Deep k-Means: Jointly clustering with k-Means and learning representations

Maziar Moradi Fard, Thibaut Thonet, Éric Gaussier

Version of Record: https://www.sciencedirect.com/science/article/pii/S0167865520302749
Manuscript_e9cd2c4f2e4c74b63605b0bd90293987

Pattern Recognition Letters
journal homepage: www.elsevier.com

Deep k-Means: Jointly clustering with k-Means and learning representations

Maziar Moradi Fard[a], Thibaut Thonet[a,b,**], Eric Gaussier[a]

[a] *Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, F-38000 Grenoble, France*
[b] *NAVER LABS Europe, F-38240 Meylan, France*

ABSTRACT

We study in this paper the problem of jointly clustering and learning representations. As several previous studies have shown, learning representations that are both faithful to the data to be clustered and adapted to the clustering algorithm can lead to better clustering performance, all the more so that the two tasks are performed jointly. We propose here such an approach for k-Means clustering based on a continuous reparametrization of the objective function that leads to a truly joint solution. The behavior of our approach is illustrated on various datasets showing its efficacy in learning representations for objects while clustering them.

1. Introduction

Clustering is a long-standing problem in the machine learning and data mining fields, and thus accordingly fostered abundant research. Traditional clustering methods, *e.g.*, k-Means (MacQueen, 1967) and Gaussian Mixture Models (GMMs) (Bishop, 2006), fully rely on the original data representations and may then be ineffective when the data points (*e.g.*, images and text documents) live in a high-dimensional space – a problem commonly known as the curse of dimensionality. Significant progress has been made in the last decade or so to learn better, low-dimensional data representations (Hinton and Salakhutdinov, 2006). The most successful techniques to achieve such high-quality representations rely on deep neural networks (DNNs), which apply successive non-linear transformations to the data in order to obtain increasingly high-level features. Auto-encoders (AEs) are a special instance of DNNs which are trained to embed the data into a (usually dense and low-dimensional) vector at the bottleneck of the network, and then attempt to reconstruct the input based on this vector. The appeal of AEs lies in the fact that they are able to learn representations in a fully unsupervised way. The representation learning breakthrough enabled by DNNs spurred the recent development of numerous deep clustering approaches which aim at jointly learning the data points' representations as well as their cluster assignments.

In this study, we specifically focus on the k-Means-related deep clustering problem. Contrary to previous approaches that alternate between continuous gradient updates and discrete cluster assignment steps (Yang et al., 2017), we show here that one can solely rely on gradient updates to learn, truly jointly, representations and clustering parameters. This ultimately leads to a better deep k-Means method which is also more scalable as it can fully benefit from the efficiency of stochastic gradient descent (SGD). In addition, we perform a careful comparison of different methods by *(a)* relying on the same auto-encoders, as the choice of auto-encoders impacts the results obtained, *(b)* tuning the hyperparameters of each method on a small validation set, instead of setting them without clear criteria, and *(c)* enforcing, whenever possible, that the same initialization and sequence of SGD minibatches are used by the different methods. The last point is crucial to compare different methods as these two factors play an important role and the variance of each method is usually not negligible.

2. Related work

In the wake of the groundbreaking results obtained by DNNs in computer vision, several deep clustering algorithms were specifically designed for image clustering (Yang et al., 2016; Chang et al., 2017; Dizaji et al., 2017; Hu et al., 2017; Hsu and Lin, 2018). These works have in common the exploitation of Convolutional Neural Networks (CNNs), which extensively contributed to last decade's significant advances in computer vision. Inspired by agglomerative clustering, Yang et al. (2016)

**Corresponding author: Tel.: +33476614166
e-mail: thibaut.thonet@gmail.com (Thibaut Thonet)

proposed a recurrent process which successively merges clusters and learn image representations based on CNNs. In (Chang et al., 2017), the clustering problem is formulated as binary pairwise-classification so as to identify the pairs of images which should belong to the same cluster. Due to the unsupervised nature of clustering, the CNN-based classifier in this approach is only trained on noisily labeled examples obtained by selecting increasingly difficult samples in a curriculum learning fashion. Dizaji et al. (2017) jointly trained a CNN auto-encoder and a multinomial logistic regression model applied to the AE's latent space. Similarly, Hsu and Lin (2018) alternate between representation learning and clustering where mini-batch k-Means is utilized as the clustering component. Differently from these works, Hu et al. (2017) proposed an information-theoretic framework based on data augmentation to learn discrete representations, which may be applied to clustering or hash learning. Although these different algorithms obtained state-of-the-art results on image clustering (Aljalbout et al., 2018), their ability to generalize to other types of data (e.g., text documents) is not guaranteed due to their reliance on essentially image-specific techniques – Convolutional Neural Network architectures and data augmentation.

Nonetheless, many general-purpose – non-image-specific – approaches to deep clustering have also been recently designed (Huang et al., 2014; Peng et al., 2016; Xie et al., 2016; Dilokthanakul et al., 2017; Guo et al., 2017; Hu et al., 2017; Ji et al., 2017; Jiang et al., 2017; Peng et al., 2017; Yang et al., 2017). Generative models were proposed in (Dilokthanakul et al., 2017; Jiang et al., 2017) which combine variational AEs and GMMs to perform clustering. Alternatively, Peng et al. (2016, 2017); Ji et al. (2017) framed deep clustering as a subspace clustering problem in which the mapping from the original data space to a low-dimensional subspace is learned by a DNN. Xie et al. (2016) defined the Deep Embedded Clustering (DEC) method which simultaneously updates the data points' representations, initialized from a pre-trained AE, and cluster centers. DEC uses soft assignments which are optimized to match stricter assignments through a Kullback-Leibler divergence loss. IDEC was subsequently proposed in (Guo et al., 2017) as an improvement to DEC by integrating the AE's reconstruction error in the objective function.

