Neurocomputing 184 (2016) 232–242

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Auto-encoder based dimensionality reduction

Yasi Wang, Hongxun Yao *, Sicheng Zhao

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

ARTICLE INFO

Article history:
Received 22 January 2015
Received in revised form
3 August 2015
Accepted 4 August 2015
Available online 27 November 2015

Keywords:
Auto-encoder
Dimensionality reduction
Visualization
Intrinsic dimensionality
Dimensionality-accuracy

ABSTRACT

Auto-encoder—a tricky three-layered neural network, known as auto-association before, constructs the "building block" of deep learning, which has been demonstrated to achieve good performance in various domains. In this paper, we try to investigate the dimensionality reduction ability of auto-encoder, and see if it has some kind of good property that might accumulate when being stacked and thus contribute to the success of deep learning.

Based on the above idea, this paper starts from auto-encoder and focuses on its ability to reduce the dimensionality, trying to understand the difference between auto-encoder and state-of-the-art dimensionality reduction methods. Experiments are conducted both on the synthesized data for an intuitive understanding of the method, mainly on two and three-dimensional spaces for better visualization, and on some real datasets, including MNIST and Olivetti face datasets. The results show that auto-encoder can indeed learn something different from other methods. Besides, we preliminarily investigate the influence of the number of hidden layer nodes on the performance of auto-encoder and its possible relation with the intrinsic dimensionality of input data.

1. Introduction

Artificial neural network is a mathematical and computational model composed of a large number of neurons that can simulate the structural and functional characteristics of biological neural network. It is a self-adaptive system, which changes the internal structure according to the external input, and is commonly used to model the complex relationship between input and output. The development of artificial neural network is a tortuous road. Perceptron is the starting point of modern neural computation, and the proof of perceptron convergence theorem in 1962 triggered the first climax of research for neural network represented by perceptron. In 1965, Minsky and Papert [2,20] pointed out the defects of perceptron and took a pessimistic view on the research of neural network, which made neural network study from rise into stagnation. By the early 1980s, related work by Hopfield et al. [16] showed the potential of neural network which made neural network study from stagnation to boom. In the mid-1990s, with the advent of the support vector machine (SVM) [10], researchers realized some limitations of artificial neural network and the research for neural network fell into low tide period again.

Auto-encoder, known as auto-association before, is a tricky three-layered neural network and was studied by a number of researchers in 1990s. Bourlard and Kamp [7] discussed the relationship between auto-association by multi-layer perceptrons and singular value decomposition in 1988. They showed that for auto-association with linear output units, the optimal weight values could be derived by purely linear techniques relying on singular value decomposition and low rank matrix approximation. In 1991, Kramer [18] introduced how to conduct nonlinear principal component analysis using auto-associative neural networks with three hidden layers. Due to the difficulty in training, deep network with multi-layer stacked auto-encoders did not exert its superior strength for a long time. Some researchers committed themselves to investigating several fundamental issues of neural networks with one or two hidden layers [8,1,22,23,3,9,17,25,21,27].

Until 2006, Geoffrey Hinton [15] solved this problem in Science Magazine to a large extent and broke the stalemate that neural network was in low tide. The method was to do layer-wise pre-training [14] to multi-layer auto-encoders using RBM [13]. That is the very hot topic in recent years—deep learning. From then on, deep learning sweeps across the industry and the academia like a wave. Recent research results have also demonstrated that deep learning indeed achieves state-of-the-art performances among various areas [4–6]. Particularly, deep learning has already been successfully applied to the industry in speech area. In the field of image analysis, there are also many good results. By building a

* Corresponding author.
E-mail addresses: wangyasi@hit.edu.cn (Y. Wang), h.yao@hit.edu.cn (H. Yao), zsc@hit.edu.cn (S. Zhao).

http://dx.doi.org/10.1016/j.neucom.2015.08.104
0925-2312/

9-layered locally connected sparse auto-encoder, Le et al. [24] discovered that the network was sensitive to high-level concepts such as cat faces. Krizhevsky et al. [19] attained record-breaking performance on the ImageNet in 2012 with a large and deep convolutional neural network. Zeiler et al. [26] demonstrated state-of-the-art performance on Caltech-101 and Caltech-256 datasets. Zhu et al. [28] extracted face identity-preserving features from an image under any pose and illumination which could be used to reconstruct the face image in canonical view by designing a deep network.

Although achieving comparable performance and widely applied, deep learning is kind of like a "black-box" actually and there is no very sufficient and strict theoretical system to support. So a problem is: we have impressive performance using deep learning but we do not know why theoretically. In deep learning, a number of researchers tend to make progress by employing increasingly deep models and complex unsupervised learning algorithms.

This paper comes from the idea that whether auto-encoder has some kind of good property which might accumulate when being stacked and thus contribute to the success of deep learning. We start from a "building block" of deep learning—auto-encoder and focus on its dimensionality reduction ability. When restricting the number of hidden layer nodes less than the number of original input nodes in an auto-encoder, the desired dimensionality reduction effect can be achieved.

Based on the above analysis, the main contributions of the paper can be summarized as follows:

1. We start from auto-encoder and focus on its ability to reduce the dimensionality, trying to understand the difference between auto-encoder and state-of-the-art dimensionality reduction methods. The results show that auto-encoder indeed learn something different from other methods.
2. We preliminarily investigate the influence of the number of hidden layer nodes on the performance of auto-encoder on MNIST and Olivetti face datasets. The results reveal its possible relation with the intrinsic dimensionality.

On the whole, we expect that analyzing the fundamental methods in deep learning, e.g. auto-encoder, could help us understand deep learning better.

2. Auto-encoder based dimensionality reduction

In this section, we briefly introduce auto-encoder, four representative dimensionality reduction methods and the concept of dimensionality reduction and intrinsic dimensionality. Then we focus on auto-encoder's dimensionality reduction ability, and investigate the influence of the number of hidden layer nodes in auto-encoder.

2.1. Auto-encoder

Suppose the original input $\boldsymbol{x}$ belongs to n-dimensional space and the new representation $\boldsymbol{y}$ belongs to m-dimensional space, an auto-encoder is a special and tricky three-layered neural network in which we set the output $h_{W,b}(\boldsymbol{x}) = (\tilde{x_1}, \tilde{x_2}, \ldots, \tilde{x_n})^T$ equal to the input $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. J is the reconstruction error. It is an unsupervised learning algorithm and uses back propagation algorithm for training

$$h_{W,b}(\boldsymbol{x}) = g(f(\boldsymbol{x})) \approx \boldsymbol{x}$$

$$J(W, b; \boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2} \| h_{W,b}(\boldsymbol{x}) - \boldsymbol{y} \|^2$$

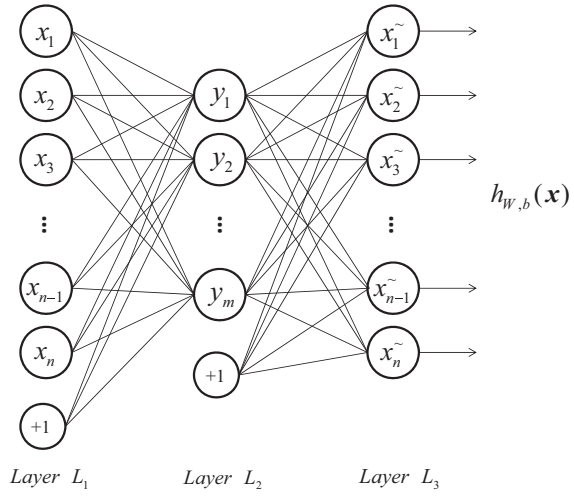


Fig. 1. The structure of auto-encoder.

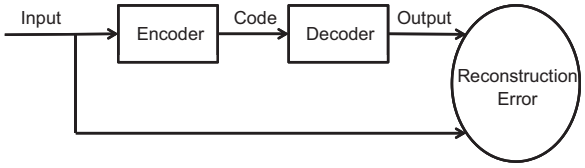


Fig. 2. The visualization description of auto-encoder.

