K-AUTOENCODERS DEEP CLUSTERING

Yaniv Opochinsky, Shlomo E. Chazan, Sharon Gannot and Jacob Goldberger

Faculty of Engineering, Bar-Ilan University, Ramat-Gan, 5290002, Israel.

ABSTRACT

In this study we propose a deep clustering algorithm that extends the k-means algorithm. Each cluster is represented by an autoencoder instead of a single centroid vector. Each data point is associated with the autoencoder which yields the minimal reconstruction error. The optimal clustering is found by learning a set of autoencoders that minimize the global reconstruction mean-square error loss. The network architecture is a simplified version of a previous method that is based on mixture-of-experts. The proposed method is evaluated on standard image corpora and performs on par with state-of-the-art methods which are based on much more complicated network architectures.

Index Terms— clustering, autoencoders, deep networks

1. INTRODUCTION

k-means is a commonly used clustering algorithm. Clustering high-dimensional datasets is hard since the inter-point distances become less informative in high-dimensional spaces. As a result, representation learning is often used to map the input data into a low-dimensional feature space. In recent years, inspired by the success of deep neural networks in supervised learning, there have been many attempts to apply unsupervised deep learning approaches to clustering. Most methods involve clustering over the low-dimensional feature space of an autoencoder [1][2][3][4][5], a variational autoencoder [6] [7] or a Generative Adversarial Network (GAN) [8][9]. Recent in-depth overviews of deep clustering methods can be found in [10] and [11].

The most important facet of deep learning is that it allows a system to automatically discover the most suitable non-linear representations for a specified task. A deep version of k-means is based on learning a nonlinear data representation and applying k-means in the embedded space. However, a straightforward implementation of this deep k-means algorithm leads to a trivial solution where the feature vectors are collapsed to a single point in the embedded space and the centroids are collapsed into a single entity. For this reason, to avoid data collapse, the objective function of most deep clustering algorithms is composed of a clustering term computed in the embedded space and a regularization term in the form of a reconstruction error. Deep Embedded Clustering (DEC) [12] is first pre-trained using an autoencoder reconstruction loss and then optimizes cluster centroids in the embedded space through a Kullback-Leibler divergence loss. The Deep Clustering Network (DCN) [4] is another autoencoder-based method that uses k-means for clustering. Similar to the DEC, in the first phase, the network is pre-trained using the autoencoder reconstruction loss. However, in the second phase, in contrast to DEC, the network is jointly trained using a mathematical combination of the autoencoder reconstruction loss and the k-means clustering loss function. Thus, because strict cluster assignments were used during training (instead of probabilities such as in DEC) the method requires an alternation process between network training and cluster updates.

Here we propose a deep clustering algorithm where each cluster is represented by an autoencoder neural network and the clustering itself is performed by assigning the data point to the autoencoder which best reconstructs the input. The algorithm is a direct deep extension of the k-means algorithm where the cluster centroid is obtained in a data-driven manner instead of using a constant vector. The algorithm is a deep variant of the k-means algorithm that has the advantage of not suffering from data and centroid collapse and therefore does not require a regularization term. Previous algorithms have applied the mixture of experts paradigm [13] to represent each cluster by an autoencoder [14] [15]. We show here that this concept can be implemented by a much simpler network architecture with equally good clustering results.

2. K-AUTOENCODERS DEEP CLUSTERING

Consider the problem of clustering a set of n points $x_1, ..., x_n \in \mathbb{R}^d$ into k clusters. The k-means algorithm represents each cluster by a centroid. In our approach, rather than representing a cluster by a centroid, we represent each cluster by an autoencoder that specializes in reconstructing objects belonging to that cluster. If the dataset is properly clustered, we expect all the points assigned to the same cluster to be similar. Hence, the task of a cluster-specialized autoencoder should be significantly easier than using a single autoencoder for the entire data. We thus expect that good clustering should result in a small reconstruction error. The clustering itself is carried out by assigning each point to the autoencoder that best reconstructs it. We next describe the proposed clustering algorithm formally.