Few approaches were directly influenced by k-Means clustering (Huang et al., 2014; Yang et al., 2017). The Deep Embedding Network (DEN) model (Huang et al., 2014) first learns representations from an AE while enforcing locality-preserving constraints and group sparsity; clusters are then obtained by simply applying k-Means to these representations. Yet, as representation learning is decoupled from clustering, the performance is not as good as the one obtained by methods that rely on a joint approach. Besides (Hsu and Lin, 2018), mentioned before in the context of images, the only study, to our knowledge, that directly addresses the problem of jointly learning representations and clustering with k-Means (and not an approximation of it) is the Deep Clustering Network (DCN) approach (Yang et al., 2017). However, as in (Hsu and Lin, 2018), DCN alternatively learns (rather than jointly learns) the object representations, the cluster centroids and the cluster assignments, the latter being based on discrete optimization steps which cannot benefit from the efficiency of stochastic gradient descent. The approach proposed here, entitled *Deep k-Means* (DKM), addresses this problem.

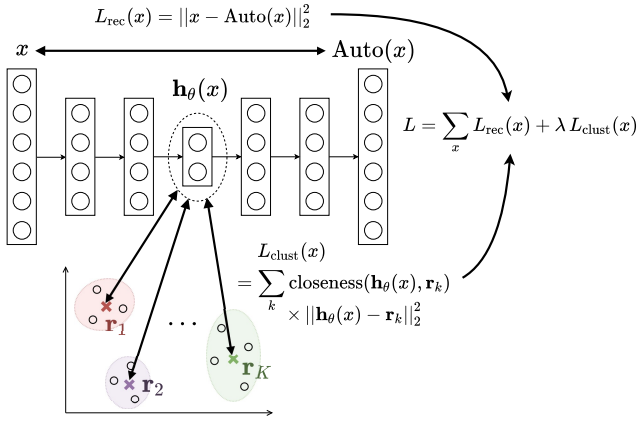


Fig. 1. Overview of the proposed Deep k-Means approach instantiated with losses based on the Euclidean distance.

3. Deep k-Means

In the remainder, x denotes an object from a set $\mathcal{X}$ of objects to be clustered. $\mathbb{R}^p$ represents the space in which learned data representations are to be embedded. K is the number of clusters to be obtained, $\mathbf{r}_k \in \mathbb{R}^p$ the representative of cluster k, $1 \leq k \leq K$, and $\mathcal{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_K\}$ the set of representatives. Functions f and g define some distance in $\mathbb{R}^p$ which are assumed to be fully differentiable wrt their variables. For any vector $\mathbf{y} \in \mathbb{R}^p$, $c_f(\mathbf{y}; \mathcal{R})$ gives the closest representative of $\mathbf{y}$ according to f.

The deep k-Means problem takes the following form:

$$\begin{cases} \min\limits_{\mathcal{R},\theta} \sum\limits_{x \in \mathcal{X}} g(x, A(x;\theta)) + \lambda f(\mathbf{h}_\theta(x), c_f(\mathbf{h}_\theta(x); \mathcal{R})), \\ \text{with: } c_f(\mathbf{h}_\theta(x); \mathcal{R}) = \operatorname{argmin}\limits_{\mathbf{r} \in \mathcal{R}} f(\mathbf{h}_\theta(x), \mathbf{r}). \end{cases} \tag{1}$$

g measures the error between an object x and its reconstruction $A(x;\theta)$ provided by an auto-encoder, θ representing the set of the auto-encoder's parameters. A regularization term on θ can be included in the definition of g. However, as most auto-encoders do not use regularization, we dispense with such a term here. $\mathbf{h}_\theta(x)$ denotes the representation of x in $\mathbb{R}^p$ output by the AE's encoder part and $f(\mathbf{h}_\theta(x), c_f(\mathbf{h}_\theta(x); \mathcal{R}))$ is the clustering loss corresponding to the k-Means objective function in the embedding space. Finally, λ in Problem (1) regulates the trade-off between seeking good representations for x – *i.e.*, representations that are faithful to the original examples – and representations that are useful for clustering purposes. Similar optimization problems can be formulated when f and g are similarity functions or a mix of similarity and distance functions. The approach proposed here directly applies to such cases. Figure 1 illustrates the overall framework retained in this study with f and g both based on the Euclidean distance. The closeness term in the clustering loss will be further clarified below.

The original k-Means problem is recovered by considering that the encoder and decoder used in the auto-encoder correspond to the identity function, hence $\mathbf{h}_\theta(x) = x$, and that both

f and g correspond to the Euclidean distance. In such a case, $g(x, A(x;\theta)) = 0$, x and $\mathbf{r}$ lie in the same vector space, and the problem in (1) takes the form:

$$\begin{cases} \min\limits_{\mathcal{R}} \sum\limits_{x\in\mathcal{X}} \|x - c(x;\mathcal{R})\|_2^2, \\ \text{with: } c(x;\mathcal{R}) = \operatorname{argmin}\limits_{\mathbf{r}\in\mathcal{R}} \|x - \mathbf{r}\|_2^2. \end{cases} \tag{2}$$

This is exactly the standard k-Means problem. The formulation in (1) thus encompasses what is generally referred to as *generalized k-Means*, in which any dissimilarity can be used in place of the Euclidean distance. Inasmuch as the formulation and the solution for the generalized version and the standard one when learning representations is the same, we simply refer to Problem (1) as deep k-Means.