Figs. 1 and 2 show the structure and the visualization description of an auto-encoder respectively. As shown in Fig. 2, from the first layer to the second layer amounts to an encoder f and from the second layer to the third layer amounts to a decoder g. We then minimize the reconstruction error J by adjusting parameters in the encoder and the decoder to get the code.

Auto-encoder can be seen as a way to transform representation. When restricting the number of hidden layer nodes m greater than the number of original input nodes n and adding sparsity constraint, the result will be similar to sparse coding. When restricting the number of hidden layer nodes m less than the number of original input nodes n, we can get a compressed representation of the input, which actually achieves desired dimensionality reduction effect. This paper mainly focuses on the latter case.

The following is a very quick introduction of four representative dimensionality reduction methods [12], which are to be used to compare with auto-encoder.

PCA: Principal component analysis is a very popular linear technique for dimensionality reduction. Given a dataset on $\mathbb{R}^n$, PCA aims to find a linear subspace of dimension d lower than n which attempts to maintain most of the variability of the data.

LDA: Linear discriminant analysis is another popular linear dimensionality reduction method. The basic idea is to ensure the samples after projection to have maximum-between-cluster-distance and minimum-in-cluster-distance in the new subspace.

LLE: Locally linear embedding is a nonlinear approach to reduce dimensionality by computing low-dimensional, neighborhood preserving embedding of high-dimensional data. A dataset of dimensionality n, which is assumed to lie on or near a smooth nonlinear manifold of dimensionality $d < n$, is mapped into a single global coordinate system of lower dimensionality d. The global nonlinear structure is recovered by locally linear fits.

Isomap: Isomap is a nonlinear generalization of classical multidimensional scaling. The main idea is to perform multidimensional scaling, not in the input space, but in the geodesic space of the nonlinear data manifold. The geodesic distance

represents the shortest paths along the curved surface of the manifold measured as if the surface were flat. This can be approximated by a sequence of short steps between neighboring sample points.

2.2. Dimensionality reduction ability of auto-encoder

As a kind of common rule, superficially high-dimensional and complex phenomena can actually be dominated by a small amount of simple variables in most situations. Dimensionality reduction is an old and young, dynamic research topic [11,12]. It is looking for a projection method that maps the data from high feature space to low feature space. Dimensionality reduction methods in general can be divided into two categories, linear and nonlinear.

This paper describes auto-encoder's dimensionality reduction ability by comparing auto-encoder with several linear and nonlinear dimensionality reduction methods in both a number of cases from two-dimensional and three-dimensional spaces for more intuitive results and real datasets including MNIST and Olivetti face datasets. More details and results can be seen in experiment part.

2.3. The number of hidden layer nodes in auto-encoder & intrinsic dimensionality

In the process of dimensionality reduction, discarding some dimensions inevitably leads to the loss of information. Thus the primary problem to be solved is to keep the main and important characteristics of the original data as much as possible. So the process of dimensionality reduction is closely related to original data characteristics.

Intrinsic dimensionality, as an important intrinsic characteristic of high-dimensional data, can reflect the essential dimension of data well. Sample data in high-dimensional space generally cannot diffuse in the whole space; they actually lie in a low-dimensional manifold embedded in high-dimensional space and the dimensionality of the manifold is the intrinsic dimensionality of the data.

This paper preliminarily investigates the influence of the number of hidden layer nodes on the performance of auto-encoder on MNIST and Olivetti face datasets by recording the change of performance of the classifier when the dimensionality of the projected representation varies. More details and results can be seen in experiment part.

3. Experiment

3.1. Experimental setup

There are mainly two parts of experiments included. The first part is conducted both on the synthesized data and real datasets to evaluate auto-encoders dimensionality reduction ability. The second part is conducted on real datasets to investigate the influence of the number of hidden layer nodes on performance of auto-encoder.

3.1.1. Dimensionality reduction

In this subsection, we evaluate auto-encoder's dimensionality reduction ability compared with several popular dimensionality reduction methods in a number of cases. We pick PCA and LDA as representative of linear methods, LLE and Isomap as representative of nonlinear methods.

On synthesized data: The experiments are conducted on synthesized data in two-dimensional and three-dimensional spaces because in these situations we can get better visualization results for an intuitive understanding of auto-encoder. Fig. 3 shows two cases from two-dimensional space and Fig. 4 shows three transformations of helix cases from three-dimensional space. In these cases, we use in total five methods including auto-encoder to do the dimensionality reduction, i.e.

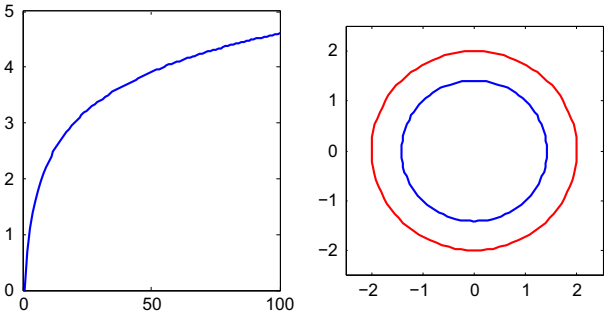


Fig. 3. Two cases from two-dimensional space.

$$\mathbb{R}^2 \rightarrow \mathbb{R}^1$$

$$\mathbb{R}^3 \rightarrow \mathbb{R}^1$$

to see how the results auto-encoder get differ from the other four dimensionality reduction methods. After the dimensionality reduction, the original points from two or three-dimensional spaces are projected into one-dimensional space and the data obtained are zoomed to a certain range further to have better visual quality. Results can been found in Section 3.2.

On real datasets: MNIST is a classical dataset of handwritten digits with 60,000 training images and 10,000 testing images. Every image is normalized and centralized to unified size. Image size is 28 by 28. Olivetti face dataset contains 400 images from 40 different people, 10 images for each. Image size is 64 by 64, we first resize the image from 64*64 to 28*28 before the real process to speed up.

The images in both two datasets are grayscale images. Fig. 5 shows a number of images from MNIST and Olivetti face datasets.

Take pixel intensity as the original representation and use auto-encoder and PCA to do the dimensionality reduction. When using auto-encoder, we visualize the function learned by auto-encoder by weights between the first layer and the second layer to see what does auto-encoder exactly learn. Besides, when the dimensionality of the transformed new representation is 2 or 3, i.e., when the original image is projected to two or three-dimensional spaces, we can visualize the comparison results between auto-encoder and PCA. Results can been found in Section 3.2.

3.1.2. Influence of the number of hidden layer nodes

In this subsection, we preliminarily investigate the influence of the number of hidden layer nodes on the performance of auto-encoder on MNIST and Olivetti face datasets.

For MNIST dataset, every image has a total of 28*28 = 784 pixels and the intrinsic dimensionality is known to be 10. Use auto-encoder to reduce the dimensionality from 784 to s, i.e.

$$\mathbb{R}^{784} \rightarrow \mathbb{R}^{s}$$

and do the classification with softmax classifier. Record the change of performance of the classifier when the dimensionality of the new representation s varies.

The same goes for the Olivetti face dataset. Results can been found in Section 3.2.

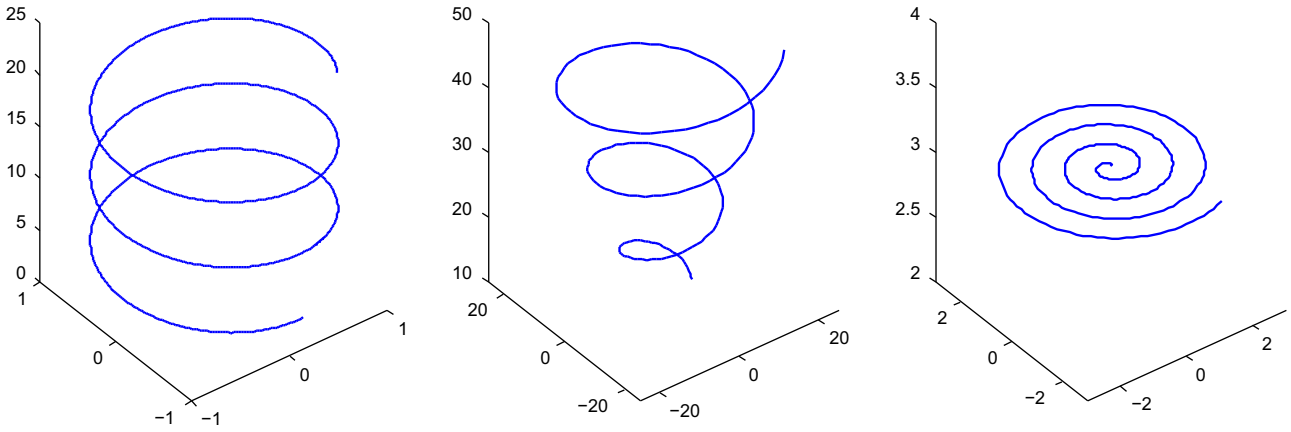


Fig. 4. Three cases from three-dimensional space.

