from Huazhong University of Science and Technology, Wuhan, China, in 2005, and the Ph.D. degree in Electrical and Computer Engineering from University of Florida, Gainesville, in 2009, respectively. He is currently an Associate Professor in the Department of Electrical Engineering and Computer Science at Case Western Reserve University. His research interests include network science and economics, energy systems, security and privacy, and big data. He has been serving as an Editor for IEEE Wireless Communications Letters, IEEE Journal on Selected Areas in Communications – Cognitive Radio Series, and IEEE Communications Surveys and Tutorials, a Feature Editor for IEEE Wireless Communications, and a Technical Program Committee (TPC) Co-Chair for Ad-hoc, Mesh, Machine-to-Machine and Sensor Networks Track, IEE VTC 2014, Physical Layer Track, Wireless Communications Symposium, WTS 2014, and Wireless Networking Symposium, IEEE ICC 2013. He received the NSF CAREER Award in 2012 and is a member of the IEEE and the ACM.