Unsupervised Deep Embedding for Clustering Analysis

Junyuan Xie JXIE@CS.WASHINGTON.EDU
University of Washington

Ross Girshick RBG@FB.COM
Facebook AI Research (FAIR)

Ali Farhadi ALI@CS.WASHINGTON.EDU
University of Washington

Abstract

Clustering is central to many data-driven application domains and has been studied extensively in terms of distance functions and grouping algorithms. Relatively little work has focused on learning representations for clustering. In this paper, we propose Deep Embedded Clustering (DEC), a method that simultaneously learns feature representations and cluster assignments using deep neural networks. DEC learns a mapping from the data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective. Our experimental evaluations on image and text corpora show significant improvement over state-of-the-art methods.

1. Introduction

Clustering, an essential data analysis and visualization tool, has been studied extensively in unsupervised machine learning from different perspectives: What defines a cluster? What is the right distance metric? How to efficiently group instances into clusters? How to validate clusters? And so on. Numerous different distance functions and embedding methods have been explored in the literature. Relatively little work has focused on the unsupervised learning of the feature space in which to perform clustering.

A notion of *distance* or *dissimilarity* is central to data clustering algorithms. Distance, in turn, relies on representing the data in a feature space. The k-means clustering algorithm (MacQueen et al., 1967), for example, uses the Euclidean distance between points in a given feature space, which for images might be raw pixels or gradient-orientation histograms. The choice of feature space is customarily left as an application-specific detail for the end-user to determine. Yet it is clear that the choice of feature space is crucial; for all but the simplest image datasets, clustering with Euclidean distance on raw pixels is completely ineffective. In this paper, we revisit cluster analysis and ask: *Can we use a data driven approach to solve for the feature space and cluster memberships jointly?*

We take inspiration from recent work on deep learning for computer vision (Krizhevsky et al., 2012; Girshick et al., 2014; Zeiler & Fergus, 2014; Long et al., 2014), where clear gains on benchmark tasks have resulted from learning better features. These improvements, however, were obtained with *supervised* learning, whereas our goal is *unsupervised* data clustering. To this end, we define a parameterized non-linear mapping from the data space X to a lower-dimensional feature space Z, where we optimize a clustering objective. Unlike previous work, which operates on the data space or a shallow linear embedded space, we use stochastic gradient descent (SGD) via backpropagation on a clustering objective to learn the mapping, which is parameterized by a deep neural network. We refer to this clustering algorithm as *Deep Embedded Clustering*, or DEC.

Optimizing DEC is challenging. We want to simultaneously solve for cluster assignment and the underlying feature representation. However, unlike in supervised learning, we cannot train our deep network with labeled data. Instead we propose to iteratively refine clusters with an auxiliary target distribution derived from the current soft cluster assignment. This process gradually improves the clustering as well as the feature representation.

Our experiments show significant improvements over state-of-the-art clustering methods in terms of both accuracy and running time on image and textual datasets. We evaluate DEC on MNIST (LeCun et al., 1998), STL (Coates et al.,

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

2011), and REUTERS (Lewis et al., 2004), comparing it with standard and state-of-the-art clustering methods (Nie et al., 2011; Yang et al., 2010). In addition, our experiments show that DEC is significantly less sensitive to the choice of hyperparameters compared to state-of-the-art methods. This robustness is an important property of our clustering algorithm since, when applied to real data, supervision is not available for hyperparameter cross-validation.

Our contributions are: (a) joint optimization of deep embedding and clustering; (b) a novel iterative refinement via soft assignment; and (c) state-of-the-art clustering results in terms of clustering accuracy and speed. Our Caffe-based (Jia et al., 2014) implementation of DEC is available at `https://github.com/piiswrong/dec`.

2. Related work

Clustering has been extensively studied in machine learning in terms of feature selection (Boutsidis et al., 2009; Liu & Yu, 2005; Alelyani et al., 2013), distance functions (Xing et al., 2002; Xiang et al., 2008), grouping methods (MacQueen et al., 1967; Von Luxburg, 2007; Li et al., 2004), and cluster validation (Halkidi et al., 2001). Space does not allow for a comprehensive literature study and we refer readers to (Aggarwal & Reddy, 2013) for a survey.

