# Chapter 10
# A Novel Method Based on Data Visual Autoencoding for Time-Series Classification

**Chen Qian, Yan Wang and Lei Guo**

**Abstract** A variety of techniques based on numerical characteristics are currently presented for mining time-series data. However, we find that time-series data generally contain curves sharing some set of visual characteristics and features. These characteristics offer a deeper understanding of time-series data, and open up a potential new technique for time-series analysis. Particularly beneficial from recent advances in deep neural networks, representations and features can be automatically learnt by deep learning architectures such as autoencoders. Based on that, our work proposes a novel method, named time-series visualization (TSV), to efficiently detect visual characteristics from curves of time-series data and use these characteristics for intelligent analysis. Architecture and algorithm of TSV based on stacked autoencoders are introduced in this paper. Further, important factors affecting the performance of TSV are discussed based on empirical results. Through empirical evaluation, it is demonstrated that TSV has better efficiency and higher classification accuracy on analyzing the datasets with significant curve feature.

**Keyword** Time series · Autoencoder · Classification · Input dropout · TSV

## 10.1 Introduction

In the last decade, interest in mining time-series data is like an explosion which, in turn, resulted in lots of researches proposed to introduce new techniques to index, classify, cluster, and segment time series. However, most of these techniques have limited performance because the form of time-series data is inconstant but their focus is mainly on numerical characteristics of data.

Similarity measure is one of the most important ways toward mining time-series data. The most straightforward similarity measure for time series is the Euclidean

C. Qian · Y. Wang (✉) · L. Guo
School of Automation Science and Electrical Engineering,
Beihang University, Beijing 100191, People's Republic of China
e-mail: w-yan@buaa.edu.cn

Z. Deng and H. Li (eds.), *Proceedings of the 2015 Chinese Intelligent Automation Conference*, Lecture Notes in Electrical Engineering 336,
DOI 10.1007/978-3-662-46469-4_10

Distance (ED) [1], which has two advantages: Linear complexity and parameter-free. However, ED is quite sensitive to noise and misalignments which means it is unable to handle the time-shifting series. Inspired by this motivation, Berndt and Clifford [2] introduced dynamic time warping (DTW) which can be used to measure the similarity between time series with local shifts. However, DTW is too slow to be of practical use, even though it provides good measuring accuracy [3]. Thus, many methods have been proposed based on DTW to improve the efficiency of DTW [4–6]. In addition, longest common subsequences (LCSS) was proposed based on the model introduced by André-Jönsson and Badal [7], which was another group of similarity measures. Other famous examples for this category include edit distance on real sequence (EDR) [8] and edit distance with real penalty (ERP) [9]. However, most of these similarity measures focus on numerical characteristics, which make them quite sensitive to changes of time-series data.

Interestingly, for human, it is intuitional to identify the similarity of time series through curves rather than the real data. Based that, we think if it is possible to get a good representation of time series from the curves and use it for time-series analysis. Fortunately, autoencoder provides a potential way to achieve that autoencoder is a learning circuit to encode the inputs into some representations that are as close as possible to outputs [10]. It was first proposed by Hinton and his group in the 1980s, but with the recent revival of interest in "deep networks," [11] autoencoder is coming back to the center stage. We believe that autoencoder and human visual system are quite similar in some aspects [12]; and in this paper, we try to construct a new method named time-series visualization (TSV) for time-series classification based on good representations learnt from curves of time-series data. First, representations can be learnt by stacked autoencoder (SAE) during the pretraining process. Since the learning performance of normal SAE architecture seems not good enough for image patches, dropout is introduced into input layer to reduce model complexity. Experimental results show that input dropout improves 86 % training accuracy and reduces 16 % running time. Second, a normal neural network classifier is trained by using encoding weights as initial connect weights between input layer and hidden layer during the training process. Finally, the trained classifier can be applied to classify similar time-series data.

The remainder of this paper is organized as follows: Section. 10.2 discusses the architecture of SAE with an input dropout that is used in the rest of the paper. Section 10.3 describes the architecture of TSV. Then, the experimental results and comparisons are presented on classification for time-series data. Finally, Sect. 10.4 presents our conclusions.

## 10.2 The Architecture of TSV for Time-Series Classification

The architecture of TSV for time-series classification is given in Fig. 10.1 intuitively.