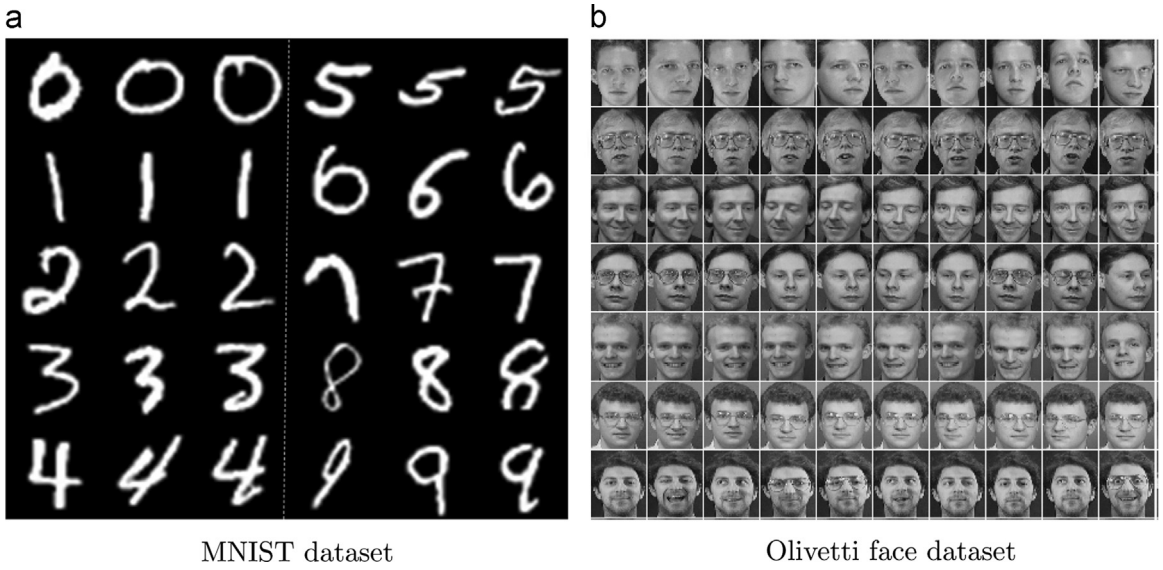


Fig. 5. Images from two datasets: (a) MNIST dataset and (b) Olivetti face dataset.

3.2. Dimensionality reduction

3.2.1. On synthesized data from $\mathbb{R}^2$ to $\mathbb{R}^1$

In this subsection, we implement five methods within two different cases in two-dimensional space to see how auto-encoder differs from other dimensionality reduction methods.

The first case: In the first and simplest logarithmic curve case, all methods do a good job. Results of PCA, LDA and Isomap are stable, as shown in Fig. 6. Results of LLE and auto-encoder are relatively not that stable, which means when you run the program twice you may get two different results. Two examples of LLE and auto-encoder are shown in Fig. 7. Although the results are not stable, most of the results can be seen as "acceptable", which means that the points after dimensionality reduction are monotonic increasing or decreasing, thus keeping the geometry characteristic of the original points.

In this case, it seems that auto-encoder performs no different from other four methods, but go a little more deeply, you will find some difference.

As can be seen from Figs. 6 and 7, there are 100 two-dimensional points $(x_1, x_2, \ldots x_{100})$ in the logarithmic curve indicated by red '+' ; we use five dimensionality reduction methods to reduce the dimensionality and project the original points to 100 one-dimensional points $(y_1, y_2, \ldots y_{100})$ on the horizontal axis indicated by black dots.

Use *dis* to represent the distance between two points and use *dif* to denote the difference values between each two adjacent points, i.e.,

$$dif(i) = y_{i+1} - y_i = dis(y_i, y_{i+1}) \iff dis(x_i, x_{i+1})$$

Since we want to evaluate the quality of the dimensionality reduction, it is appropriate to use $dis(y_i, y_{i+1})$ to measure $dis(x_i, x_{i+1})$. *dif* is a unary array variable and has 99 values in it. We get five *dif*s for five methods, and draw a bar graph as shown in Fig. 8.

From Fig. 8, we can find the following points: (1) for PCA, LLE and Isomap three methods, *dif* is close to a constant. Since we use new projection points to represent the original points, this means in these three methods, $dis(x_1, x_2)$, $dis(x_{50}, x_{51})$ and $dis(x_{99}, x_{100})$ are almost the same. That, however, is not the case. The distances between each two adjacent points in the logarithmic curve should not be regarded as the same. It is more obvious for $dis(x_1, x_2)$ and $dis(x_{99}, x_{100})$. (2) For LDA, *dif* is monotonic decreasing which means $dis(x_1, x_2)$ is much larger than $dis(x_{99}, x_{100})$. (3) For auto-encoder, *dif*

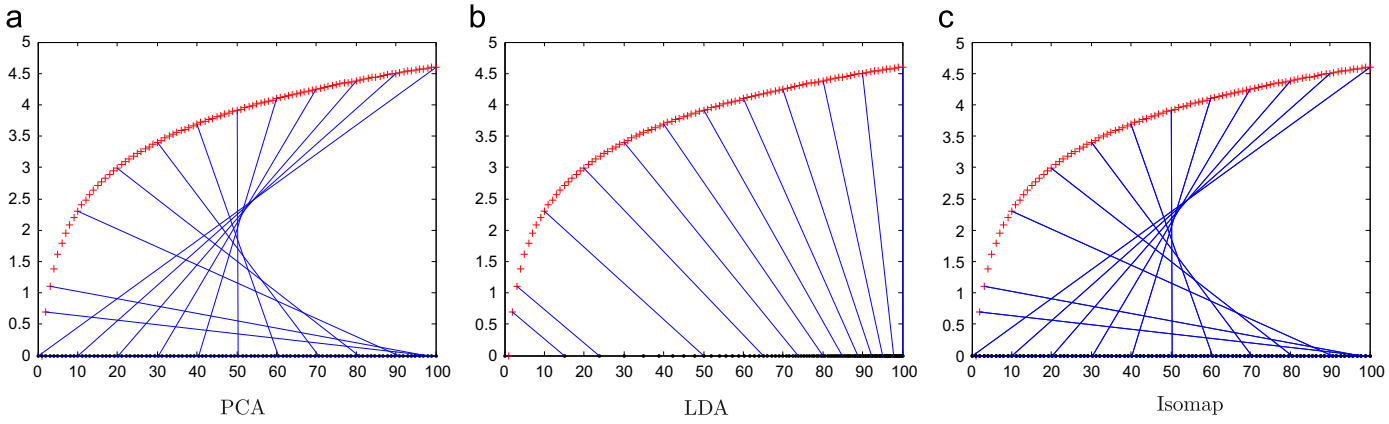


Fig. 6. Three stable results in the logarithmic curve case: (a) PCA, (b) LDA and (c) Isomap.