One branch of popular methods for clustering is k-means (MacQueen et al., 1967) and Gaussian Mixture Models (GMM) (Bishop, 2006). These methods are fast and applicable to a wide range of problems. However, their distance metrics are limited to the original data space and they tend to be ineffective when input dimensionality is high (Steinbach et al., 2004).

Several variants of k-means have been proposed to address issues with higher-dimensional input spaces. De la Torre & Kanade (2006); Ye et al. (2008) perform joint dimensionality reduction and clustering by first clustering the data with k-means and then projecting the data into a lower dimensions where the inter-cluster variance is maximized. This process is repeated in EM-style iterations until convergence. However, this framework is limited to linear embedding; our method employs deep neural networks to perform non-linear embedding that is necessary for more complex data.

Spectral clustering and its variants have gained popularity recently (Von Luxburg, 2007). They allow more flexible distance metrics and generally perform better than k-means. Combining spectral clustering and embedding has been explored in Yang et al. (2010); Nie et al. (2011). Tian et al. (2014) proposes an algorithm based on spectral clustering, but replaces eigenvalue decomposition with deep autoencoder, which improves performance but further increases memory consumption.

Most spectral clustering algorithms need to compute the full graph Laplacian matrix and therefore have quadratic or super quadratic complexities in the number of data points. This means they need specialized machines with large memory for any dataset larger than a few tens of thousands of points. In order to scale spectral clustering to large datasets, approximate algorithms were invented to trade off performance for speed (Yan et al., 2009). Our method, however, is linear in the number of data points and scales gracefully to large datasets.

Minimizing the Kullback-Leibler (KL) divergence between a data distribution and an embedded distribution has been used for data visualization and dimensionality reduction (van der Maaten & Hinton, 2008). T-SNE, for instance, is a non-parametric algorithm in this school and a parametric variant of t-SNE (van der Maaten, 2009) uses deep neural network to parametrize the embedding. The complexity of t-SNE is $O(n^2)$, where n is the number of data points, but it can be approximated in $O(n \log n)$ (van Der Maaten, 2014).

We take inspiration from parametric t-SNE. Instead of minimizing KL divergence to produce an embedding that is faithful to distances in the original data space, we define a centroid-based probability distribution and minimize its KL divergence to an auxiliary target distribution to simultaneously improve clustering assignment and feature representation. A centroid-based method also has the benefit of reducing complexity to $O(nk)$, where k is the number of centroids.

3. Deep embedded clustering

Consider the problem of clustering a set of n points $\{x_i \in X\}_{i=1}^n$ into k clusters, each represented by a centroid $\mu_j, j = 1, \ldots, k$. Instead of clustering directly in the *data space* X, we propose to first transform the data with a non-linear mapping $f_\theta : X \to Z$, where θ are learnable parameters and Z is the latent *feature space*. The dimensionality of Z is typically much smaller than X in order to avoid the "curse of dimensionality" (Bellman, 1961). To parametrize f_θ, deep neural networks (DNNs) are a natural choice due to their theoretical function approximation properties (Hornik, 1991) and their demonstrated feature learning capabilities (Bengio et al., 2013).

The proposed algorithm (DEC) clusters data by *simultaneously* learning a set of k cluster centers $\{\mu_j \in Z\}_{j=1}^k$ in the feature space Z and the parameters θ of the DNN that maps data points into Z. DEC has two phases: (1) parameter initialization with a deep autoencoder (Vincent et al., 2010) and (2) parameter optimization (i.e., clustering), where we iterate between computing an auxiliary target distribution and minimizing the Kullback–Leibler (KL) divergence to

it. We start by describing phase (2) parameter optimization/clustering, given an initial estimate of θ and $\{\mu_j\}_{j=1}^k$.

