Auto-encoder Based Data Clustering

Chunfeng Song[1], Feng Liu[2], Yongzhen Huang[1], Liang Wang[1], and Tieniu Tan[1]

[1] National Laboratory of Pattern Recognition (NLPR),
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
[2] School of Automation, Southeast University, Nanjing, 210096, China

Abstract. Linear or non-linear data transformations are widely used processing techniques in clustering. Usually, they are beneficial to enhancing data representation. However, if data have a complex structure, these techniques would be unsatisfying for clustering. In this paper, based on the auto-encoder network, which can learn a highly non-linear mapping function, we propose a new clustering method. Via simultaneously considering data reconstruction and compactness, our method can obtain stable and effective clustering. Experiments on three databases show that the proposed clustering model achieves excellent performance in terms of both accuracy and normalized mutual information.

Keywords: Clustering, Auto-encoder, Non-linear transformation.

1 Introduction

Data clustering [4] is a basic problem in pattern recognition, whose goal is grouping similar data into the same cluster. It attracts much attention and various clustering methods have been presented, most of which either deal with the original data, e.g., K-means [10], its linear transformation, e.g., spectral clustering [7], or its simple non-linear transformation, e.g., kernel K-means [2]. However, if original data are not well distributed due to large intra-variance as shown in the left part of Figure 1, it would be difficult for traditional clustering algorithms to achieve satisfying performance.

To address the above problem, we attempt to map original data space to a new space which is more suitable for clustering. The auto-encoder network [1] is a good candidate to handle this problem. It provides a non-linear mapping function by iteratively learning the encoder and the decoder. The encoder is actually the non-linear mapping function, and the decoder demands accurate data reconstruction from the representation generated by the encoder. This process is iterative, which guarantees that the mapping function is stable and effective to represent the original data. Different from kernel K-means [2], which also introduces non-linear transformations with fixed kernel functions, the non-linear function in auto-encoder is learned by optimizing an objective function.

The auto-encoder network is originally designed for data representation, and it aims to minimize the reconstruction error. However, to the best of our knowledge, though widely used, the auto-encoder network has not been utilized for

J. Ruiz-Shulcloper and G. Sanniti di Baja (Eds.): CIARP 2013, Part I, LNCS 8258, pp. 117–124, 2013.

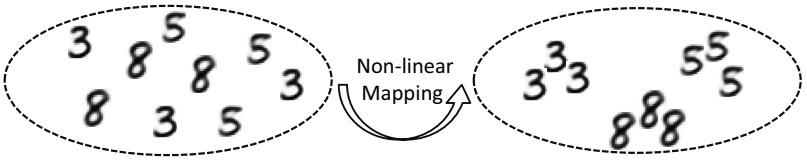


Fig. 1. Left: Original distribution of data. Due to large intra-variance, it is difficult to classify them correctly. Right: By applying a non-linear transformation, the data become compact with respect to their corresponding cluster centers in the new space.

clustering tasks. To make it suitable for clustering, we propose a new objective function embedded into the auto-encoder model. It contains two parts: the reconstruction error and the distance between data and their corresponding cluster centers in the new space. During optimization, data representation and clustering centers are updated iteratively, from which a stable performance of clustering is achieved and the new representation is more compact with respect to the cluster centers. The right part of Figure 1 illustrates such an example. To evaluate the effectiveness of this model, we conduct a series of experiments in three widely used databases for clustering. The experimental results show that our method performs much better than traditional clustering algorithms.

The rest of the paper is organized as follows: firstly we propose our method in Section 2, then experimental settings and results are provided in Section 3. Finally, Section 4 concludes the paper and discusses future work.

2 Proposed Model

In this section, we explain the proposed clustering model in details. As shown in Figure 2, the data layer (e.g., the pixel representation) of an image is firstly mapped to the code layer, which is then used to reconstruct the data layer. The objective is minimizing the reconstruction error as well as the distance between data points and corresponding clusters in the code layer. This process is implemented via a four-layer auto-encoder network, in which a non-linear mapping is resolved to enhance data representation in the data layer. For clarity, in the next subsections, we firstly introduce the auto-encoder network, and then explain how to use it for clustering.