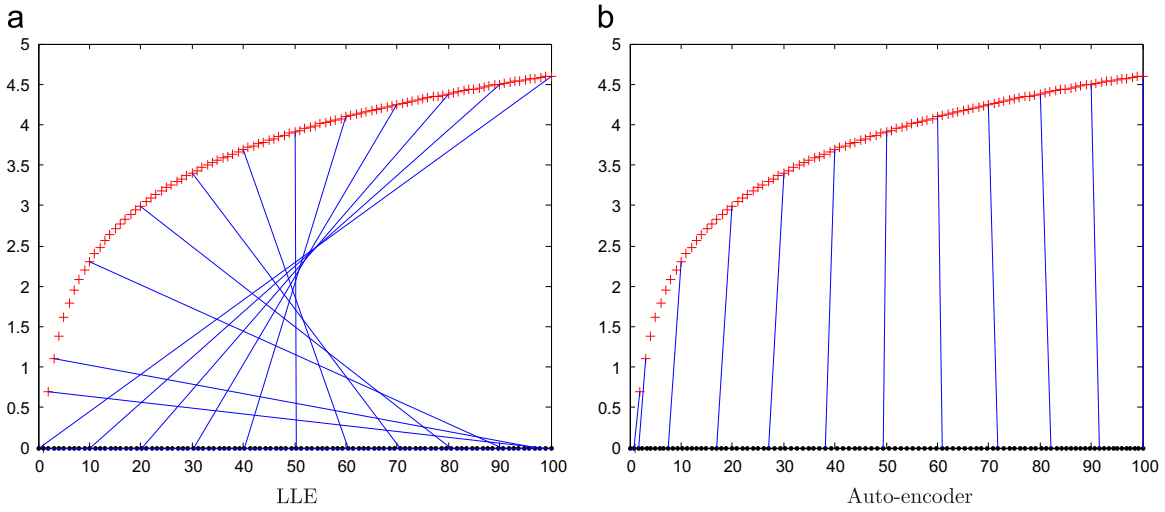


Fig. 7. Two unstable results in the logarithmic curve case: (a) LLE and (b) auto-encoder.

has the tendency of ascending first and descending in succession which means $dis(x_1, x_2)$ is nearly the same with $dis(x_{99}, x_{100})$, while $dis(x_{50}, x_{51})$ is a little larger than them.

It is hard to say which method is better or which result is more accurate. It depends on the way you think. The first 20 points increase slowly in the horizontal axis direction but quickly in the vertical axis direction, while the last 20 points are the opposite. Relatively, the middle points just fall in between. From this perspective, auto-encoder seems to do a better projection than other four methods and has a better local distance preserving nature.

The second case: In the second concentric circles case, results of PCA, LDA, LLE and Isomap are the same, as shown in Fig. 9(a). This is a good result, since four symmetric points are projected into the same point. Thus, in this case, these four methods all do a good job. However the result of auto-encoder is relatively different, as shown in Fig. 9(b). It seems like a bad and confusing result at first sight, but again go in deeper and try to change an angle to consider this problem, you will find something.

Let us move to Fig. 10. In Fig. 10(a), there are four points in the concentric circles which are mapped to the maximum and minimum values of each circle after dimensionality reduction, indicated by pink and cyan lines respectively, and the tangents of these four points are parallel. In Fig. 10(b), there are two groups of symmetrical points which happen to be mapped to the same value zero, indicated by green lines. Auto-encoder maps corresponding points within two close concentric circles into almost the same point. Every half a circle corresponds to a "minimum-to-maximum" cycle. In other words, the

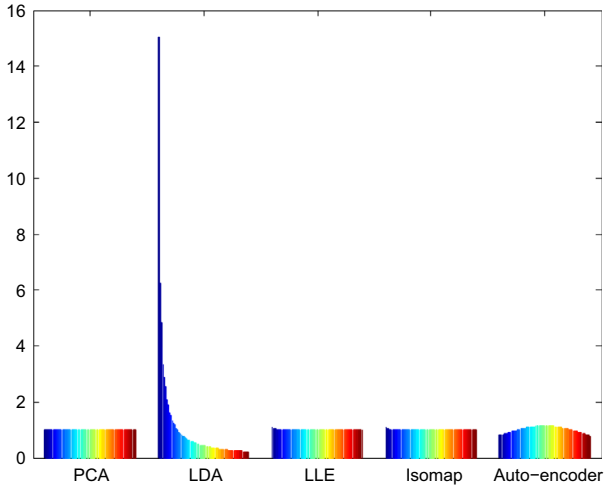


Fig. 8. Dif of five methods in the logarithmic curve case.

concentric circles shown in Fig. 10 can be seen as made up of four semicircles, or four repetitive structures and auto-encoder maps four repetitive semicircles to almost the same range.

We can do a little bit mathematical deduction here, which is simple enough to explain why a single point is projected to the maximum value or zero. Use (x_1, x_2) to denote the original point, y to denote the projected point, (w_1, w_2) to denote the trained

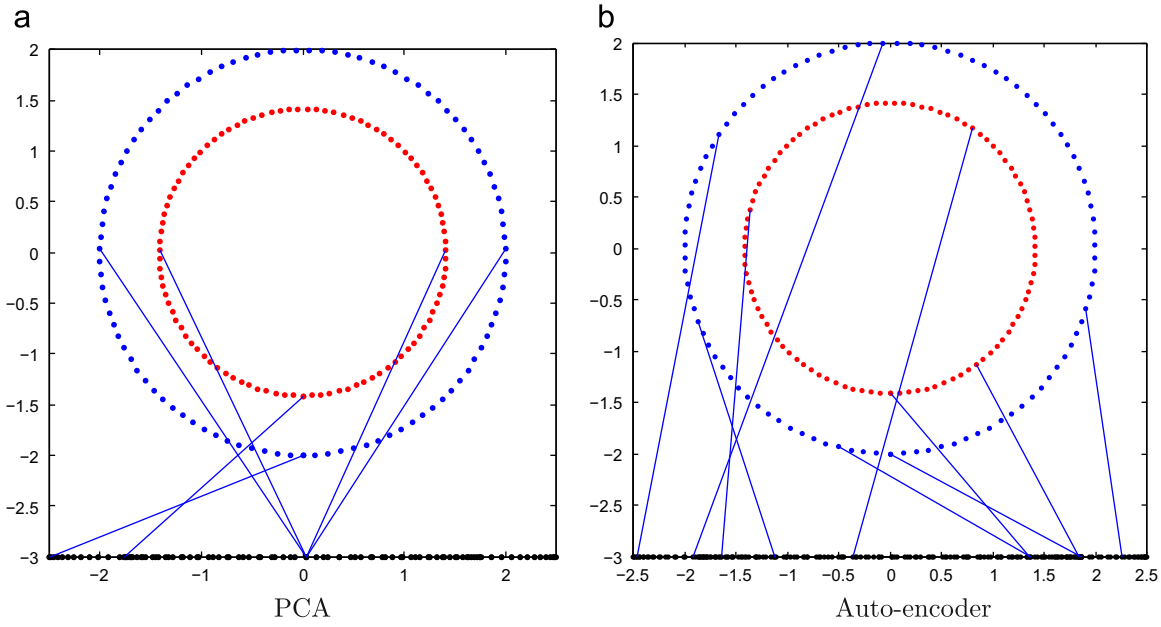


Fig. 9. Two results in the concentric circles case: (a) PCA and (b) auto-encoder.

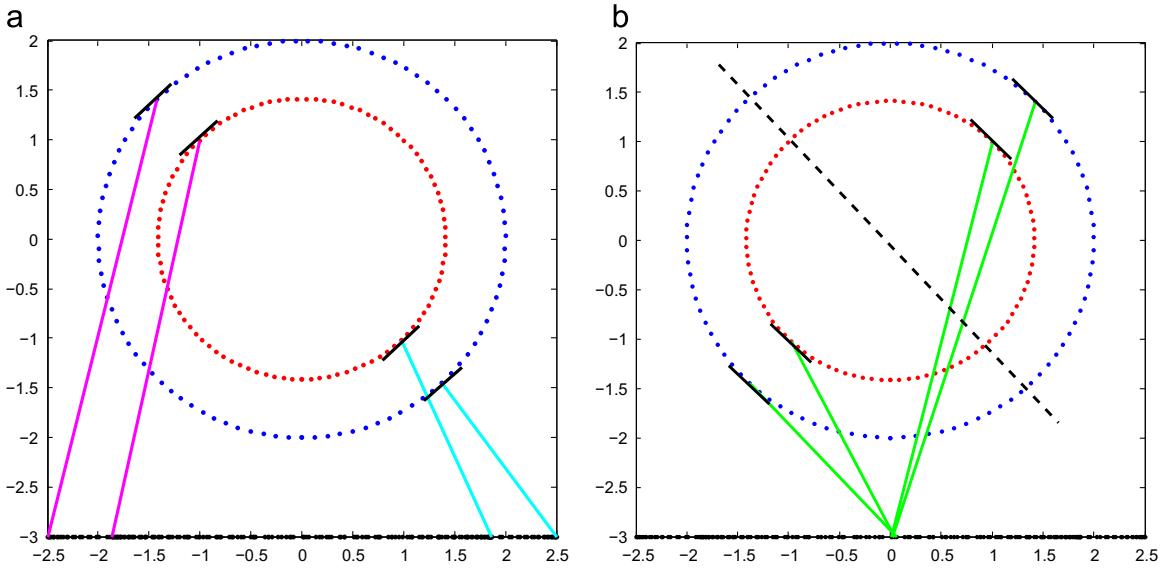


Fig. 10. Result of auto-encoder in the concentric circles case.