3.1. Clustering with KL divergence

Given an initial estimate of the non-linear mapping f_θ and the initial cluster centroids $\{\mu_j\}_{j=1}^k$, we propose to improve the clustering using an unsupervised algorithm that alternates between two steps. In the first step, we compute a soft assignment between the embedded points and the cluster centroids. In the second step, we update the deep mapping f_θ and refine the cluster centroids by learning from current high confidence assignments using an auxiliary target distribution. This process is repeated until a convergence criterion is met.

3.1.1. Soft Assignment

Following van der Maaten & Hinton (2008) we use the Student's t-distribution as a kernel to measure the similarity between embedded point z_i and centroid μ_j:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|z_i - \mu_{j'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \tag{1}$$

where $z_i = f_\theta(x_i) \in Z$ corresponds to $x_i \in X$ after embedding, α are the degrees of freedom of the Student's t-distribution and q_{ij} can be interpreted as the probability of assigning sample i to cluster j (i.e., a soft assignment). Since we cannot cross-validate α on a validation set in the unsupervised setting, and learning it is superfluous (van der Maaten, 2009), we let $\alpha = 1$ for all experiments.

3.1.2. KL divergence minimization

We propose to iteratively refine the clusters by learning from their high confidence assignments with the help of an auxiliary target distribution. Specifically, our model is trained by matching the soft assignment to the target distribution. To this end, we define our objective as a KL divergence loss between the soft assignments q_i and the auxiliary distribution p_i as follows:

$$L = \mathrm{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{2}$$

The choice of target distributions P is crucial for DEC's performance. A naive approach would be setting each p_i to a delta distribution (to the nearest centroid) for data points above a confidence threshold and ignore the rest. However, because q_i are soft assignments, it is more natural and flexible to use softer probabilistic targets. Specifically, we would like our target distribution to have the following properties: (1) strengthen predictions (i.e., improve cluster purity), (2) put more emphasis on data points assigned with high confidence, and (3) normalize loss contribution of each centroid to prevent large clusters from distorting the hidden feature space.

In our experiments, we compute p_i by first raising q_i to the second power and then normalizing by frequency per cluster:

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \tag{3}$$

where $f_j = \sum_i q_{ij}$ are soft cluster frequencies. Please refer to section 5.1 for discussions on empirical properties of L and P.

Our training strategy can be seen as a form of self-training (Nigam & Ghani, 2000). As in self-training, we take an initial classifier and an unlabeled dataset, then label the dataset with the classifier in order to train on its own high confidence predictions. Indeed, in experiments we observe that DEC improves upon the initial estimate in each iteration by learning from high confidence predictions, which in turn helps to improve low confidence ones.

3.1.3. Optimization

We jointly optimize the cluster centers $\{\mu_j\}$ and DNN parameters θ using Stochastic Gradient Descent (SGD) with momentum. The gradients of L with respect to feature-space embedding of each data point z_i and each cluster centroid μ_j are computed as:

$$\begin{aligned}\frac{\partial L}{\partial z_i} &= \frac{\alpha+1}{\alpha}\sum_j (1 + \frac{\|z_i - \mu_j\|^2}{\alpha})^{-1} \\ &\quad \times (p_{ij} - q_{ij})(z_i - \mu_j),\end{aligned} \tag{4}$$

$$\begin{aligned}\frac{\partial L}{\partial \mu_j} &= -\frac{\alpha+1}{\alpha}\sum_i (1 + \frac{\|z_i - \mu_j\|^2}{\alpha})^{-1} \\ &\quad \times (p_{ij} - q_{ij})(z_i - \mu_j).\end{aligned} \tag{5}$$

The gradients $\partial L/\partial z_i$ are then passed down to the DNN and used in standard backpropagation to compute the DNN's parameter gradient $\partial L/\partial \theta$. For the purpose of discovering cluster assignments, we stop our procedure when less than $tol\%$ of points change cluster assignment between two consecutive iterations.