2.1 Auto-encoders

Without loss of generality, we take an one-layer auto-encoder network as an example. It consists of an encoder and a decoder. The encoder maps an input x_i to its hidden representation h_i. The mapping function is usually non-linear and the following is a common form:

$$h_i = f(x_i) = \frac{1}{1 + \exp(-(W_1 x_i + b_1))}, \tag{1}$$

where W_1 is the encoding weight, b_1 is the corresponding bias vector.

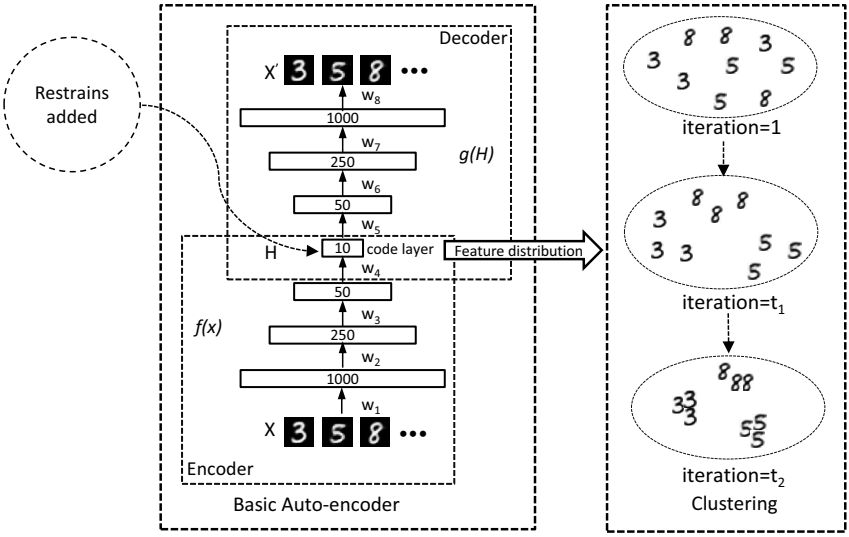


Fig. 2. Framework of the proposed method

The decoder seeks to reconstruct the input x_i from its hidden representation h_i. The transformation function has a similar formulation:

$$x_i' = g(h_i) = \frac{1}{1 + \exp(-(W_2 h_i + b_2))}, \tag{2}$$

where W_2, b_2 are the decoding weight and the decoding bias vector respectively. The auto-encoder model aims to learn a useful hidden representation by minimizing the reconstruction error. Thus, given N training samples, the parameters W_1, W_2, b_1 and b_2 can be resolved by the following optimization problem:

$$\min \frac{1}{N} \sum_{i=1}^{N} \|x_i - x_i'\|^2. \tag{3}$$

Generally, an auto-encoder network is constructed by stacking multiple one-layer auto-encoders. That is, the hidden representation of the previous one-layer auto-encoder is fed as the input of the next one. For more details of the auto-encoder network and its optimization, readers are referred to [1].

2.2 Clustering Based on Auto-encoder

Auto-encoder is a powerful model to train a mapping function, which ensures the minimum reconstruction error from the code layer to the data layer. Usually, the code layer has less dimensionality than the data layer. Therefore, auto-encoder can learn an effective representation in a low dimensional space, and it can be considered as a non-linear mapping model, performing much better than PCA [3]. However, auto-encoder contributes little to clustering because it does not pursue that similar input data obtain the same representations in the code

layer, which is the nature of clustering. To solve this problem, we propose a new objective function and embed it into the auto-encoder model:

$$\min_{W,b} \frac{1}{N} \sum_{i=1}^{N} \|x_i - x_i'\|^2 - \lambda \cdot \sum_{i=1}^{N} \|f^t(x_i) - c_i^*\|^2 \tag{4}$$

$$c_i^* = \arg\min_{c_j^{t-1}} \|f^t(x_i) - c_j^{t-1}\|^2, \tag{5}$$

where N is the number of samples in the dataset; $f^t(\cdot)$ is the non-linear mapping function at the t^{th} iteration; c_j^{t-1} is the j^{th} cluster center computed at the $(t-1)^{th}$ iteration[1]; and c_i^* is the closest cluster center of the i^{th} sample in the code layer. This objective ensures that the data representations in the code layer are close to their corresponding cluster centers, and meanwhile the reconstruction error is still under control, which is important to obtain stable non-linear mapping.