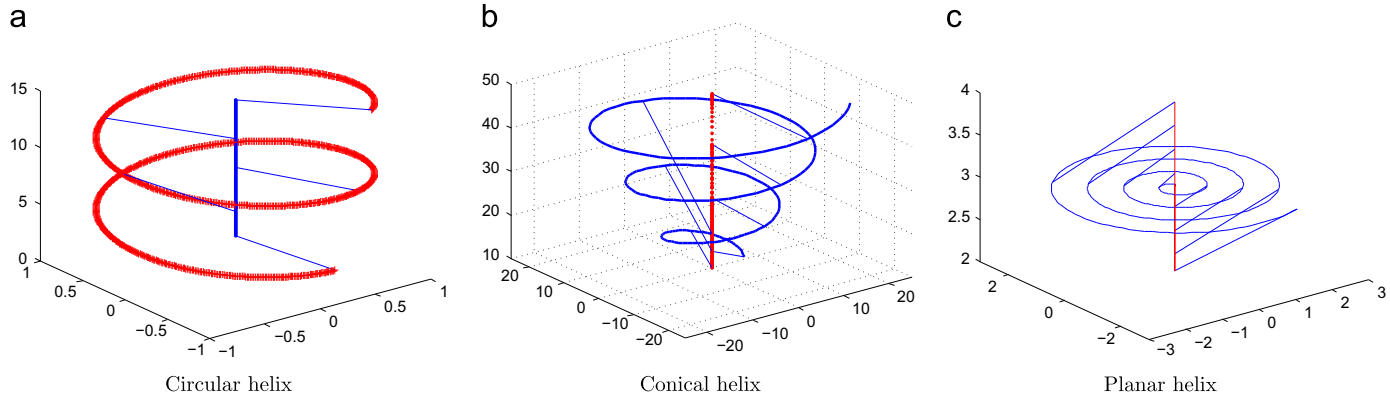


Fig. 11. Three results in three helix cases: (a) circular helix, (b) conical helix and (c) planar helix.

weights and b to denote the trained bias. Consider the encoder only, we can have the following:

$$y = sigmoid(w_1 * x_1 + w_2 * x_2 + b)$$

By analyzing the trained weights (w_1, w_2) and original points (x_{max1}, x_{max2}), (x_{min1}, x_{min2}) projected to the maximum and the minimum value, we find two equations: (1) $w_1 = -w_2$; (2) $x_{min1} = x_{max2}$, $x_{min2} = x_{max1}$.

Plug these equations into the formula above, we can have:

$$\begin{aligned} y_{max} &= sigmoid(w_1 * x_{max1} + w_2 * x_{max2} + b) \\ &= sigmoid(w_1 * (x_{max1} - x_{max2}) + b) \\ y_{min} &= sigmoid(w_1 * x_{min1} + w_2 * x_{min2} + b) \\ &= sigmoid(w_1 * (x_{max2} - x_{max1}) + b) \end{aligned}$$

When $(x_{max1} - x_{max2})$ achieves the maximum value, $(x_{max2} - x_{max1})$ achieves the minimum value at the same time. To reach this, x_{max1} should be the maximum and x_{max2} should be the minimum among the coordinates of the original points, which is true.

3.2.2. On synthesized data from $\mathbb{R}^3$ to $\mathbb{R}^1$

In this subsection, we implement PCA and auto-encoder within three transformations of helix cases in three-dimensional spaces.

In these three cases, auto-encoder is always doing a good job. Results are shown in Fig. 11. A similar analysis of *dif* of the circular helix can be found in Fig. 12. From the results we can see, in some cases, auto-encoder not only reduces dimensionality, but can also detect repetitive structures. From the viewpoint of information theory, compressing the data dimensionality is to find redundancy while repetitive structure is a kind of redundancy. We believe that this is a good property for many applications.

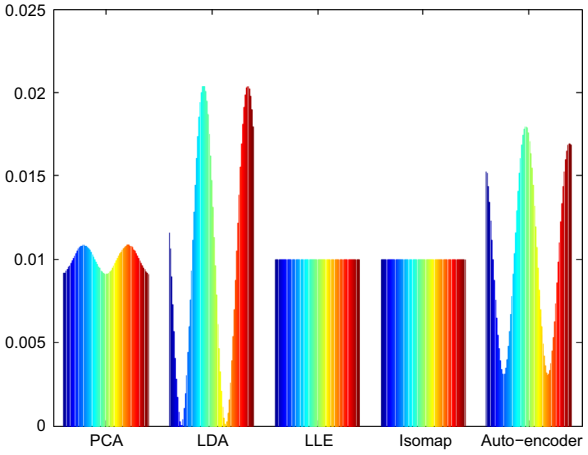


Fig. 12. Dif of five methods in the circular helix case.

3.2.3. On real datasets from $\mathbb{R}^{784}$ to $\mathbb{R}^s$

In this subsection, we implement PCA and auto-encoder within real datasets—MNIST and Olivetti face dataset to see how auto-encoder differs from PCA and give visualization results.

Visualize the function learned by auto-encoder: In this case, we want to see what does auto-encoder exactly learn after training. For MNIST dataset, we set s equals to 2, 10 and 300 to visualize the function learned by auto-encoder by weights between the first layer and the second layer. For Olivetti face dataset, we resize the image from 64*64 to 28*28 and then set s equals to 10, 40 and 300 for visualization. Results are shown in Figs. 13 and 14. Obviously, the features auto-encoder learned, e.g. strokes and faces, are important for computer vision tasks.

Visualize the comparison results between auto-encoder and PCA: In this case, we set s equals to 2 and 3 and use auto-encoder and PCA to reduce the dimensionality respectively in MNIST dataset. Then we visualize the comparison results between auto-encoder and PCA. Results are shown in Figs. 15–17.

Fig. 15 shows when $s=2$, every training example is projected to a point in two-dimensional space using auto-encoder and PCA respectively. All training examples of 10 classes are denoted in different colors. Same goes for Fig. 16, except that $s=3$.

Fig. 17 shows training examples in every single class. Fig. 17(a) displays the visualization of projected points of digit 1 using auto-encoder, while Fig. 17(g) displays the visualization of digit 1 using PCA. From the results, we can see that in this case auto-encoder performs better than PCA. Compared to PCA, auto-encoder tends to project images of the same class to edges and corners, which can lead to preferable results than PCA.

3.3. Influence of the number of hidden layer nodes

3.3.1. MNIST dataset

Using auto-encoder to reduce the dimensionality from 784 to s, we record the change of softmax classifier performance when the dimensionality s varies.

For every dimensionality, we run the program for five times and get the average result. Results are shown in Fig. 18. Blue "*" represents average result of auto-encoder before finetuning and blue "." represents average result of auto-encoder after finetuning.