3.2. Parameter initialization

Thus far we have discussed how DEC proceeds given initial estimates of the DNN parameters θ and the cluster centroids $\{\mu_j\}$. Now we discuss how the parameters and centroids are initialized.

We initialize DEC with a stacked autoencoder (SAE) because recent research has shown that they consistently produce semantically meaningful and well-separated representations on real-world datasets (Vincent et al., 2010; Hinton & Salakhutdinov, 2006; Le, 2013). Thus the unsuper-

Table 1. Dataset statistics.

Dataset	# Points	# classes	Dimension	% of largest class
MNIST (LeCun et al., 1998)	70000	10	784	11%
STL-10 (Coates et al., 2011)	13000	10	1428	10%
REUTERS-10K	10000	4	2000	43%
REUTERS (Lewis et al., 2004)	685071	4	2000	43%

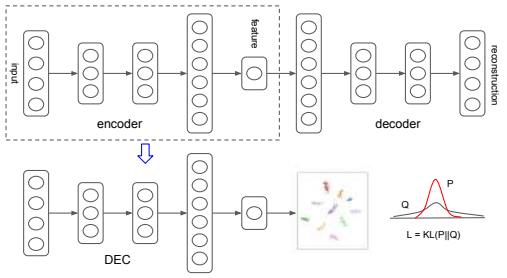


Figure 1. Network structure

vised representation learned by SAE naturally facilitates the learning of clustering representations with DEC.

We initialize the SAE network layer by layer with each layer being a denoising autoencoder trained to reconstruct the previous layer's output after random corruption (Vincent et al., 2010). A denoising autoencoder is a two layer neural network defined as:

$$\tilde{x} \sim Dropout(x) \tag{6}$$

$$h = g_1(W_1\tilde{x} + b_1) \tag{7}$$

$$\tilde{h} \sim Dropout(h) \tag{8}$$

$$y = g_2(W_2\tilde{h} + b_2) \tag{9}$$

where $Dropout(\cdot)$ (Srivastava et al., 2014) is a stochastic mapping that randomly sets a portion of its input dimensions to 0, g_1 and g_2 are activation functions for encoding and decoding layer respectively, and $\theta = \{W_1, b_1, W_2, b_2\}$ are model parameters. Training is performed by minimizing the least-squares loss $\|x - y\|_2^2$. After training of one layer, we use its output h as the input to train the next layer. We use rectified linear units (ReLUs) (Nair & Hinton, 2010) in all encoder/decoder pairs, except for g_2 of the *first* pair (it needs to reconstruct input data that may have positive and negative values, such as zero-mean images) and g_1 of the *last* pair (so the final data embedding retains full information (Vincent et al., 2010)).

After greedy layer-wise training, we concatenate all encoder layers followed by all decoder layers, in reverse layer-wise training order, to form a deep autoencoder and then finetune it to minimize reconstruction loss. The final result is a multilayer deep autoencoder with a bottleneck coding layer in the middle. We then discard the decoder layers and use the encoder layers as our initial mapping between the data space and the feature space, as shown in Fig. 1.

To initialize the cluster centers, we pass the data through the initialized DNN to get embedded data points and then perform standard k-means clustering in the feature space Z to obtain k initial centroids $\{\mu_j\}_{j=1}^k$.

4. Experiments

4.1. Datasets

We evaluate the proposed method (DEC) on one text dataset and two image datasets and compare it against other algorithms including k-means, LDGMI (Yang et al., 2010), and SEC (Nie et al., 2011). LDGMI and SEC are spectral clustering based algorithms that use a Laplacian matrix and various transformations to improve clustering performance. Empirical evidence reported in Yang et al. (2010); Nie et al. (2011) shows that LDMGI and SEC outperform traditional spectral clustering methods on a wide range of datasets. We show qualitative and quantitative results that demonstrate the benefit of DEC compared to LDGMI and SEC.