3.1. Continuous generalization of Deep k-Means

We now introduce a parameterized version of the above problem that constitutes a *continuous generalization*, whereby we mean here that all functions considered are continuous wrt the introduced parameter.[1] To do so, we first note that the clustering objective function can be rewritten as $f(\mathbf{h}_\theta(x), c_f(\mathbf{h}_\theta(x);\mathcal{R})) = \sum_{k=1}^{K} f_k(\mathbf{h}_\theta(x);\mathcal{R})$ with:

$$f_k(\mathbf{h}_\theta(x);\mathcal{R}) = \begin{cases} f(\mathbf{h}_\theta(x), \mathbf{r}_k) & \text{if } \mathbf{r}_k = c_f(\mathbf{h}_\theta(x);\mathcal{R}) \\ 0 & \text{otherwise} \end{cases}$$

Let us now assume that we know some function $G_{k,f}(\mathbf{h}_\theta(x), \alpha;\mathcal{R})$ such that:

(i) $G_{k,f}$ is differentiable wrt to $\theta, \mathcal{R}$ and continuous wrt α (differentiability wrt $\mathcal{R}$ means differentiability wrt to all dimensions of $\mathbf{r}_k$, $1 \le k \le K$);

(ii) $\exists \alpha_0 \in \mathbb{R} \cup \{-\infty, +\infty\}$ such that:

$$\lim_{\alpha\to\alpha_0} G_{k,f}(\mathbf{h}_\theta(x), \alpha;\mathcal{R}) = \begin{cases} 1 & \text{if } \mathbf{r}_k = c_f(\mathbf{h}_\theta(x);\mathcal{R}) \\ 0 & \text{otherwise} \end{cases}$$

Then, one has, $\forall x \in \mathcal{X}$: $\lim\limits_{\alpha\to\alpha_0} f(\mathbf{h}_\theta(x), \mathbf{r}_k)\, G_{k,f}(\mathbf{h}_\theta(x), \alpha;\mathcal{R}) = f_k(\mathbf{h}_\theta(x);\mathcal{R})$, showing that the problem in (1) is equivalent to:

$$\min_{\mathcal{R},\theta} \overbrace{\lim_{\alpha\to\alpha_0} \sum_{x\in\mathcal{X}} g(x, A(x;\theta)) \; + \lambda \sum_{k=1}^{K} f(\mathbf{h}_\theta(x), \mathbf{r}_k)\, G_{k,f}(\mathbf{h}_\theta(x), \alpha;\mathcal{R})}^{\mathcal{F}(\mathcal{X},\alpha;\theta,\mathcal{R})} \tag{3}$$

All functions in the above formulation are fully differentiable wrt both θ and $\mathcal{R}$. One can thus estimate θ and $\mathcal{R}$ through a simple, joint optimization based on stochastic gradient descent (SGD) for a given α:

$$(\theta,\, \mathcal{R}) \leftarrow (\theta,\, \mathcal{R}) - \eta\, \frac{1}{|\tilde{\mathcal{X}}|}\, \nabla_{(\theta,\mathcal{R})} \mathcal{F}(\tilde{\mathcal{X}}, \alpha; \theta, \mathcal{R}) \tag{4}$$

with η the learning rate and $\tilde{\mathcal{X}}$ a random mini-batch of $\mathcal{X}$.

[1] Note that, independently from this work, a similar relaxation has been previously proposed in (Agustsson et al., 2017) – wherein soft-to-hard quantization is performed on an embedding space learned by an AE for compression. However, given the different nature of the goal here – clustering – our proposed learning framework substantially differs from theirs.

3.2. Choice of $G_{k,f}$

Several choices are possible for $G_{k,f}$. A simple choice, used throughout this study, is based on a parameterized softmax function. The fact that the softmax function can be used as a differentiable surrogate to argmax or argmin is well known and has been applied in different contexts, as in the recently proposed Gumbel-softmax distribution employed to approximate categorical samples (Jang et al., 2017; Maddison et al., 2017). The parameterized softmax function which we adopted takes the following form:

$$G_{k,f}(\mathbf{h}_\theta(x), \alpha;\mathcal{R}) = \frac{e^{-\alpha f(\mathbf{h}_\theta(x), \mathbf{r}_k)}}{\sum_{k'=1}^{K} e^{-\alpha f(\mathbf{h}_\theta(x), \mathbf{r}_{k'})}} \tag{5}$$

with $\alpha \in [0, +\infty)$. The function $G_{k,f}$ defined by Eq. 5 is differentiable wrt $\theta, \mathcal{R}$ and α (condition *(i)*) as it is a composition of functions differentiable wrt these variables. Furthermore, one has:

Property 1. *(condition (ii)) If $c_f(\mathbf{h}_\theta(x);\mathcal{R})$ is unique for all $x \in \mathcal{X}$, then:*

$$\lim_{\alpha\to+\infty} G_{k,f}(\mathbf{h}_\theta(x), \alpha;\mathcal{R}) = \begin{cases} 1 \text{ if } \mathbf{r}_k = c_f(\mathbf{h}_\theta(x);\mathcal{R}) \\ 0 \text{ otherwise} \end{cases}$$

The proof, which is straightforward, is detailed in the Supplementary Material.