Denote the autoencoder associated with cluster i by $f_i(x;\theta_i)$ where θ_i is the parameter set of the network autoencoder. We can view the reconstructed object $f_i(x;\theta_i) \in \mathbb{R}^d$ as a data-driven centroid of cluster i that is tuned to the input x. The goal of the training procedure is to find a clustering of the data such that the reconstruction error of the most suitable autoencoder is minimized. To find the network parameters we jointly train the deep autoencoders. The clustering is thus computed by minimizing the following loss function:

$$L(\theta_1, \ldots, \theta_k) = \sum_{t=1}^{n} \min_{i=1}^{k} d(x_t, \hat{x}_t(i)) \tag{1}$$

such that $\hat{x}_t(i) = f_i(x_t;\theta_i)$ is the reconstruction of x_t by the i-th autoencoder. In our implementation we set

$$d(x_t, \hat{x}_t(i)) = \|x_t - \hat{x}_t(i)\|^2.$$

The gradient is:

$$\frac{\partial L}{\partial \theta_i} = \sum_{t \in N_i} (x_t - \hat{x}_t(i))^\top \frac{d}{d\theta_i} f_i(x_t;\theta_i) \tag{2}$$

where N_i is the set of all the data points that are best reconstructed by the i-th autoencoder, i.e.:

$$N_i = \{t | i = \arg\min_j \|x_t - \hat{x}_t(j)\|\}.$$

In the minimization of (1) we simultaneously perform data clustering and learn a 'centroid' representation for each cluster in the form of an autoencoder. Unlike most previously proposed deep clustering methods, there is no risk of collapsing to a trivial solution, where all the data points are mapped to the same cluster. Because our clustering goal is to minimize the reconstruction error, it is naturally better to use k different autoencoders for reconstruction. Hence, there is no need to add regularization terms to the loss function (which might influence the clustering accuracy) in order to to prevent data collapse.

The proposed clustering algorithm can be viewed as a deep extension of the k-means algorithm. Assume we replace each autoencoder in our network by a constant function $f_i(x_t, \theta_i) \equiv \mu_i \in \mathbb{R}^d$. In so doing, we obtain exactly the classical k-means algorithm. The cost function (1) is reduced to the standard k-means cost function:

$$L(\mu_1, \ldots, \mu_k) = \sum_{t=1}^{n} \min_{i=1}^{k} \|x_t - \mu_i\|^2. \quad (3)$$

The centroids found by the k-means algorithm are the constant vectors that minimize the reconstruction error. The proposed algorithm substitutes the constant centroid with a data-driven representation of the input computed by an autoencoder. To underscore the close relationship to k-means, we dub our algorithm the k-Deep-AutoEncoder (k-DAE) clustering.

Determining the number of clusters in a data set is a frequent problem in data clustering, and is a distinct issue from the process of actually solving the clustering problem. There are many methods to set the value of k in the k-means algorithm (e.g. the elbow method and the silhouette method). These methods can be easily adapted to the k-DAE clustering. The hard clustering cost function of k-means (3) can be replaced by a soft clustering which is implemented by the EM algorithm applied to learn a mixture of Gaussians. In a similar way, in the k-DAE clustering we can also replace the hard clustering by a soft clustering cost function:

$$L(\theta_1, \ldots, \theta_k) = -\sum_{t=1}^{n} \log(\sum_{i=1}^{k} \exp(-d(x_t, \hat{x}_t(i)))). \quad (4)$$

In recent years, network pre-training has been largely rendered obsolete for supervised tasks, given the availability of large labeled training datasets. However, for difficult optimization problems that unsupervised clustering tasks cannot handle (such as the one presented in (1)), initialization is still crucial. To initialize the parameters of the network, we first train a single autoencoder and use the layer-wise pre-training method, as described in [16], for training autoencoders. After training the autoencoder, we carry out a k-means clustering on the output of the bottleneck layer to obtain the initial clustering values. The k-means assigns a label to each data point. Note that in the pre-training procedure, a *single* autoencoder is trained on the entire database. The points that were assigned by the k-means algorithm to the i-th cluster, are next used to pre-train the i-th autoencoder $f_i(x; \theta_i)$. Once all the network parameters have been initialized by this pre-training procedure, the network parameters are jointly trained to minimize the autoencoding reconstruction error defined by the loss function (1). The architecture of the network trained by the k-DAE algorithm and the final clustering procedure are depicted in Fig. 1. The clustering algorithm is summarized in Table 1.