In order to study the performance and generality of different algorithms, we perform experiment on two image datasets and one text data set:

- **MNIST**: The MNIST dataset consists of 70000 handwritten digits of 28-by-28 pixel size. The digits are centered and size-normalized (LeCun et al., 1998).
- **STL-10**: A dataset of 96-by-96 color images. There are 10 classes with 1300 examples each. It also contains 100000 unlabeled images of the same resolution (Coates et al., 2011). We also used the unlabeled set when training our autoencoders. Similar to Doersch et al. (2012), we concatenated HOG feature and a 8-by-8 color map to use as input to all algorithms.
- **REUTERS**: Reuters contains about 810000 English news stories labeled with a category tree (Lewis et al., 2004). We used the four root categories: corporate/industrial, government/social, markets, and economics as labels and further pruned all documents that are labeled by multiple root categories to get 685071

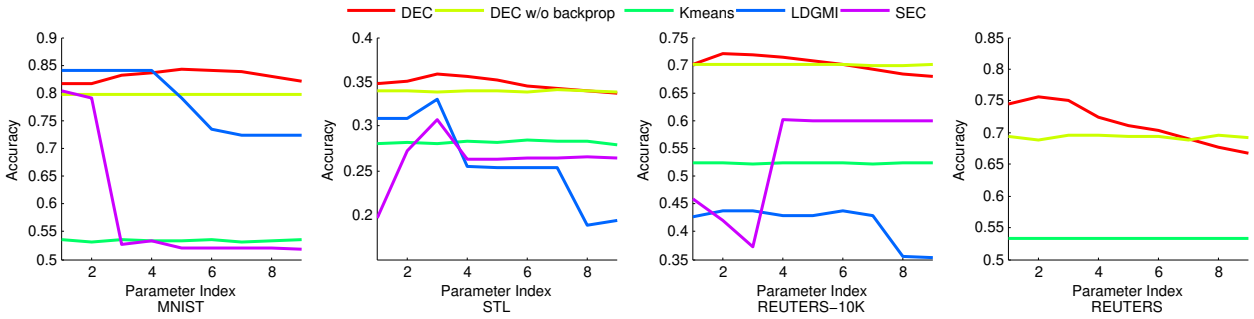


Figure 2. Clustering accuracy for different hyperparameter choices for each algorithm. DEC outperforms other methods and is more robust to hyperparameter changes compared to either LDGMI or SEC. Robustness is important because cross-validation is not possible in real-world applications of cluster analysis. This figure is best viewed in color.

Table 2. Comparison of clustering accuracy (Eq. 10) on four datasets.

Method	MNIST	STL-HOG	REUTERS-10k	REUTERS
k-means	53.49%	28.39%	52.42%	53.29%
LDMGI	84.09%	33.08%	43.84%	N/A
SEC	80.37%	30.75%	60.08%	N/A
DEC w/o backprop	79.82%	34.06%	70.05%	69.62%
DEC (ours)	**84.30%**	**35.90%**	**72.17%**	**75.63%**

articles. We then computed tf-idf features on the 2000 most frequently occurring word stems. Since some algorithms do not scale to the full Reuters dataset, we also sampled a random subset of 10000 examples, which we call REUTERS-10k, for comparison purposes.

A summary of dataset statistics is shown in Table 1. For all algorithms, we normalize all datasets so that $\frac{1}{d}\|x_i\|_2^2$ is approximately 1, where d is the dimensionality of the data space point $x_i \in X$.

4.2. Evaluation Metric

We use the standard unsupervised evaluation metric and protocols for evaluations and comparisons to other algorithms (Yang et al., 2010). For all algorithms we set the number of clusters to the number of ground-truth categories and evaluate performance with *unsupervised clustering accuracy (ACC)*:

$$ACC = \max_m \frac{\sum_{i=1}^n \mathbf{1}\{l_i = m(c_i)\}}{n}, \qquad (10)$$

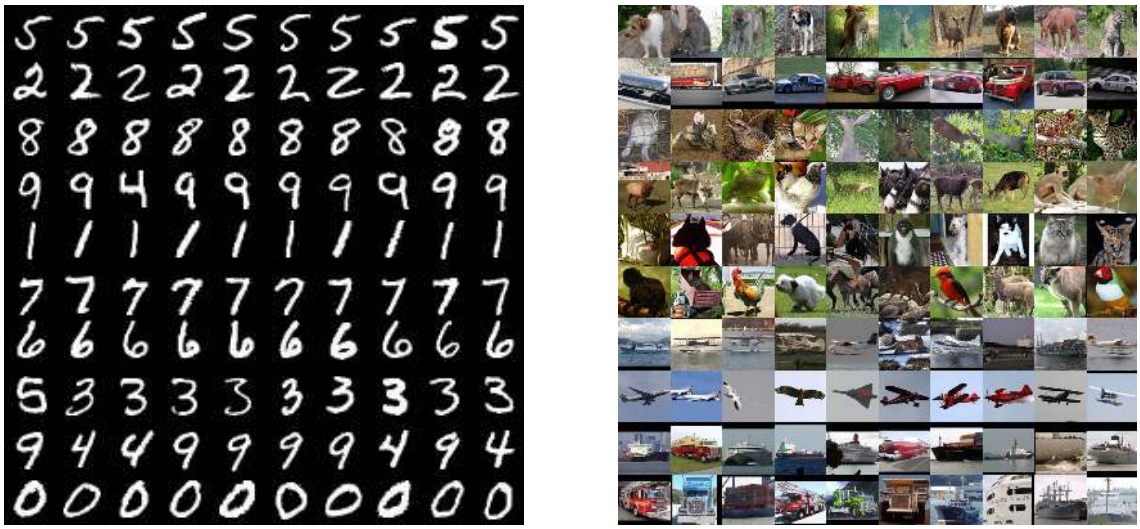
where l_i is the ground-truth label, c_i is the cluster assignment produced by the algorithm, and m ranges over all possible one-to-one mappings between clusters and labels.

Intuitively this metric takes a cluster assignment from an *unsupervised* algorithm and a ground truth assignment and then finds the best matching between them. The best mapping can be efficiently computed by the Hungarian algorithm (Kuhn, 1955).

4.3. Implementation

Determining hyperparameters by cross-validation on a validation set is not an option in unsupervised clustering. Thus we use commonly used parameters for DNNs and avoid dataset specific tuning as much as possible. Specifically, inspired by van der Maaten (2009), we set network dimensions to d–500–500–2000–10 for all datasets, where d is the data-space dimension, which varies between datasets. All layers are densely (fully) connected.

During greedy layer-wise pretraining we initialize the weights to random numbers drawn from a zero-mean Gaussian distribution with a standard deviation of 0.01. Each layer is pretrained for 50000 iterations with a dropout rate of 20%. The entire deep autoencoder is further finetuned for 100000 iterations without dropout. For both layer-wise pretraining and end-to-end finetuning of the autoencoder the minibatch size is set to 256, starting learning rate is set to 0.1, which is divided by 10 every 20000 iterations, and weight decay is set to 0. All of the above parameters are set to achieve a reasonably good reconstruction loss and are held constant across all datasets. Dataset-specific settings of these parameters might improve performance on each dataset, but we refrain from this type of unrealistic parameter tuning. To initialize centroids, we run k-means with 20 restarts and select the best solution.



(a) MNIST (b) STL-10

Figure 3. Each row contains the top 10 scoring elements from one cluster.

In the KL divergence minimization phase, we train with a constant learning rate of 0.01. The convergence threshold is set to $tol = 0.1\%$. Our implementation is based on Python and Caffe (Jia et al., 2014) and is available at `https://github.com/piiswrong/dec`.

For all baseline algorithms, we perform 20 random restarts when initializing centroids and pick the result with the best objective value. For a fair comparison with previous work (Yang et al., 2010), we vary one hyperparameter for each algorithm over 9 possible choices and report the best accuracy in Table 2 and the range of accuracies in Fig. 2. For LDGMI and SEC, we use the same parameter and range as in their corresponding papers. For our proposed algorithm, we vary λ, the parameter that controls annealing speed, over $2^i \times 10, i = 0, 1, ..., 8$. Since k-means does not have tunable hyperparameters (aside from k), we simply run them 9 times. GMMs perform similarly to k-means so we only report k-means results. Traditional spectral clustering performs worse than LDGMI and SEC so we only report the latter (Yang et al., 2010; Nie et al., 2011).