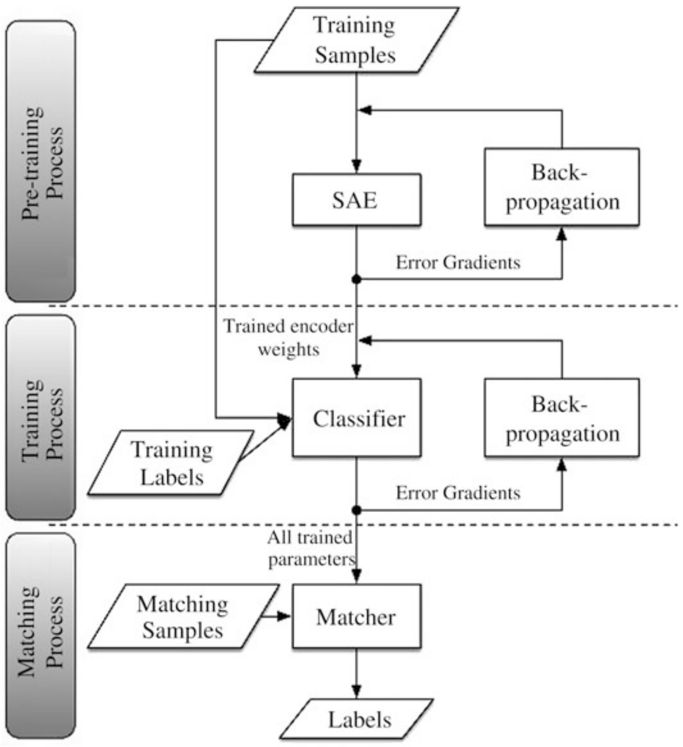

**Fig. 10.1** The architecture of TSV for time-series classification

Three processes are executed successively in an overall similarity matching of TSV. It is worth noting that not all the parameters trained from the previous level will be transported to the next level as shown in Fig. 10.1. For example, only the trained weights between input layer and hidden layer of SAE will be used in the next training of the NN classifier.

Here we define an input $X = [x_1, x_2 \ldots x_n]^T$, and the autoencoder transforms the input $X$ into an output $Y = [y_1, y_2 \ldots y_n]^T$ with learnt representation. In order to drive a general architecture of autoencoder network, a three-layer neural network architecture is applied. However, autoencoders are distinguished from more general neural networks by the fact that their outputs are desired to be the same to their inputs. The hidden layer detects features in input data. Then the corresponding decoder takes encodings, and attempts to reconstruct the original input.

Dropout was proposed by Hinton et al. [13] as an approach to improve the performance of fully connected neural network layers. When the dropout is applied in a fully connected layer, each element of the layer is kept with probability $p$, otherwise set to 0 with probability $(1 - p)$. Our dropout algorithm is modified by introducing dropout probability $(1 - p)$ into input layer. The architecture and introduction of SAE with input dropout is specified below.