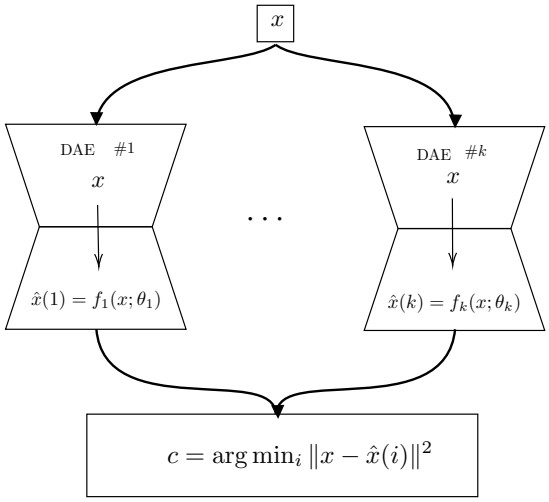

Fig. 1: A block diagram of the proposed k-DAE clustering.

A recent study proposed the Deep Autoencoder MIxture Clustering (DAMIC) deep clustering algorithm [14] which is based on the mixture-of-experts (MoE) framework [13]. The MoE model comprises several expert models and a gate model. Each of the experts provides a decision and the gate is a latent variable that selects the relevant expert based on the input data. In the DAMIC algorithm the experts are autoencoders where each autoencoder's expertise is to reconstruct a sample from the associated cluster. In contrast to the current approach, the DAMIC algorithm has an additional gate component which is a deep classifier that selects one of the autoencoders. The DAMIC clustering cost function is given by:

$$L(\theta_1, \ldots, \theta_k, \theta_c) = \\ -\sum_{t=1}^{n} \log\left(\sum_{i=1}^{k} p(c_t = i | x_t; \theta_c) \exp(-d(x_t, \hat{x}_t(i)))\right) \quad (5)$$

where c_t is a random variable representing the cluster of sample t, and θ_c is the parameter set of the deep classifier. After convergence, the classifier gate is used to assign each point to its cluster. A similar idea was proposed in [15] which also applied a mixture of autoencoders for clustering. Unlike [14] they used the latent representations learned by the autoencoders as the features for the clustering network. This enforces the autoencoders to encode discriminative information on the difference between clusters rather than cluster-based reconstruction information alone. They also used regularization terms to avoid data collapsing that need to be tuned for each dataset separately.

The crux of our approach is to show that the gate network is redundant here and we can apply a k-DAE clustering using a much simpler network architecture with similar and even slightly better performance. In applications of the MoE paradigm to either classification or regression tasks, there is no obvious way to decide which expert decision is better and we need to train a gate network to select the most suitable expert for each instance. In our setup, however, the reconstruction error of each expert is the natural measure to select the best expert and thus obviating the need for an additional gate network. By getting rid of the gate network, the clustering algorithm turns into a direct deep extension of the classical k-means algorithm.

Table 1: The k Deep AutoEncoders (k-DAE) Clustering algorithm.

Goal: clustering $x_1, \ldots, x_n \in \mathbb{R}^d$ into k clusters.

Network components:

- A set of autoencoders (one for each cluster):

$$x \to \hat{x}(i) = f_i(x; \theta_i), \qquad i = 1, \ldots, k$$

Pre-training:

- Train a single autoencoder for the entire dataset.
- Apply a k-means algorithm in the embedded space.
- Use the k-means clustering to initialize the network parameters.

Training: clustering is obtained by minimizing the reconstruction error:

$$L(\theta_1, \ldots, \theta_k) = \sum_{t=1}^{n} \min_i d(x_t, \hat{x}_t(i))$$

The final (hard) clustering is:

$$\hat{c}_t = \arg\min_{i=1}^{k} d(x_t, \hat{x}_t(i)), \qquad t = 1, ..., n.$$

3. EXPERIMENTS

In this section, we evaluate the clustering results of our approach. We carried out experiments on different datasets and compared the proposed method to the state-of-the-art standard and k-means related deep clustering algorithms.

3.1. Datasets

We tested several standard image datasets to demonstrate the general applicability of our approach. The MNIST dataset consists of 70,000 images (28×28 pixels, 10 classes) which contain hand-written digit images. The Fashion dataset [17] consists of 70,000 images with similar dimensions as the MNIST dataset and ten fashion classes. The USPS [18] handwritten digit database contains ten classes (0–9 digit characters) and each class has 1100 images.