The assumption that $c_f(\mathbf{h}_\theta(x);\mathcal{R})$ is unique for all objects is necessary for $G_{k,f}$ to take on binary values in the limit; it is not necessary to hold for small values of α. In the unlikely event that the above assumption does not hold for some x and large α, one can slightly perturbate the representatives equidistant to x prior to updating them. We have never encountered this situation in practice.

Finally, Eq. 5 defines a valid (according to conditions (*i*) and (*ii*)) function $G_{k,f}$ that can be used to solve the deep k-Means problem (3). We adopt this function in the remainder of this study.

3.3. Choice of α

The parameter α can be defined in different ways. Indeed, α can play the role of an inverse temperature such that, when α is 0, each data point in the embedding space is equally close, through $G_{k,f}$, to all the representatives (corresponding to a completely soft assignment), whereas when α is $+\infty$, the assignment is hard. In the first case, for the deep k-Means optimization problem, all representatives are equal and set to the point $\mathbf{r} \in \mathbb{R}^p$ that minimizes $\sum_{x\in\mathcal{X}} f(\mathbf{h}_\theta(x), \mathbf{r})$. In the second case, the solution corresponds to exactly performing k-Means in the embedding space, the latter being learned jointly with the clustering process. Following a deterministic annealing approach (Rose et al., 1990), one can start with a low value of α (close to 0), and gradually increase it till a sufficiently large value is obtained. At first, representatives are randomly initialized. As the problem is smooth when α is close to 0, different initializations are likely to lead to the same local minimum in the first iteration; this local minimum is used for the new values of the representatives for the second iteration, and so on. The continuity of $G_{k,f}$ wrt α implies that,

Algorithm 1: Deep k-Means algorithm

Input: data $\mathcal{X}$, number of clusters K, balancing parameter λ, scheme for α, number of epochs T, number of minibatches N, learning rate η
Output: autoencoder parameters θ, cluster representatives $\mathcal{R}$
Initialize θ and $\mathbf{r}_k$, $1 \leq k \leq K$ (randomly or through pretraining)
for $\alpha = m_\alpha$ **to** M_α **do** `# inverse temperature`
  for $t = 1$ **to** T **do** `# epochs per` α
    for $n = 1$ **to** N **do** `# minibatches`
      Draw a minibatch $\tilde{\mathcal{X}} \subset \mathcal{X}$
      Update $(\theta, \mathcal{R})$ using SGD (Eq. 4)
    end
  end
end

provided the increment in α is not too important, one evolves smoothly from the initial local minimum to the last one. In the above deterministic annealing scheme, α allows one to initialize cluster representatives. The initialization of the auto-encoder can as well have an important impact on the results obtained and prior studies (*e.g.*, Huang et al. (2014); Xie et al. (2016); Guo et al. (2017); Yang et al. (2017)) have relied on pretraining for this matter. In such a case, one can choose a high value for α to directly obtain the behavior of the k-Means algorithm in the embedding space after pretraining. We evaluate both approaches in our experiments.

Algorithm 1 summarizes the deep k-Means algorithm for the deterministic annealing scheme, where m_α (respectively M_α) denote the minimum (respectively maximum) value of α, and T is the number of epochs per each value of α for the stochastic gradient updates. Even though M_α is finite, it can be set sufficiently large to obtain in practice a hard assignment to representatives. Alternatively, when using pretraining, one sets $m_\alpha = M_\alpha$ (*i.e.*, a constant α is used).

3.4. Scaling-down phenomenon

The loss functions defined in 1 and 3 – as well as the loss used in the DCN approach (Yang et al., 2017) and potentially in other approaches – may in theory induce a behavior in the learning procedure where the clustering loss can be made arbitrarily small while preserving the reconstruction capacity of the AE by "scaling down" the subspace where the object embeddings and the cluster representatives live – thus reducing the distance between embeddings and representatives. This is not a problem *per se* as the clustering problem is still well defined and the clusters obtained are still meaningful. This situation indeed contrasts with the one where all objects and cluster representatives collapse into a single point, a situation which can happen when no reconstruction loss is used, as in Xie et al. (2016) and Yang et al. (2016). In such a case, no valid cluster is obtained.

We tested L2 regularization on the auto-encoder parameters to assess whether scaling down happens or not in practice. Indeed, by symmetry of the encoder and decoder, having small weights in the encoder, leading to scaling down, requires having large weights in the decoder for reconstruction; L2 regularization penalizes such large weights and thus prevents scaling down in the encoder. We have however not observed any difference in our experiments with the case where no regularization is used, showing that the scaling-down phenomenon does not happen in practice. For the sake of simplicity, we dispense with L2 regularization in the remainder.

4. Experiments

In order to evaluate the clustering results of our approach, we conducted experiments on different datasets and compared it against state-of-the-art standard and k-Means-related deep clustering models.