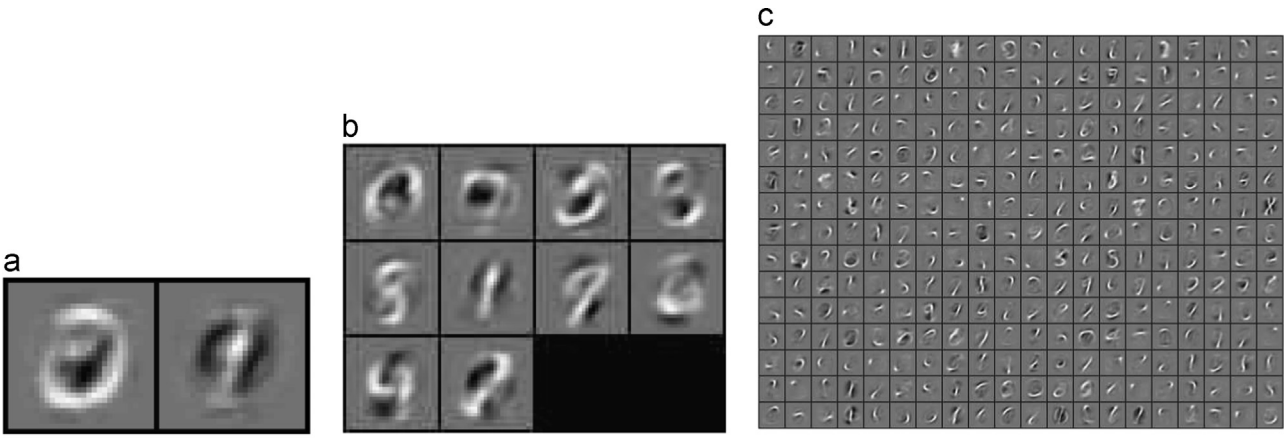


Fig. 13. Visualization of MNIST dataset: (a) $s = 2$, (b) $s = 10$ and (c) $s = 300$.

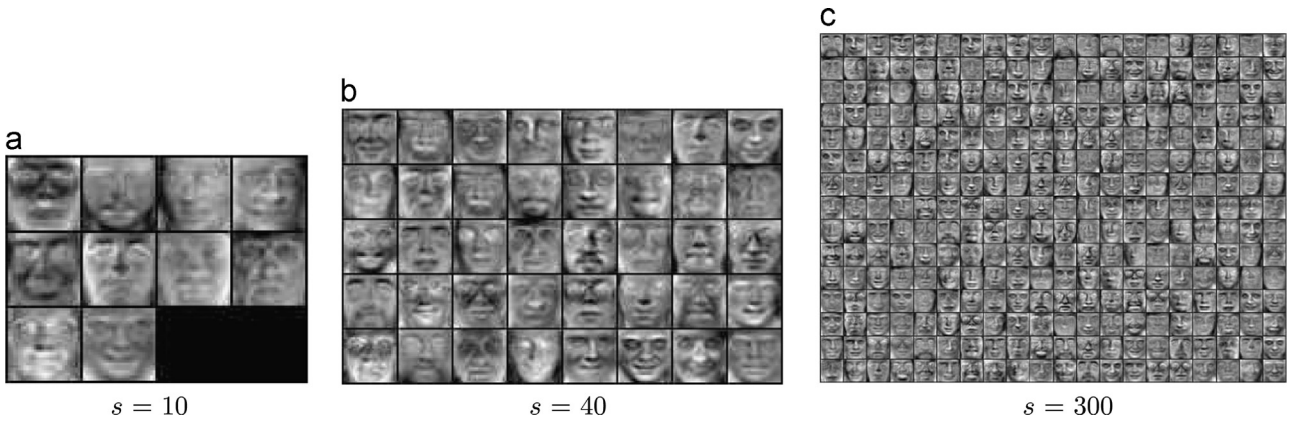


Fig. 14. Visualization of Olivetti face dataset (a) $s=10$, (b) $s=40$ and (c) $s=300$.

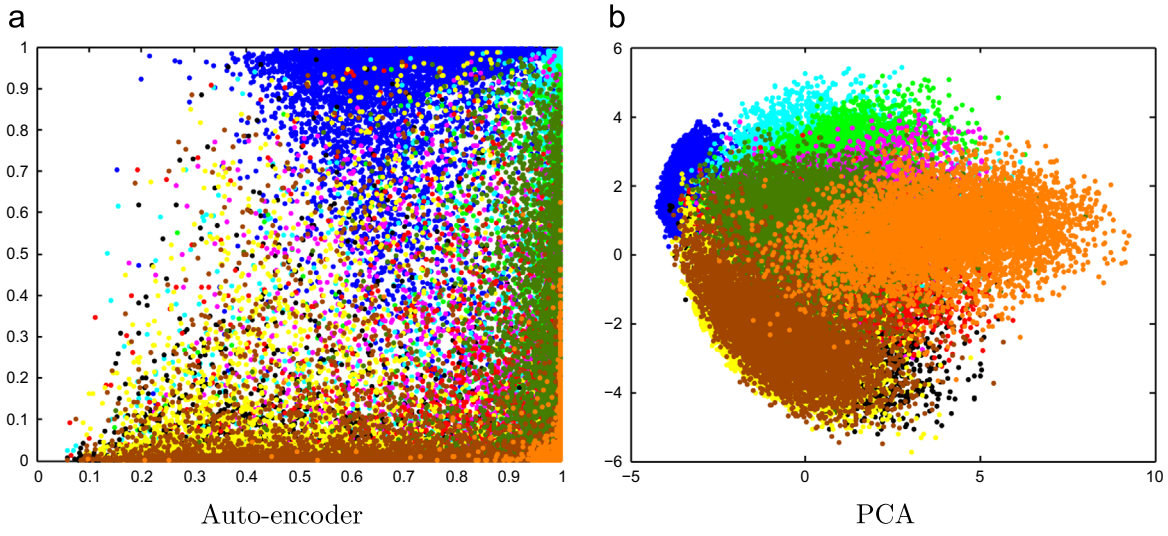


Fig. 15. Dimensionality reduction: $\mathbb{R}^{784} \rightarrow \mathbb{R}^2$: (a) auto-encoder and (b) PCA.

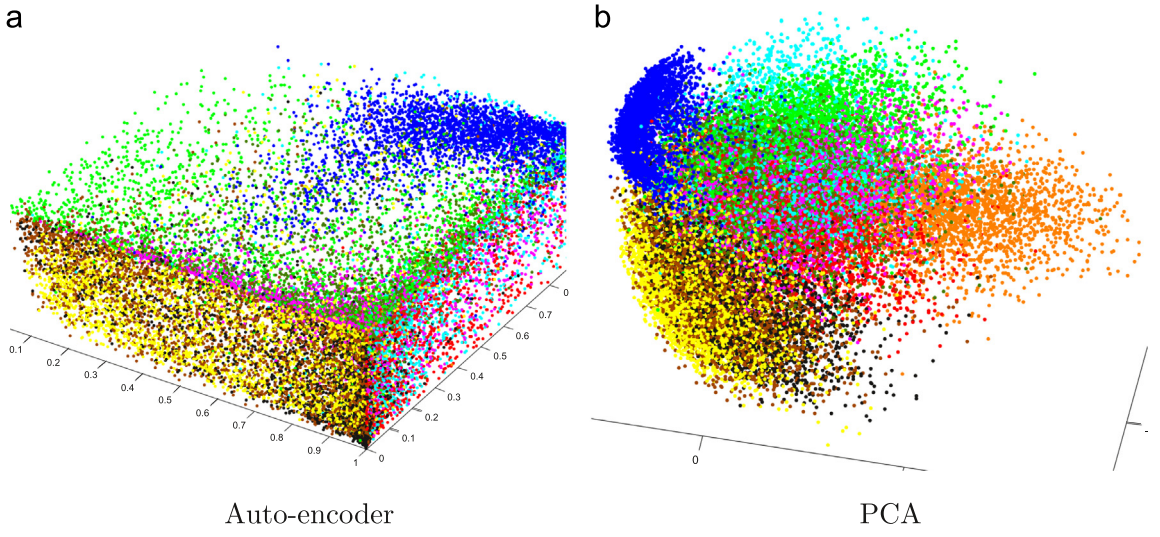


Fig. 16. Dimensionality reduction: $\mathbb{R}^{784} \rightarrow \mathbb{R}^3$: (a) auto-encoder and (b) PCA.

From Fig. 18, the accuracy rises at first goes down later and then starts to level off, rising slowly at the same time. There is an observation that when the dimensionality s is near 10, just equal to the intrinsic dimensionality of original data, accuracy is relatively high. This might reveals a possible relation between the number of hidden layer nodes and the intrinsic dimensionality of original data and gives us some advice on choosing the number of hidden layer nodes.