3.2. Evaluation measures

The clustering performance of the methods was evaluated with respect to the following three standard measures: normalized mutual information (NMI) [19], adjusted Rand index (ARI) [20], and clustering accuracy (ACC) [19]. The NMI is an information-theoretic measure based on the mutual information of the ground-truth classes and the obtained clusters, normalized using the entropy of each. The ACC measures the proportion of data points for which the obtained clusters can be correctly mapped to ground-truth classes. Finally, the ARI is a variant of the Rand index that is adjusted for the chance grouping of elements. Note that NMI and ACC lie in the range of 0 to 1 where one is the perfect clustering result and zero the worst. ARI is a value between minus one to (plus) one, where one is the best clustering performance and minus one the worst. All results reported in this paper were obtained with the mean performance over 5 trials.

Table 2: Objective measures of clustering results of the MNIST, Fashion and USPS databases.

MNIST	k-DAE	DAMIC	DCN	DEC	KM	AE+KM
NMI	**0.86**	**0.86**	0.81	0.80	0.50	0.73
ARI	**0.82**	**0.82**	0.75	0.75	0.37	0.69
ACC	**0.88**	**0.88**	0.83	0.84	0.53	0.81

Fashion	k-DAE	DAMIC	DCN	DEC	KM	AE+KM
NMI	**0.65**	**0.65**	0.55	0.54	0.51	0.62
ARI	**0.48**	**0.48**	0.42	0.40	0.37	0.46
ACC	**0.60**	**0.60**	0.50	0.51	0.47	0.59

USPS	k-DAE	DAMIC	DCN	DEC	KM	AE+KM
NMI	**0.80**	0.78	0.68	0.77	0.63	0.68
ARI	**0.71**	0.70	-	-	0.55	0.61
ACC	**0.77**	0.75	0.69	0.76	0.67	0.71

3.3. Baseline methods

The proposed algorithm (k-DAE) was compared to the following methods:

K-means (KM): The classic k-means.

Autoencoder followed by K-means (AE+KM): The method first learns a single autoencoder and then applies k-means clustering on the embedded space. This is the initialization step of the proposed algorithm.

Deep Clustering Network (DCN): The algorithm performs joint reconstruction and k-means clustering at the same time. The loss comprises penalties on both the reconstruction and the clustering losses [4].

Deep Embedding Clustering (DEC): The algorithm performs joint embedding and clustering in the embedded space. The loss function only contains a clustering loss term [12].

Deep Autoencoder MIxture Clustering (DAMIC): This algorithm applies an autoencoder deep clustering with an additional gate network that performs the actual clustering [14].

3.4. Network implementation

The proposed method was implemented with the deep learning toolbox Tensorflow [21]. All neurons in the proposed architecture except the output layer used exponential linear unit (elu) as the transfer function. The output layer in all the DAEs was the sigmoid function. Batch normalization [22] was utilized on all layers, and the ADAM optimizer [23] was used for both the pre-training as well as the training phase. In the pre-training phase, the DAE networks were trained with the binary cross-entropy loss function. We set the number of epochs for the training phases to be 50. However, *early stopping* was used to prevent mis-convergence of the loss. The mini-batch size was 256.

3.5. Results

For the MNIST database we used a 5-layer network with 1024, 256, 10, 256, 1024 neurons. Similar networks were used for the other two

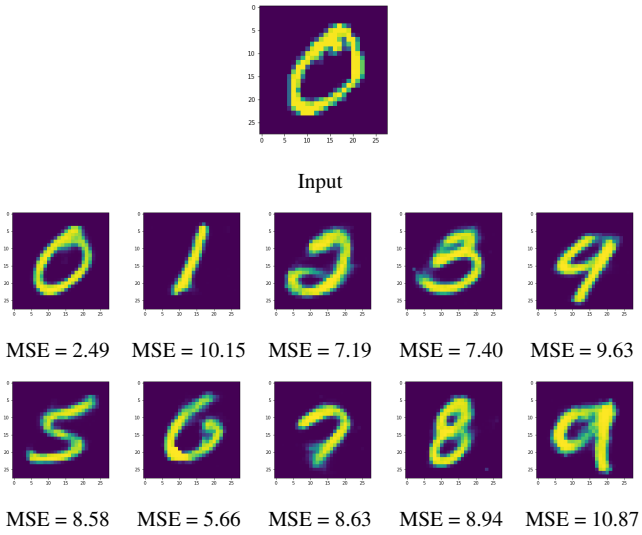

Fig. 2: The outputs and the reconstruction error of 10 DAEs with a input digit '0'.