4.1. Datasets

The datasets used in the experiments are standard clustering benchmark collections. We considered both image and text datasets to demonstrate the general applicability of our approach. Image datasets consist of **MNIST** (70,000 images, 28×28 pixels, 10 classes) and **USPS** (9,298 images, 16×16 pixels, 10 classes) which both contain hand-written digit images. We reshaped the images to one-dimensional vectors and normalized the pixel intensity levels (between 0 and 1 for MNIST, and between -1 and 1 for USPS). The text collections we considered are the 20 Newsgroups dataset (hereafter, **20NEWS**) and the RCV1-v2 dataset (hereafter, **RCV1**). For 20NEWS, we used the whole dataset comprising 18,846 documents labeled into 20 different classes. Similarly to (Xie et al., 2016; Guo et al., 2017), we sampled from the full RCV1-v2 collection a random subset of 10,000 documents, each of which pertains to only one of the four largest classes. Because of the text datasets' sparsity, and as proposed in (Xie et al., 2016), we selected the 2000 words with the highest tf-idf values to represent each document.

4.2. Baselines and deep k-Means variants

Clustering models may use different strategies and different clustering losses, leading to different properties. As our goal in this work is to study the k-Means clustering algorithm in embedding spaces, we focus on the family of k-Means-related models and compare our approach against state-of-the-art models from this family, using both standard and deep clustering models. For the standard clustering methods, we used: the k-Means clustering approach (MacQueen, 1967) with initial cluster center selection (Arthur and Vassilvitskii, 2007), denoted **KM**; an approach denoted as **AE-KM** in which dimensionality reduction is first performed using an auto-encoder followed by k-Means applied to the learned representations.[2] We compared as well against the only previous, "true" deep clustering k-Means-related method, the Deep Clustering Network (DCN) approach described in (Yang et al., 2017). DCN is, to the best of our

[2] We did not consider variational auto-encoders (Kingma and Welling, 2014) in our baselines as Jiang et al. (2017) previously compared variational AE + GMM and "standard" AE + GMM, and found that the latter consistently outperformed the former.

knowledge, the current most competitive clustering algorithm among k-Means-related models.

In addition, we consider here the Improved Deep Embedded Clustering (IDEC) model (Guo et al., 2017) as an additional baseline. IDEC is a general-purpose state-of-the-art approach in the deep clustering family. It is an improved version of the DEC model (Xie et al., 2016) and thus constitutes a strong baseline. For both DCN and IDEC, we studied two variants: with pretraining (**DCN$^{\mathbf{p}}$** and **IDEC$^{\mathbf{p}}$**) and without pretraining (**DCN$^{\mathbf{np}}$** and **IDEC$^{\mathbf{np}}$**). The pretraining we performed here simply consists in initializing the weights by training the auto-encoder on the data to minimize the reconstruction loss in an end-to-end fashion – greedy layer-wise pretraining (Bengio et al., 2006) did not lead to improved clustering in our preliminary experiments.

The proposed Deep k-Means (DKM) is, as DCN, a "true" k-Means approach in the embedding space; it jointly learns AE-based representations and relaxes the k-Means problem by introducing a parameterized softmax as a differentiable surrogate to k-Means argmin. In the experiments, we considered two variants of this approach. **DKM$^{\mathbf{a}}$** implements an annealing strategy for the inverse temperature α and does not rely on pretraining. The scheme we used for the evolution of the inverse temperature α in DKM$^{\mathrm{a}}$ is given by the following recursive sequence: $\alpha_{n+1} = 2^{1/\log(n)^2} \times \alpha_n$ with $m_\alpha = \alpha_1 = 0.1$. The rationale behind the choice of this scheme is that we want α to spend more iterations on smaller values and less on larger values while preserving a gentle slope. Alternatively, we studied the variant **DKM$^{\mathbf{p}}$** which is initialized by pretraining an auto-encoder and then follows Algorithm 1 with a constant α such that $m_\alpha = M_\alpha = 1000$. Such a high α is equivalent to having hard cluster assignments while maintaining the differentiability of the optimization problem.

Implementation details.. For IDEC, we used the Keras code shared by their authors.[3] Our own code for DKM is based on TensorFlow. To enable full control of the comparison between DCN and DKM – DCN being the closest competitor to DKM – we also re-implemented DCN in TensorFlow. The code for both DKM and DCN is available online.[4]

Choice of f and g.. The functions f and g in Problem (1) define which distance functions is used for the clustering loss and reconstruction error, respectively. In this study, both f and g are simply instantiated with the Euclidean distance on all datasets. For the sake of comprehensiveness, we report in the supplementary material results for the cosine distance on 20NEWS.

4.3. Experimental setup

Auto-encoder description and training details.. The auto-encoder we used in the experiments is the same across all datasets and is borrowed from previous deep clustering studies (Xie et al., 2016; Guo et al., 2017). Its encoder is a fully-connected multilayer perceptron with dimensions d-500-500-2000-K, where d is the original data space dimension and K is the number of clusters to obtain. The decoder is a mirrored version of the encoder. All layers except the one preceding the embedding layer and the one preceding the output layer are applied a ReLU activation function (Nair and Hinton, 2010) before being fed to the next layer. For the sake of simplicity, we did not rely on any complementary training or regularization strategies such as batch normalization or dropout. The auto-encoder weights are initialized following the Xavier scheme (Glorot and Bengio, 2010). For all deep clustering approaches, the training is based on the Adam optimizer (Kingma and Ba, 2015) with standard learning rate $\eta = 0.001$ and momentum rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The minibatch size is set to 256 on all datasets following (Guo et al., 2017). We emphasize that we chose exactly the same training configuration for all models to facilitate a fair comparison.