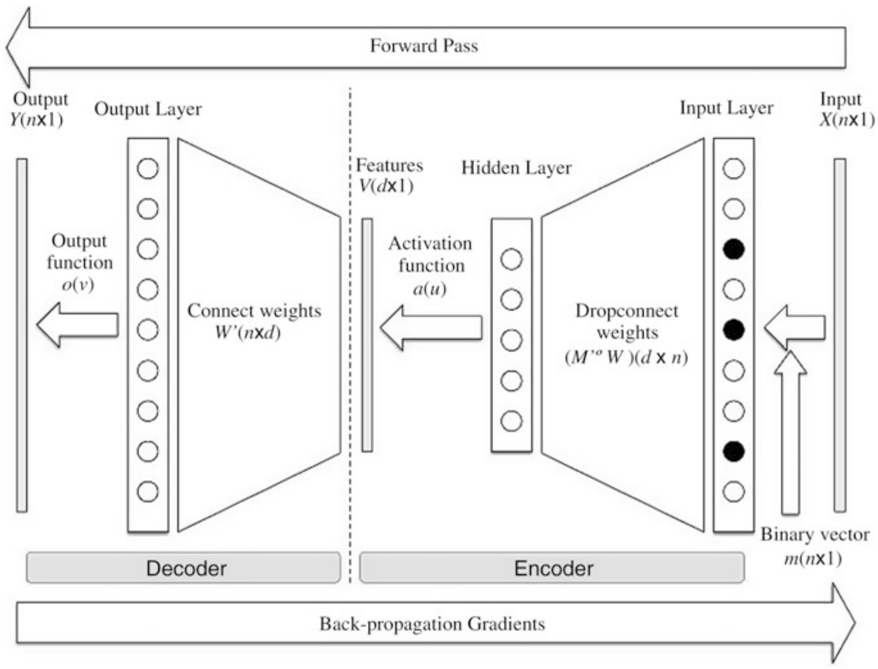

**Fig. 10.2** The architecture of SAE with input dropouts

*Encoder*. The deterministic mapping $f_\theta$ that transforms an input into hidden features is called the encoder [14]. Each input vector $X$ and the weight matrix $W$ followed by a nonlinear activation function $a(u)$ such as tanh, sigmoid, or relu, which can be expressed as (Fig. 10.2):

$$V = f_\theta(X) = a(WX + b),$$

where $V$ is a feature matrix extracted by the encoder. $f_\theta$ is an affine mapping and its parameter set is $\theta = \{W, b\}$, where $W$ is a $d \times n$ weight matrix and $b$ is an offset vector of hidden dimensionality $d$. Because we have introduced dropout probability $(1 - p)$ into input layer, $f_\theta$ can be rewritten as

$$V = a(W(mX) + b) = a((M' \circ W)X + b),$$

where $m$ is a binary vector of size $n$ with each element $m_j$ drawn independently from Bernoulli$(p)$, and $M'$ is a $d \times n$ drop connect weight matrix with a same binary value in each row. Then we made an approximation

$$\sum_{M'} a((M' \circ W)X + b) \approx a(\sum_{M'} ((M' \circ W)X + b))$$

**Table 10.1** Training SAE with input dropouts

| |
|---|
| Initialization: input $X \rightarrow$ image format $\rightarrow$ stacked vector $X'$, initialize parameters $\{\theta_0, \theta'_0\}$, and learning rate $\eta$. |
| Input: Randomly selected input $x'$ and parameters $\{\theta_{t-1}, \theta'_{t-1}\}$ from step $t-1$. |
| **Forward Pass:** |
| Select randomly drawn mask $m$: $m \sim \text{Bernoulli}(p)$ |
| Compute hidden features: $v = a(w(x' \circ m) + b)$ |
| Compute output: $y = o(w'v + b')$ |
| **Back-propagation Gradients:** |
| Compute the loss function of decoding layer and encoding layer and, respectively, expressed as $L'_{W'}$ and $L'_W$. |
| Update weights of decoding layer: $W'_t = W'_{t-1} - \eta L'_{W'}$ |
| Update weights of encoding layer: $W_t = W_{t-1} - \eta(M' \circ L'_W)$ |

*Decoder*. The decoder mapping $g_{\theta'}$ is used to reconstruct the output $Y$. It can be regarded as a reverse process of encoder. Thus, the decoder can be expressed as follows with its appropriately sized parameters $\theta' = \{W', b'\}$.

$$Y = g_{\theta'}(V) = o(W'V + b')$$

According to informax principle put forward by Linsker [15], a good representation is to retain a significant amount of information from the input, which means to learn parameters $\{\theta, \theta'\}$ that minimize the overall distortion function expressed as follows:

$$\min E(X, Y) = \min_{\theta,\theta',M} \sum_{i=1}^{n} \Delta(X, Y; \theta, \theta', M') = \min_{\theta,\theta'} \sum_{M} \left(p(M') \sum_{i=1}^{n} \Delta(X, Y; \theta, \theta')\right)$$

Once the randomly drawn mask $m$ is chosen, it is applied to train the parameters $\{\theta, \theta'\}$ via stochastic gradient descent (SGD) by back-propagation gradients of the loss function. Specific calculation steps of SGD training with input dropout are provided in Table 10.1.