Two components need to be optimized: the mapping function $f(\cdot)$ and the cluster centers c. To solve this problem, an alternate optimization method is proposed, which firstly optimizes $f(\cdot)$ while keeps c fixed, and then updates the cluster center:

$$c_j^t = \frac{\sum_{x_i \in C_j^{t-1}} f^t(x_i)}{|C_j^{t-1}|}, \tag{6}$$

where C_j^{t-1} is the set of samples belonging to the j^{th} cluster at the $(t-1)^{th}$ iteration and $|C_j|$ is the number of samples in this cluster. The sample assignment computed in the last iteration is used to update the cluster centers of the current iteration. Note that sample assignment at the first iteration C^0 is initialized randomly. For clarity, we conclude our method in Algorithm 1.

Algorithm 1. Auto-encoder based data clustering algorithm

1: **Input:** Dataset X, the number of clusters K, hyper-parameter λ, the maximum number of iterations T.
2: **Initialize** sample assignment C^0 randomly.
3: **Set** t to 1.
4: **repeat**
5: Update the mapping network by minimizing Eqn. (4) with stochastic gradient descent for one epoch.
6: Update cluster center c^t via Eqn. (6).
7: Partition X into K clusters and update the sample assignment C^t via Eqn. (5).
8: $t = t + 1$.
9: **until** $t > T$
10: **Output:** Final sample assignment C.

[1] We use stochastic gradient descent (SGD) [5] to optimize the parameters of auto-encoder.

3 Experiments

3.1 Experimental Setups

Database. All algorithms are tested on 3 databases: MNIST[2], USPS[3] and YaleB[4]. They are widely used for evaluating clustering algorithms.

1. **MNIST** contains 60,000 handwritten digits images (0∼9) with the resolution of 28×28.
2. **USPS** consists of 4,649 handwritten digits images (0∼9) with the resolution of 16×16.
3. **YaleB** is composed of 5,850 faces image over ten categories, and each image has 1200 pixels.

Parameters. Our clustering model is based on a four-layers auto-encoder network with the structure of 1000-250-50-10. The parameter λ in Eqn. (4) is set by cross validation. That is 0.1 on MNIST, 0.6 on USPS and YaleB. The weights W in the auto-encoder network are initialized via a standard restricted Boltzmann machine (RBM) pre-training [3].

Baseline Algorithms. To demonstrate the effectiveness of our method, we compare our method with three classic and widely used clustering algorithms: K-means [10], spectral clustering [7] and N-cut [9].

Evaluation Criterion. Two metrics are used to evaluate experimental results explained as follows.

1. **Accuracy (ACC)** [11]. Given an image x_i, let c_i be the resolved cluster label and r_i be the ground truth label. ACC is defined as $\sum_{i=1}^{N} \delta(r_i, map(c_i))/N$, where N is the number of instances in the dataset and $\delta(x, y)$ is the delta function that equals one if $x = y$ and zero otherwise. $Map(c_i)$ is the function that maps each cluster label c_i to the equivalent label from the datasets. The best mapping can be found by using the Kuhn-Munkres algorithm [8].
2. **Normalized mutual information (NMI)** [6]. Let R denote the label obtained from the ground truth and C be the label obtained by clustering. The NMI is defined as $\mathrm{MI}(R,C)/\max(\mathrm{H}(R), \mathrm{H}(C))$, where $\mathrm{H}(X)$ is the entropies of X, and $\mathrm{MI}(X,Y)$ is the mutual information of X and Y.

3.2 Quantitative Results

In this subsection, we firstly evaluate the influence of the iteration number in our algorithm. Figure 3 shows the change of NMI and ACC as the iteration number increases on three databases.