The number of pretraining epochs is set to 50 for all models relying on pretraining. The number of fine-tuning epochs for DCN$^{\mathrm{p}}$ and IDEC$^{\mathrm{p}}$ is fixed to 50 (or equivalently in terms of iterations: 50 times the number of minibatches). We set the number of training epochs for DCN$^{\mathrm{np}}$ and IDEC$^{\mathrm{np}}$ to 200. For DKM$^{\mathrm{a}}$, we used the 40 terms of the sequence α described in Section 4.2 as the annealing scheme and performed 5 epochs for each α term (*i.e.*, 200 epochs in total). DKM$^{\mathrm{p}}$ is fine-tuned by performing 100 epochs with constant $\alpha = 1000$. The cluster representatives are initialized randomly from a uniform distribution $U(-1, 1)$ for models without pretraining. In case of pretraining, the cluster representatives are initialized by applying k-Means to the pretrained embedding space.

Hyperparameter selection.. The hyperparameters λ for DCN and DKM and γ for IDEC, that define the trade-off between the reconstruction and the clustering error in the loss function, were determined by performing a line search on the set $\{10^i \mid i \in [-4, 3]\}$. To do so, we randomly split each dataset into a validation set (10% of the data) and a test set (90%). Each model is trained on the whole data and only the validation set labels are leveraged in the line search to identify the optimal λ or γ (optimality is measured with respect to the clustering accuracy metric). We provide the validation-optimal λ and γ obtained for each model and dataset in the supplementary material. The performance reported in the following sections corresponds to the evaluation performed only on the *held-out test set*.

While one might argue that such procedure affects the unsupervised nature of the clustering approaches, we believe that a clear and transparent hyperparameter selection methodology is preferable to a vague or hidden one. Moreover, although we did not explore such possibility in this study, it might be possible to define this trade-off hyperparameter in a data-driven way.

Experimental protocol.. We observed in pilot experiments that the clustering performance of the different models is subject to non-negligible variance from one run to another. This variance is due to the randomness in the initialization and in the minibatch sampling for the stochastic optimizer. When pretraining is used,

[3] `https://github.com/XifengGuo/IDEC-toy`. We used this version instead of `https://github.com/XifengGuo/IDEC` as only the former enables auto-encoder pretraining in a non-layer-wise fashion.

[4] `https://github.com/MaziarMF/deep-k-means`

the variance of the general pretraining phase and that of the model-specific fine-tuning phase add up, which makes it difficult to draw any confident conclusion about the clustering ability of a model. To alleviate this issue, we compared the different approaches using seeded runs whenever this was possible. This has the advantage of removing the variance of pretraining as seeds guarantee exactly the same results at the end of pretraining (since the same pretraining is performed for the different models). Additionally, it ensures that the same sequence of minibatches will be sampled. In practice, we used seeds for the models implemented in TensorFlow (KM, AE-KM, DCN and DKM). Because of implementation differences, seeds could not give the same pretraining states in the Keras-based IDEC. All in all, we randomly selected 10 seeds and for each model performed one run per seed. Additionally, to account for the remaining variance and to report statistical significance, we performed a Student's t-test from the 10 collected samples (i.e., runs).

4.4. Clustering results

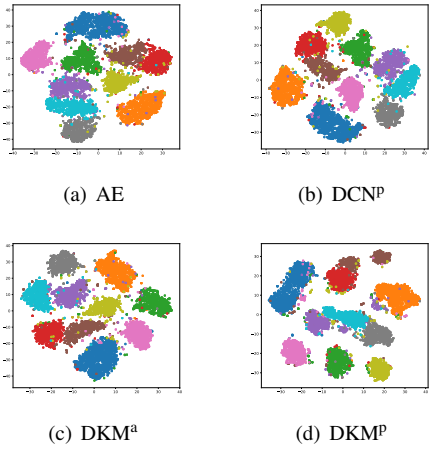
The results for the evaluation of the k-Means-related clustering methods on the different benchmark datasets are summarized in Table 1. The clustering performance is evaluated with respect to two standard measures (Cai et al., 2011): Normalized Mutual Information (NMI) and the clustering accuracy (ACC). We report for each dataset/method pair the average and standard deviation of these metrics computed over 10 runs and conduct significance testing as previously described in the experimental protocol. The bold (resp. underlined) values in each column of Table 1 correspond to results with no statistically significant difference ($p > 0.05$) to the best result with (resp. without) pretraining for the corresponding dataset/metric.

We first observe that when no pretraining is used, DKM with annealing (DKM$^{\text{a}}$) markedly outperforms DCN$^{\text{np}}$ on all datasets. DKM$^{\text{a}}$ achieves clustering performance similar to that obtained by pretraining-based methods. This confirms our intuition that the proposed annealing strategy can be seen as an alternative to pretraining.

Among the approaches integrating representation learning with pretraining, the AE-KM method, that separately performs dimension reduction and k-Means clustering, overall obtains the worst results. This observation is in line with prior studies (Yang et al., 2017; Guo et al., 2017) and underlines again the importance of jointly learning representations and clustering. We note as well that, apart from DKM$^{\text{a}}$, pretraining-based deep clustering approaches substantially outperform their non-pretrained counterparts, which stresses the importance of pretraining.