## 10.3 Experiments on TSV: Classification for Time-Series Data

In this section, we experimentally evaluate the performance of TSV on time-series classification. Four benchmarks are applied to perform comparisons for classification accuracy with some other measures such as ED, DTW, EDR, and LCSS, which are used in references [3, 16, 17]. These four datasets contain curve features, including the popular CBF dataset, ECG200 dataset, synthetic control dataset, and trace dataset.

For a fair comparison, we keep all parameters of TSV invariable for four datasets, which is 900 input nodes with 70 % dropouts, 100 hidden nodes, learning rate $\eta = 1$, and we trained SAE 3000 epochs and classifier 1000 epochs, respectively. Note that, the performance of TSV may not be in the best situation for every datasets we tested; however, our focus is not the best performance of TSV on a specific dataset, but a robust performance on all datasets.

Since the number of input nodes cannot be changed, resizing each image patch into a certain size is necessary. Here we restrict the size of image patch to $30 \times 30$, which seems to be a draconian restriction to some datasets like trace dataset whose length is 275. However, we are surprised to see that results of trace are all correct. Because sometimes, the dimensionality of time series is very high and details is not the key for detecting the feature of time-series data, resizing provides a way to reduce the dimensionality of data but keep the most important visual characteristics. In this way, autoencoder can be trained with lower training error and faster running time. Furthermore, dimensionality of time series is reduced to 100 hidden layer output. It seems like that curves sharing the general characteristics of time-series data are separated by the autoencoder and each training sample can be expressed as a combination of these curves. Hence, if the number of hidden nodes is chosen properly, dimensionality of time-series data can be significantly reduced.

Training image pitches of time-series samples and visualization of weights learnt by SAE with input dropout is shown in Fig. 10.3. And we compared training error and running time under different dropout probabilities. Further, the error ratios of all methods based on four benchmarks are shown in Table 10.2.

In summary, SAE with 70 % input dropout has the lowest train error and it is one of the fastest algorithms according to the running time comparison as in Fig. 10.4. It improves 86 % training accuracy and reduces 16 % running time. And from

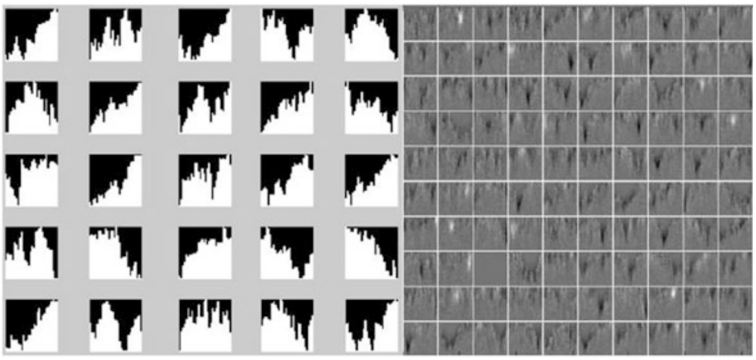

**Fig. 10.3** Training image pitches of time-series samples and visualization of weights learnt by SAE with input dropout

**Table 10.2** Error ratio of different methods

| | CBF | Synthetic control | ECG200 | Trace |
|---|---|---|---|---|
| 1-NN Euclidean distance | 0.087 | 0.143 | 0.16 | 0.36 |
| 1-NN DTW | 0.003 | **0.02** | 0.23 | 0.02 |
| 1-NN EDR | 0.013 | 0.117 | 0.21 | 0.15 |
| 1-NN ERP | **0** | 0.037 | 0.21 | 0.08 |
| 1-NN LCSS | 0.017 | 0.06 | 0.17 | 0.12 |
| SVM Euclidean distance | 0.123 | 0.0767 | 0.19 | 0.27 |
| TSV | 0.004 | 0.023 | **0.15** | **0** |