It can be found that the performance is enhanced fast in the first ten iterations, which demonstrates that our method is effective and efficient. After dozens of

[2] `http://yann.lecun.com/exdb/mnist/`
[3] `http://www.gaussianprocess.org/gpml/data/`
[4] `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`

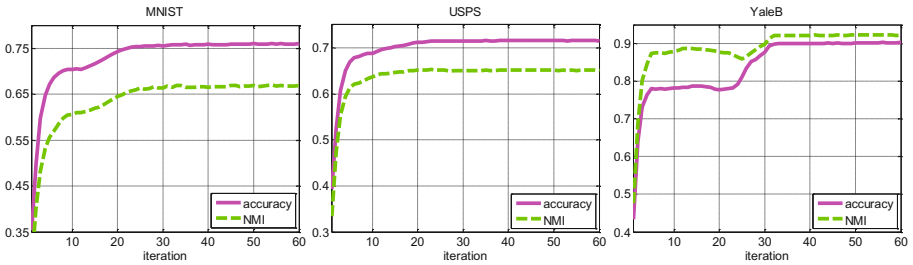


Fig. 3. Influence of the iteration number on three databases

Table 1. Performance comparison of clustering algorithms on three databases

Datasets	MNIST		USPS		YaleB	
Criterion	NMI	ACC	NMI	ACC	NMI	ACC
K-means	0.494	0.535	0.615	0.674	0.866	0.793
Spectral	0.482	0.556	**0.662**	0.693	0.881	0.851
N-cut	0.507	0.543	0.657	0.696	0.883	0.821
Proposed	**0.669**	**0.760**	0.651	**0.715**	**0.923**	**0.902**

iteration, e.g., 40∼60, both NMI and ACC become very stable. Thus, in the rest of experiments, we report the results after 50 iterations. The performances of the different methods on three datasets are shown in Table 1. Apparently that our method is better than or at least comparable to their best cases.

3.3 Visualization

In this subsection, the visualized results on MNIST are shown to provide an in-depth analysis. We draw in Figure 4 the distribution of ten categories of digits obtained by our method. Most of histograms in Figure 4 are single-peak distributions, demonstrating the compactness of data representation. Admittedly, the cases of digits 4 and 9 are not so good. We will discuss possible solutions to this problem in Section 4. The small digital images in subfigures are the reconstructed results of cluster centers in the code layer.

For comparison, we also show the average data representations over all clusters by K-means in Figure 5. The result is much worse, and can be easily understood with the motivation of our method. Generally, K-means uses a similar iteration procedure as ours in Algorithm 1 except that it is performed in the original pixel space. That is, the iteration of K-means is performed in the data layer, whereas ours in the code layer, which is mapped from the data layer with a highly non-linear function, learned by exploiting the hidden structure of data with the auto-encoder network.

3.4 Difference of Spaces

In this subsection, we analyze the difference of three spaces, i.e., the original data space, the space learned via non-linear mapping with original auto-encoder, and

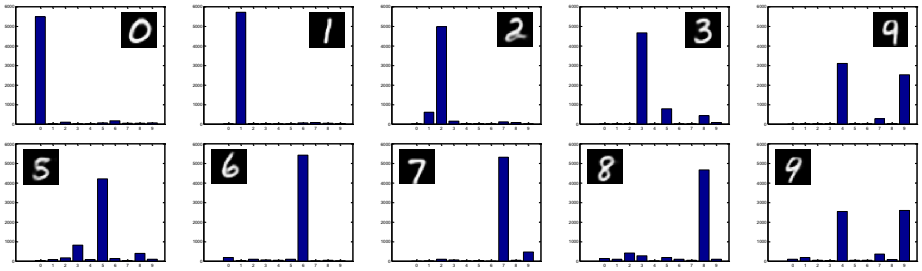


Fig. 4. Distribution of data over ten clusters and the visualized images of cluster centers after reconstruction with the learned decoder