4.4. Experiment results

We evaluate the performance of our algorithm both quantitatively and qualitatively. In Table 2, we report the best performance, over 9 hyperparameter settings, of each algorithm. Note that DEC outperforms all other methods, sometimes with a significant margin. To demonstrate the effectiveness of end-to-end training, we also show the results from freezing the non-linear mapping f_θ during clustering. We find that this ablation ("DEC w/o backprop") generally performs worse than DEC.

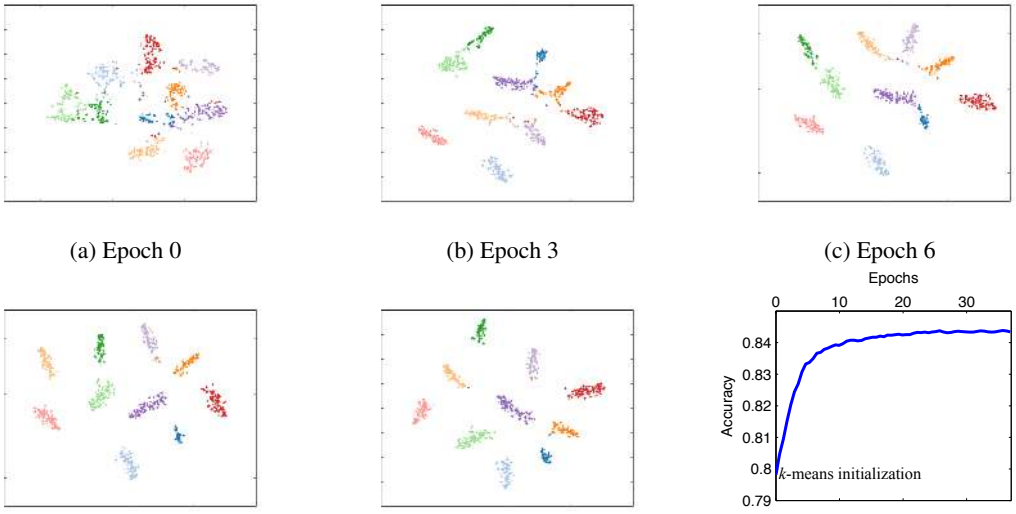
In order to investigate the effect of hyperparameters, we plot the accuracy of each method under all 9 settings (Fig. 2). We observe that DEC is more consistent across hyperparameter ranges compared to LDGMI and SEC. For DEC, hyperparameter $\lambda = 40$ gives near optimal performance on all dataset, whereas for other algorithms the optimal hyperparameter varies widely. Moreover, DEC can process the entire REUTERS dataset in half an hour with GPU acceleration while the second best algorithms, LDGMI and SEC, would need months of computation time and terabytes of memory. We, indeed, could not run these methods on the full REUTERS dataset and report N/A in Table 2 (GPU adaptation of these methods is non-trivial).

In Fig. 3 we show 10 top scoring images from each cluster in MNIST and STL. Each row corresponds to a cluster and images are sorted from left to right based on their distance to the cluster center. We observe that for MNIST, DEC's cluster assignment corresponds to natural clusters very well, with the exception of confusing 4 and 9, while for STL, DEC is mostly correct with airplanes, trucks and cars, but spends part of its attention on poses instead of categories when it comes to animal classes.