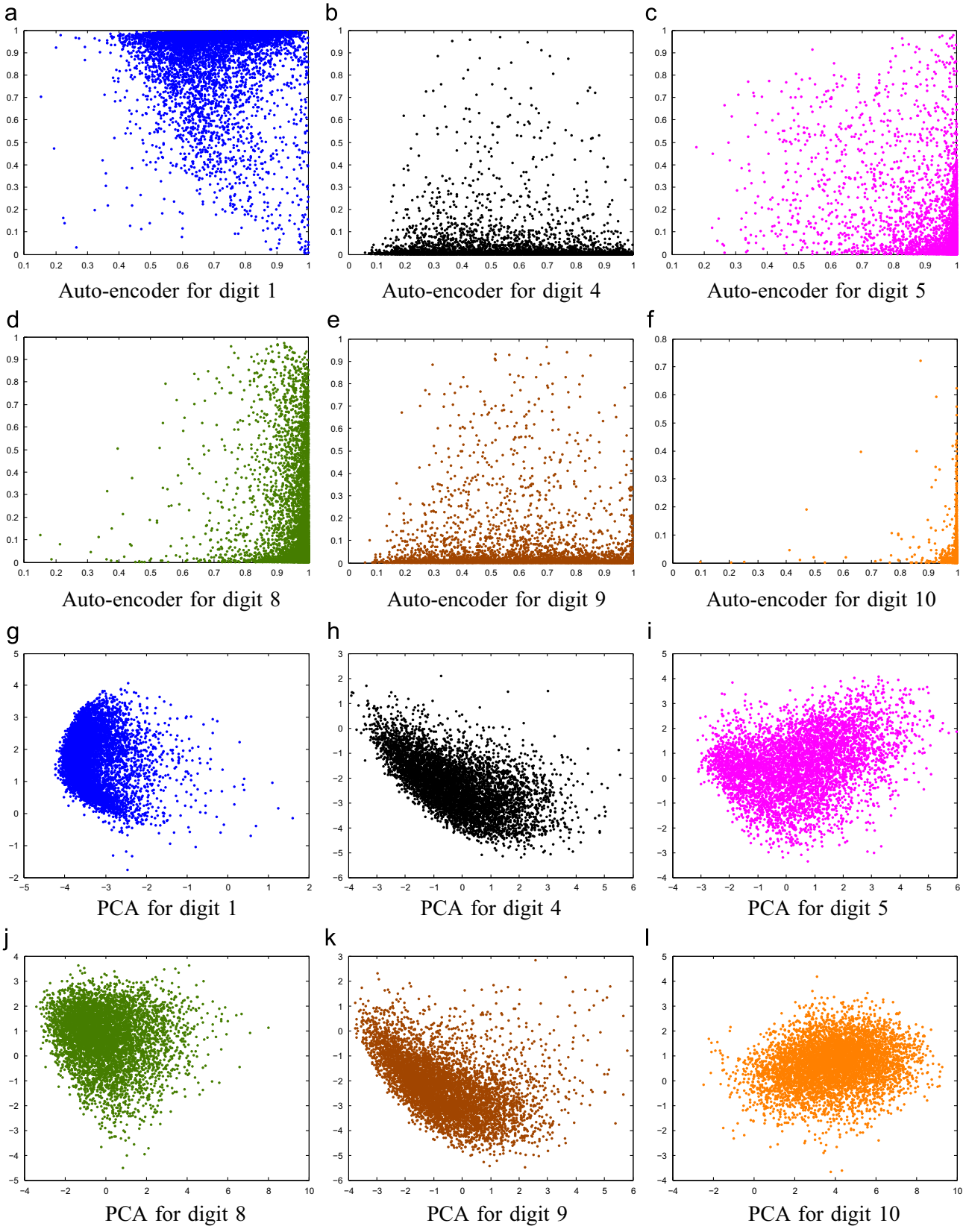


Fig. 17. Dimensionality reduction: $\mathbb{R}^{784} \rightarrow \mathbb{R}^2$: (a) auto-encoder for digit 1, (b) auto-encoder for digit 4, (c) auto-encoder for digit 5, (d) auto-encoder for digit 8, (e) auto-encoder for digit 9, (f) auto-encoder for digit 10, (g) PCA for digit 1, (h) PCA for digit 4, (i) PCA for digit 5, (j) PCA for digit 8, (k) PCA for digit 9 and (l) PCA for digit 10.

In this case, we run the program for 15 times when the dimensionality of the new representation is between 5 and 15 (near 10) to get more stable results. Red "*" represents maximum result of auto-encoder before finetuning and red "." represents maximum result of auto-encoder after finetuning. Results are shown in Fig. 19. We can see that when the dimensionality s is near 10, corresponding accuracy is basically over 50%, which is higher than neighbors.

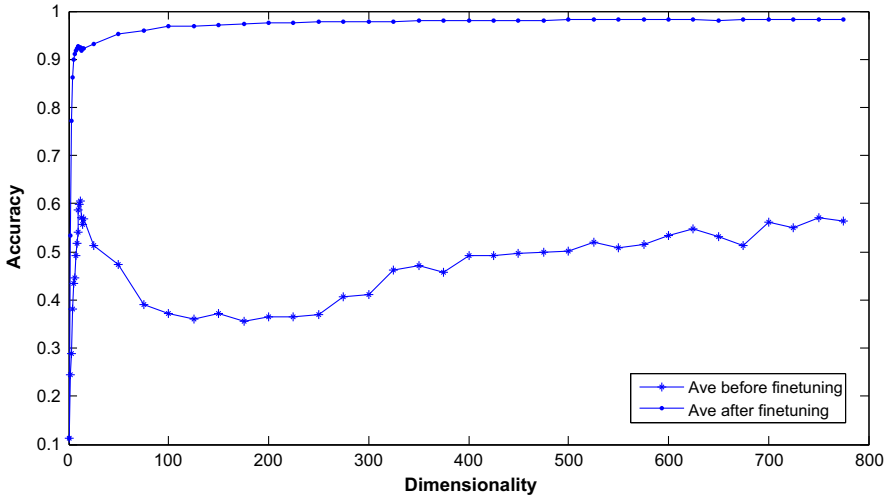


Fig. 18. Softmax classifier accuracy with the dimensionality from 1 to 775 in MNIST dataset.

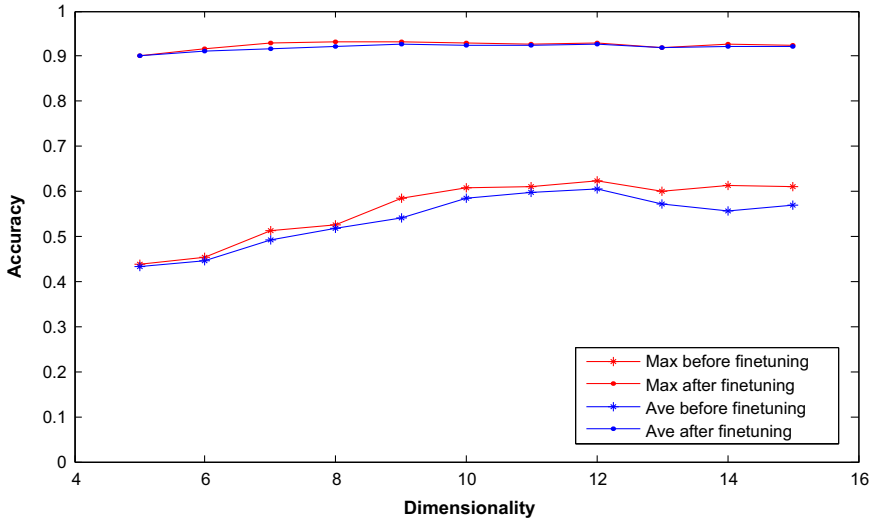


Fig. 19. Softmax classifier accuracy with the dimensionality from 5 to 15 in MNIST dataset. (For interpretation of the references to color in this figure, the reader is referred to the web version of this paper.)