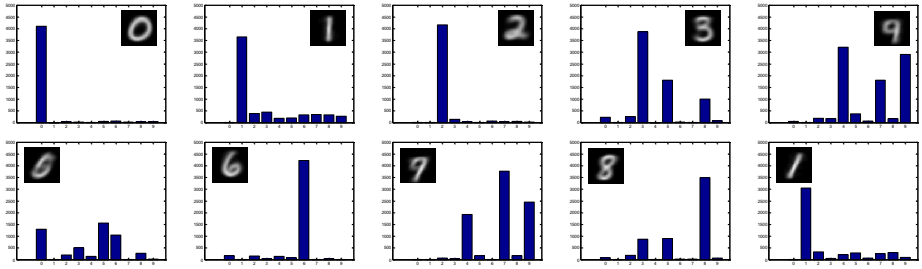
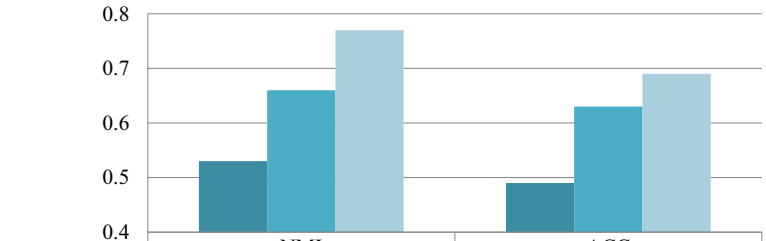


Fig. 5. Distribution of digits over 10 classes and the visualized images of 10 cluster centers generated by K-means



	NMI	ACC
Original	0.53	0.49
Auto-encoder	0.66	0.63
Proposed	**0.77**	**0.69**

Fig. 6. Performance comparison in three different spaces

the one learned by our method. Correspondingly, we apply K-means clustering in these spaces. Their clustering results are shown in Figure 6. Obviously, the clustering performance in the space of auto-encoder is much better than the one in the original space, and much worse than the one proposed by us. This result justifies two viewpoints: 1) Non-linear mapping by auto-encoder can greatly improve the representation of data for clustering; 2) Our proposed objective function, defined in Eqn. (4)∼(6), is effective to further enhance clustering due to the design of increasing data compactness as analyzed in Section 2.2.

4 Conclusions

In this paper, we have proposed a new clustering method based on the auto-encoder network. By well designing the constraint of the distance between data and cluster centers, we obtain a stable and compact representation, which is more suitable for clustering. To the best of our knowledge, this is the first attempt to develop auto-encode for clustering. As this deep architecture can learn a powerful non-linear mapping, the data can be well partitioned in the transformed space. The experimental results have also demonstrated the effectiveness of the proposed model. However, as is shown in Figure 4, some data are still mixed. This problem might be resolved by maximizing the difference among cluster centers in the code layer. Besides, a probability-based model in assigning data to their corresponding cluster centers may be a potential direction in future work, which can decrease the possibility of local optimal solution.

Acknowledgement. This work was jointly supported by National Basic Research Program of China (2012CB316300), National Natural Science Foundation of China (61175003, 61135002, 61203252), Tsinghua National Laboratory for Information Science and Technology Cross-discipline Foundation, and Hundred Talents Program of CAS.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. arXiv preprint arXiv:1206.5538 (2012)
2. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2004)
3. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786) (2006)
4. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31(3), 264–323 (1999)
5. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient backProp. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade, 2nd edn. LNCS, vol. 7700, pp. 9–48. Springer, Heidelberg (2012)
6. Li, Z., Yang, Y., Liu, J., Zhou, X., Lu, H.: Unsupervised feature selection using nonnegative spectral analysis. In: AAAI Conference on Artificial Intelligence (2012)
7. Ng, A.Y., Jordan, M.I., Weiss, Y., et al.: On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems 2, 849–856 (2002)
8. Plummer, M., Lovász, L.: Matching theory, vol. 121. North Holland (1986)
9. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8) (2000)
10. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: International Conference on Machine Learning, pp. 577–584 (2001)
11. Xu, W., Liu, X., Gong, Y.: Document clustering based on non-negative matrix factorization. In: ACM SIGIR Conference on Research and Development in Informaion Retrieval (2003)