datasets. Table 2 presents the clustering results of the proposed k-DAE method and the baseline algorithms. The results indicate that k-DAE outperforms previous attempts to define deep versions of k-means and is on par (and in some cases slightly better) with DIMAC [14] that implemented similar concepts with a much more complicated network architecture. To demonstrate the expertise of each one of the DAE in the MNIST dataset we conducted the following experiment. An image of the digit '0' was fed to the network. The outputs of the different DAEs are depicted in Fig. 2. It can be seen that each DAE memorizes its cluster members and tries to reconstruct an image that looks like them.

Note that the best reported clustering results on the MNIST data achieved by the VaDE algorithm [6] that applies variational autoencoder modeling assuming a mixture of Gaussians distribution of the latent random variable. We compared our method to state-of-the-art deep k-means-based algorithms using the same network architecture and parameter initialization and showed improvement in performance. The VaDE algorithm belongs to a different family of algorithms with different network architecture and parameter initialization strategies. Hence, a direct performance comparison is difficult since it is heavily dependent on the implementations. It is worth noting that in the Fashion dataset, our results even outperforms the VaDE and the DEC with data augmentation (DEC-DA) algorithms results [24] in this dataset.

4. CONCLUSION

In this study we presented a clustering technique that leverages the strength of deep neural networks. The cost function we optimize is very similar to the cost function of the k-means algorithm. The only difference is that the constant centroid is replaced by an autoencoder. This enables a much richer representation of each cluster. The algorithm does not cause a data collapsing problem. Hence, there is no need for regularization terms that need to be tuned for each dataset separately. Compared to previous attempts to use a set of autoencoders (e.g. DAMIC [14]), our architecture is much simpler and easily trained. Experiments on a variety of real datasets illustrated the strong performance of the proposed algorithm over the other methods.

5. REFERENCES

[1] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[3] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self augmented training," *arXiv preprint arXiv:1702.08720*, 2017.

[4] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards K-means-friendly spaces: Simultaneous deep learning and clustering," in *International Conference on Machine Learning (ICML)*, 2017.

[5] S. Fogel, H. Averbuch-Elor, D. Cohen-Or, and J. Goldberger, "Clustering-driven deep embedding with pairwise constraints," *IEEE Computer Graphics and Applications*, vol. 39, no. 4, pp. 16–27, 2019.

[6] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *arXiv preprint arXiv:1611.05148*, 2016.

[7] N. Dilokthanakul1, P. Mediano, M. Garnelo, M. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoder," *arXiv preprint arXiv:1611.02648*, 2016.

[8] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," *arXiv preprint arXiv: 1511.06390*, 2015.

[9] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Adv. Neural Inf. Process. Syst.*, 2016.

[10] E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *arXiv preprint arXiv:1801.07648*, 2018.

[11] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, 07 2018.

[12] J. Xie, R. Girshick, and A. Farhad, "Unsupervised deep embedding for clustering analysis," in *International Conference on Machine Learning (ICML)*, 2016.

[13] R. Jacobs, S. Nowlan M. Jordan, and G. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[14] S. Chazan, S. Gannot, and J. Goldberger, "Deep clustering based on a mixture of autoencoder," in *IEEE Machine Learning for Signal Processing Workshop (MLSP)*, 2019.

[15] D. Zhang, Y. Sun, B. Eriksson, and L. Balzano, "Deep unsupervised clustering using mixture of autoencoders," *arXiv preprint arXiv:1712.07788*, 2017.

[16] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[17] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[18] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[19] D. Cai, X. He, and J. Han., "Locally consistent concept factorization for document clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 902–913, 2011.

[20] K. Y. Yeung and W. L W. L. Ruzzo, "Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data," *Bioinformatics*, vol. 19, no. 9, pp. 763–774, 2001.

[21] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow: a system for large-scale machine learning.," in *OSDI*, 2016, vol. 16, pp. 265–283.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[23] D. Kingma and J. Ba, "ADAM: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] X. Guo, E. Zhu, X. Liu, and J. Yin, "Deep embedded clustering with data augmentation," in *Asian Conference on Machine Learning (ACML)*, 2018.