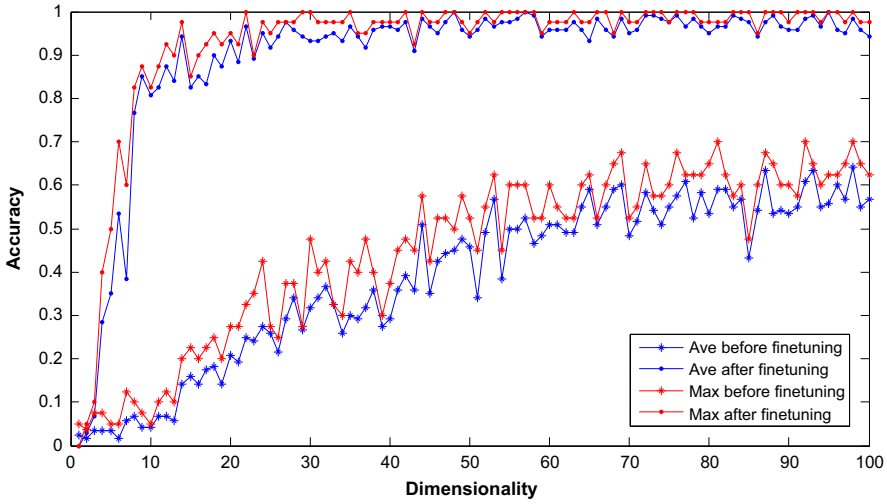


Fig. 20. Softmax classifier accuracy with the dimensionality from 1 to 100 in Olivetti face dataset. (For interpretation of the references to color in this figure, the reader is referred to the web version of this paper.)

3.3.2. Olivetti face dataset

First resize the image from 64*64 to 28*28, and the following steps are the same as above. Results are shown in Fig. 20.

However, we do not observe the same phenomenon as in MNIST dataset. A possible reason maybe MNIST dataset is simple, and a small number of hidden layer nodes are enough to model the data. When we set the number of hidden layer nodes equal to the intrinsic dimensionality, hidden layer nodes may fit the structure of original data better. So that we get almost the best result when s equals to 10 in MNIST dataset.

While the Olivetti face dataset is much more complicated than MNIST, and a small number of hidden layer nodes are not able to model the data. Thats maybe why in this case the accuracy is just rising along the dimensionality s.

4. Conclusion

This paper starts from auto-encoder and focuses on its ability to reduce the dimensionality. Firstly, experiments show that the results of auto-encoder indeed differ from other dimensionality reduction methods. In some cases, auto-encoder not only reduces dimensionality, but can also detect repetitive structures. We believe that this is a good property for many applications. Maybe in situations with repetitive structures, auto-encoder is more suitable. Secondly, the paper preliminarily investigates the influence of the number of hidden layer nodes on performance of auto-encoder. For classification on MNIST dataset without finetuning, the best results are achieved when the number of hidden layer nodes is set around the intrinsic dimensionality of the dataset. This might reveal the possible relation between the number of hidden layer nodes and the intrinsic dimensionality of the dataset.

Acknowledgments

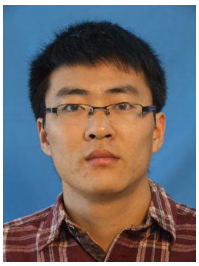
This work was supported by the National Natural Science Foundation of China (No. 61472103) and Key Program (No. 61133003).

References

[1] M.A. Sartori, P.J. Antsaklis, A simple method to derive bounds on the size and to train multilayer neural networks, IEEE Trans. Neural Netw. 2 (4) (1991) 467–471.
[2] M. Arbib, Review of 'Perceptrons: an introduction to computational geometry', IEEE Trans. Inf. Theory 15 (6) (1969) 738–739.
[3] G.B. Huang, Learning capability and storage capacity of two-hidden-layer feedforward networks, IEEE Trans. Neural Netw. 14 (2) (2003) 274–281.
[4] Yoshua Bengio, Learning deep architectures for AI, Found. Trends® Mach. Learn. 2 (1) (2009) 1–127.
[5] Yoshua Bengio, Deep learning of representations: looking forward, in: Statistical Language and Speech Processing, 2013, pp. 1–37.
[6] Yoshua Bengio, Aaron Courville, Pascal Vincent, Unsupervised feature learning and deep learning: a review and new perspectives, in: CoRR, 2012.
[7] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biol. Cybern. 59 (4–5) (1988) 291–294.
[8] S.C. Huang, Y.F. Huang, Bounds on the number of hidden neurons in multilayer perceptrons, IEEE Trans. Neural Netw. 2 (1) (1991) 47–55.
[9] C. Xiang, S.Q. Ding, T.H. Lee, Geometrical interpretation and architecture selection of MLP, IEEE Trans. Neural Netw. 16 (1) (2005) 84–96.
[10] Corinna Cortes, Vladimir Vapnik, Support-vector networks, Mach. Learn. (1995) 273–297.
[11] David Demers, Garrison Cottrell, Non-linear dimensionality reduction, Adv. Neural Inf. Process. Syst. (1993) 580–587.
[12] Ali Ghodsi, Dimensionality Reduction a Short Tutorial, Department of Statistics and Actuarial Science, 2006.
[13] Geoffrey Hinton, A practical guide to training restricted Boltzmann machines, Neural Netw.: Tricks Trade (2012) 599–619.
[14] Geoffrey Hinton, Simon Osindero, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006) (2006).
[15] Geoffrey Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[16] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biol. Cybern. 52 (3) (1985) 141–152.
[17] J. Wang, H.B. He, D.V. Prokhorov, A folded neural network autoencoder for dimensionality reduction, Proc. Comput. Sci. 13 (2012) 120–127.
[18] Mark A. Kramer, Nonlinear principal component analysis using auto-associative neural networks, AIChE J. 37 (2) (1991) 233–243.
[19] Alex Krizhevsky, Sutskever Ilya, Geoffrey Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. (2012) 1097–1105.
[20] Marvin Minsky, Seymour Papert, Perceptrons: An Introduction to Computational Geometry (Expanded edition), 1988.
[21] Japkowicz Nathalie, S. Hanson, M. Gluck, Nonlinear autoassociation is not equivalent to PCA, Neural Comput. 12 (3) (2000) 531–545.
[22] S. Tamura, Capabilities of a three layer feedforward neural network, in: IEEE International Joint Conference on Neural Networks, 1991, pp. 2757–2762.
[23] S. Tamura, M. Tateishi, Capabilities of a four-layered feedforward neural network: four layers versus three, IEEE Trans. Neural Netw. 8 (2) (1997) 251–255.
[24] Q.V. Le, Building high-level features using large scale unsupervised learning, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 8595–8598.
[25] Wang Wei, Huang Yan, Wang Yizhou, Wang Liang, Generalized autoencoder: a neural network framework for dimensionality reduction, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 496–503.
[26] Matthew D. Zeiler, Rob Fergus, Visualizing and understanding convolutional networks, in: CoRR, 2013.
[27] Hu Zhaohua, Song Yaoliang, Dimensionality reduction and reconstruction of data based on autoencoder network, J. Electron. Inf. Technol. 31 (5) (2009) 1189–1192.
[28] Zhenyao Zhu, Ping Luo, Xiaogang Wang, Xiaoou Tang, Deep learning identity-preserving face space, in: IEEE International Conference on Computer Vision, 2013, pp. 113–120.

Yasi Wang is currently a Ph.D. candidate at Harbin Institute of Technology, Harbin, China. Her research interests include image and video understanding, machine learning and deep learning.

Hongxun Yao received the B.S. and M.S. degrees in Computer Science from the Harbin Shipbuilding Engineering Institute, Harbin, China, in 1987 and in 1990, respectively, and received Ph.D. degree in Computer Science from Harbin Institute of Technology in 2003. Currently, she is a Professor with School of Computer Science and Technology, Harbin Institute of Technology. Her research interests include computer vision, multimedia computing, and human-computer interaction. She has published five books and over 200 scientific papers.



Sicheng Zhao is currently a Ph.D. candidate at Harbin Institute of Technology, Harbin, China. His research interests include affective computing, social media analysis and multimedia information retrieval.