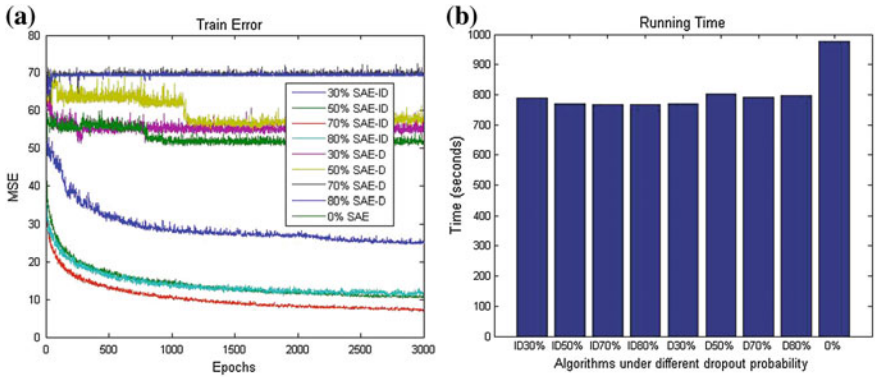

**Fig. 10.4** Comparison of the training error and running time under different dropout probabilities

Table 10.2, we can see 1-NN ERP get the best effect on CBF data, but DTW and TSV have a very close performance. However, 1-NN DTW takes the first place on synthetic control dataset; in spite of this, these three methods are still quite close. TSV performs best on last two datasets, especially the trace dataset on which error ratio is zero, and 1-NN ERP and 1-NN DTW are inferior on the ECG200 dataset. Through experiments, we find that there is no clear evidence that one classification method tested is superior to others in all dataset tests in terms of accuracy. While TSV is a little bit more effective generally on the four datasets we used, some methods like 1-NN ERP and 1-NN DTW are superior on certain datasets but inferior on some other datasets. Hence, we believe that TSV is a more effective method compared to existing methods on the dataset with significant curve feature.

## 10.4 Conclusion

In this paper, we have presented a novel method called TSV to improve the performance of classification for time-series data on the datasets with a significant curve feature. We have clearly introduced the full architecture and algorithms of TSV, including SAE with input dropouts and a normal NN classifier. Then, we evaluate the performance of TSV for time-series classification on four popular benchmarks. In this experiment, the results show that the performance of TSV is quite good in almost every datasets, which demonstrates that TSV is an effective method for time-series classification.

**Acknowledgments** This work was supported by Key Project of Chinese National Programs for Fundamental Research and Development (973 program) under Grant (2012CB720003), the National Natural Science Foundation of China (Nos. 91016004, 61004023, 61127007, 61174069) and the International S&T Cooperation Program of China (No. 2013DEE13040).

## References

1. Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases. ACM
2. Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. KDD Workshop 10(16):359–370
3. Wang X, Mueen A, Ding H et al (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Disc 26(2):275–309
4. Yi BK, Jagadish HV, Faloutsos C (1998) Efficient retrieval of similar time sequences under time warping. In: IEEE 14th international conference on data engineering, pp 201–208
5. Kim SW, Park S, Chu WW (2001) An index-based approach for similarity search supporting time warping in large sequence databases. In: IEEE 17th international conference on data engineering, pp 607–614
6. Keogh E, Ratanamahatana CA (2005) Exact indexing of dynamic time warping. Knowl Inf Syst 7(3):358–386
7. André-Jönsson H, Badal DZ (1997) Using signature files for querying time-series data. Principles of data mining and knowledge discovery. Springer, Heidelberg, pp 211–220
8. Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data, ACM, pp 491–502
9. Chen L, Ng R (2004) On the marriage of lp-norms and edit distance. In: Proceedings of the thirtieth international conference on very large data bases-vol 30, VLDB Endowment, pp 792–803
10. Baldi P (2012) Autoencoders, unsupervised learning, and deep architectures. ICML unsupervised and transfer learning, pp 37–50
11. Hinton G, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
12. Arel I, Rose DC, Karnowski TP (2010) Deep machine learning-a new frontier in artificial intelligence research. Comput Intell Mag IEEE 5(4):13–18
13. Hinton GE, Srivastava N, Krizhevsky A, et al (2012) Improving neural networks by preventing co-adaptation of feature detectors arXiv:1207.0580
14. Vincent P, Larochelle H, Lajoie I et al (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach Learn Res 11:3371–3408
15. Linsker R (1989) How to generate ordered maps by maximizing the mutual information between input and output signals. Neural Comput 1(3):402–411
16. Keogh E, Kasetty S (2003) On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Min Knowl Disc 7(4):349–371
17. Ding H, Trajcevski G, Scheuermann P et al (2008) Querying and mining of time series data: experimental comparison of representations and distance measures. Proc VLDB Endowment 1 (2):1542–1552