Furthermore, DKM$^{\text{p}}$ yields significant improvements on all collections except RCV1 over DCN$^{\text{p}}$, the other "true" deep k-Means approach. In all cases, DCN$^{\text{p}}$ shows performance on par with that of AE-KM. This places, to the best of our knowledge, DKM$^{\text{p}}$ as the current best deep k-Means clustering method.

To further confirm DKM's efficacy, we also compare it against IDEC, a state-of-the-art deep clustering algorihm which is not based on k-Means. We report the corresponding results in Table 2. Once again, DKM$^{\text{a}}$ significantly outperforms its non-pretrained counterpart, IDEC$^{\text{np}}$, except on RCV1. We note as well that, with the exception of the NMI results on MNIST, DKM$^{\text{p}}$ is always either significantly better than IDEC$^{\text{p}}$ or with no significant difference from this latter. This shows that the proposed DKM is not only the strongest k-Means-related clustering approach, but is also remarkably competitive wrt deep clustering state of the art.



(a) AE (b) DCN$^{\text{p}}$ (c) DKM$^{\text{a}}$ (d) DKM$^{\text{p}}$

Fig. 2. t-SNE visualization of the embedding spaces learned on USPS.

4.5. Illustration of learned representations

While the quality of the clustering results and that of the representations learned by the models are likely to be correlated, it is relevant to study to what extent learned representations are distorted to facilitate clustering. To provide a more interpretable view of the representations learned by k-means-related deep clustering algorithm, we illustrate the embedded samples provided by AE (for comparison), DCN$^{\text{p}}$, DKM$^{\text{a}}$, and DKM$^{\text{p}}$ on USPS in Figure 2 (best viewed in color). DCN$^{\text{np}}$ was discarded due to its poor clustering performance. We used for that matter the t-SNE visualization method (van der Maaten and Hinton, 2008) to project the embeddings into a 2D space. We observe that the representations for points from different clusters are clearly better separated and disentangled in DKM$^{\text{p}}$ than in other models. This brings further support to our experimental results, which showed the superior ability of DKM$^{\text{p}}$ to learn representations that facilitate clustering.

5. Conclusion

We have presented in this paper a new approach for jointly clustering with k-Means and learning representations by considering the k-Means clustering loss as the limit of a differentiable function. If several studies have proposed solutions to this problem with different clustering losses, to the best of our knowledge, this is the first approach that truly jointly optimizes, through simple stochastic gradient descent updates, representation and k-Means clustering losses. In addition to pretraining, that can be used in all methods, this approach can also rely on a deterministic annealing scheme for parameter initialization.

We further conducted careful comparisons with previous approaches by ensuring that the same architecture, initialization and minibatches are used. The experiments conducted on several

Table 1. Clustering results of the k-Means-related methods. Performance is measured in terms of NMI and ACC (%); higher is better. Each cell contains the average and standard deviation computed over 10 runs. Bold (resp. underlined) values correspond to results with no significant difference ($p > 0.05$) to the best approach with (resp. without) pretraining for each dataset/metric pair.

Model	MNIST		USPS		20NEWS		RCV1	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
KM	53.5±0.3	49.8±0.5	67.3±0.1	61.4±0.1	23.2±1.5	21.6±1.8	50.8±2.9	**31.3±5.4**
AE-KM	80.8±1.8	75.2±1.1	72.9±0.8	71.7±1.2	**49.0±2.9**	44.5±1.5	**56.7±3.6**	**31.5±4.3**
Deep clustering approaches without pretraining								
DCN^{np}	34.8±3.0	18.1±1.0	36.4±3.5	16.9±1.3	17.9±1.0	9.8±0.5	41.3±4.0	6.9±1.8
DKM^{a}	82.3±3.2	78.0±1.9	**75.5±6.8**	73.0±2.3	44.8±2.4	42.8±1.1	**53.8±5.5**	**28.0±5.8**
Deep clustering approaches with pretraining								
DCN^{p}	81.1±1.9	75.7±1.1	73.0±0.8	71.9±1.2	**49.2±2.9**	44.7±1.5	**56.7±3.6**	**31.6±4.3**
DKM^{p}	**84.0±2.2**	**79.6±0.9**	**75.7±1.3**	**77.6±1.1**	**51.2±2.8**	**46.7±1.2**	**58.3±3.8**	**33.1±4.9**

Table 2. Clustering results of the DKM and IDEC methods. Performance is measured in terms of NMI and ACC (%); higher is better. Each cell contains the average and standard deviation computed over 10 runs. Bold (resp. underlined) values correspond to results with no significant difference ($p > 0.05$) to the best approach with (resp. without) pretraining for each dataset/metric pair.

Model	MNIST		USPS		20NEWS		RCV1	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Deep clustering approaches without pretraining								
IDEC^{np}	61.8±3.0	62.4±1.6	53.9±5.1	50.0±3.8	22.3±1.5	22.3±1.5	**56.7±5.3**	**31.4±2.8**
DKM^{a}	82.3±3.2	78.0±1.9	**75.5±6.8**	73.0±2.3	44.8±2.4	42.8±1.1	53.8±5.5	28.0±5.8
Deep clustering approaches with pretraining								
IDEC^{p}	**85.7±2.4**	**86.4±1.0**	**75.2±0.5**	74.9±0.6	40.5±1.3	38.2±1.0	**59.5±5.7**	**34.7±5.0**
DKM^{p}	**84.0±2.2**	79.6±0.9	**75.7±1.3**	**77.6±1.1**	**51.2±2.8**	**46.7±1.2**	**58.3±3.8**	**33.1±4.9**

datasets confirm the good behavior of Deep k-Means that outperforms DCN, the current best approach for k-Means clustering in embedding spaces, on all the collections considered.

References

Agustsson, E., Mentzer, F., Tschannen, M., Cavigelli, L., Timofte, R., Benini, L., Van Gool, L., 2017. Soft-to-Hard Vector Quantization for End-to-End Learning Compressible Representations, in: Proceedings of the 31st Annual Conference on Neural Information Processing Systems, pp. 1141–1151.

Aljalbout, E., Golkov, V., Siddiqui, Y., Cremers, D., 2018. Clustering with Deep Learning: Taxonomy and New Methods. arXiv:1801.07648 .

Arthur, D., Vassilvitskii, S., 2007. K-Means++: The Advantages of Careful Seeding, in: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1025.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2006. Greedy Layer-Wise Training of Deep Networks, in: Proceedings of the 20th Annual Conference on Neural Information Processing Systems, pp. 153–160.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer.

Cai, D., He, X., Han, J., 2011. Locally Consistent Concept Factorization for Document Clustering. IEEE Transactions on Knowledge and Data Engineering 23, 902–913.

Chang, J., Wang, L., Meng, G., Xiang, S., Pan, C., 2017. Deep Adaptive Image Clustering, in: Proceedings of the 2017 IEEE International Conference on Computer Vision, pp. 5879–5887.

Dilokthanakul, N., Mediano, P.A.M., Garnelo, M., Lee, M.C.H., Salimbeni, H., Arulkumaran, K., Shanahan, M., 2017. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. arXiv:1611.02648 .

Dizaji, K.G., Herandi, A., Deng, C., Cai, W., Huang, H., 2017. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization, in: Proceedings of the 2017 IEEE International Conference on Computer Vision, pp. 5736–5745.

Glorot, X., Bengio, Y., 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks, in: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics.

Guo, X., Gao, L., Liu, X., Yin, J., 2017. Improved Deep Embedded Clustering with Local Structure Preservation, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1753–1759.

Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the Dimensionality of Data with Neural Networks. Science 313, 504–507.

Hsu, C.C., Lin, C.W., 2018. CNN-Based Joint Clustering and Representation Learning with Feature Drift Compensation for Large-Scale Image Data. IEEE Transactions on Multimedia 20, 421–429.

Hu, W., Miyato, T., Tokui, S., Matsumoto, E., Sugiyama, M., 2017. Learning Discrete Representations via Information Maximizing Self-Augmented Training, in: Proceedings of the 34th International Conference on Machine Learning, pp. 1558–1567.

Huang, P., Huang, Y., Wang, W., Wang, L., 2014. Deep Embedding Network for Clustering, in: Proceedings of the 22nd International Conference on Pattern Recognition, pp. 1532–1537.

Jang, E., Gu, S., Poole, B., 2017. Categorical Reparameterization with Gumbel-Softmax, in: Proceedings of the 5th International Conference on Learning Representations.

Ji, P., Zhang, T., Li, H., Salzmann, M., Reid, I., 2017. Deep Subspace Clustering Networks, in: Proceedings of the 31st Annual Conference on Neural Information Processing Systems, pp. 23–32.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., Zhou, H., 2017. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1965–1972.

Kingma, D.P., Ba, J.L., 2015. Adam: a Method for Stochastic Optimization, in: Proceedings of the 3rd International Conference on Learning Representations.

Kingma, D.P., Welling, M., 2014. Auto-Encoding Variational Bayes, in: Proceedings of the 2nd International Conference on Learning Representations.

van der Maaten, L., Hinton, G., 2008. Visualizing Data using t-SNE. Journal of Machine Learning Research 9, 2579–2605.

MacQueen, J., 1967. Some Methods for Classification and Analysis of Mul-

tivariate Observations, in: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297.

Maddison, C.J., Mnih, A., Teh, Y.W., 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables, in: Proceedings of the 5th International Conference on Learning Representations.

Nair, V., Hinton, G.E., 2010. Rectified Linear Units Improve Restricted Boltzmann Machines, in: Proceedings of the 27th International Conference on Machine Learning, pp. 807–814.

Peng, X., Feng, J., Lu, J., Yau, W.y., Yi, Z., 2017. Cascade Subspace Clustering, in: Proceedings of the 31th Conference on Artificial Intelligence, pp. 2478–2484.

Peng, X., Xiao, S., Feng, J., Yau, W.Y., Yi, Z., 2016. Deep Subspace Clustering with Sparsity Prior, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, pp. 1925–1931.

Rose, K., Gurewitz, E., Fox, G., 1990. A Deterministic Annealing Approach to Clustering. Pattern Recognition Letters 11, 589–594.

Xie, J., Girshick, R., Farhadi, A., 2016. Unsupervised Deep Embedding for Clustering Analysis, in: Proceedings of the 33rd International Conference on Machine Learning, pp. 478–487.

Yang, B., Fu, X., Sidiropoulos, N.D., Hong, M., 2017. Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering, in: Proceedings of the 34th International Conference on Machine Learning, pp. 3861–3870.

Yang, J., Parikh, D., Batra, D., 2016. Joint Unsupervised Learning of Deep Representations and Image Clusters, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, pp. 5147–5156.