5. Discussion

5.1. Assumptions and Objective

The underlying assumption of DEC is that the initial classifier's high confidence predictions are mostly correct. To verify that this assumption holds for our task and that our choice of P has the desired properties, we plot the magnitude of the gradient of L with respect to each embedded point, $|\partial L/\partial z_i|$, against its soft assignment, q_{ij}, to a ran-



(a) Epoch 0 (b) Epoch 3 (c) Epoch 6

(d) Epoch 9 (e) Epoch 12 (f) Accuracy vs. epochs

Figure 5. We visualize the latent representation as the KL divergence minimization phase proceeds on MNIST. Note the separation of clusters from epoch 0 to epoch 12. We also plot the accuracy of DEC at different epochs, showing that KL divergence minimization improves clustering accuracy. This figure is best viewed in color.

Table 3. Comparison of clustering accuracy (Eq. 10) on autoencoder (AE) feature.

Method	MNIST	STL-HOG	REUTERS-10k	REUTERS
AE+k-means	81.84%	33.92%	66.59%	71.97%
AE+LDMGI	83.98%	32.04%	42.92%	N/A
AE+SEC	81.56%	32.29%	61.86%	N/A
DEC (ours)	**84.30%**	**35.90%**	**72.17%**	**75.63%**

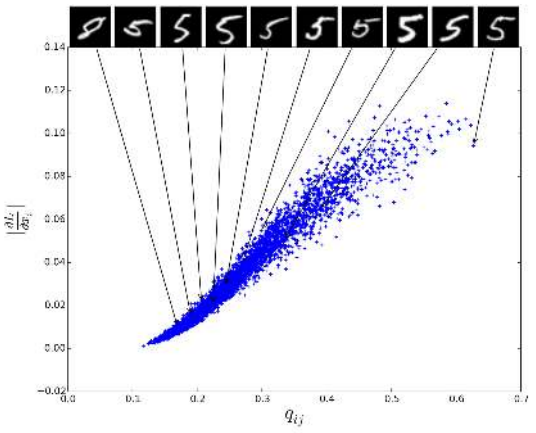


Figure 4. Gradient visualization at the start of KL divergence minimization. This plot shows the magnitude of the gradient of the loss L vs. the cluster soft assignment probability q_{ij}. See text for discussion.

domly chosen MNIST cluster j (Fig. 4).

We observe points that are closer to the cluster center (large q_{ij}) contribute more to the gradient. We also show the raw images of 10 data points at each 10 percentile sorted by q_{ij}. Instances with higher similarity are more canonical examples of "5". As confidence decreases, instances become more ambiguous and eventually turn into a mislabeled "8" suggesting the soundness of our assumptions.

5.2. Contribution of Iterative Optimization

In Fig. 5 we visualize the progression of the embedded representation of a random subset of MNIST during training. For visualization we use t-SNE (van der Maaten & Hinton, 2008) applied to the embedded points z_i. It is clear that the clusters are becoming increasingly well separated. Fig. 5 (f) shows how accuracy correspondingly improves over SGD epochs.

5.3. Contribution of Autoencoder Initialization

To better understand the contribution of each component, we show the performance of all algorithms with autoencoder features in Table 3. We observe that SEC and LDMGI's performance do not change significantly with au-

Table 4. Clustering accuracy (Eq. 10) on imbalanced subsample of MNIST.

Method \ r_{min}	0.1	0.3	0.5	0.7	0.9
k-means	47.14%	49.93%	53.65%	54.16%	54.39%
AE+k-means	66.82%	74.91%	77.93%	80.04%	81.31%
DEC	70.10%	80.92%	82.68%	84.69%	85.41%

toencoder feature, while k-means improved but is still below DEC. This demonstrates the power of deep embedding and the benefit of fine-tuning with the proposed KL divergence objective.

5.4. Performance on Imbalanced Data

In order to study the effect of imbalanced data, we sample subsets of MNIST with various retention rates. For minimum retention rate r_{min}, data points of class 0 will be kept with probability r_{min} and class 9 with probability 1, with the other classes linearly in between. As a result the largest cluster will be $1/r_{min}$ times as large as the smallest one. From Table 4 we can see that DEC is fairly robust against cluster size variation. We also observe that KL divergence minimization (DEC) consistently improves clustering accuracy after autoencoder and k-means initialization (shown as AE+k-means).

5.5. Number of Clusters

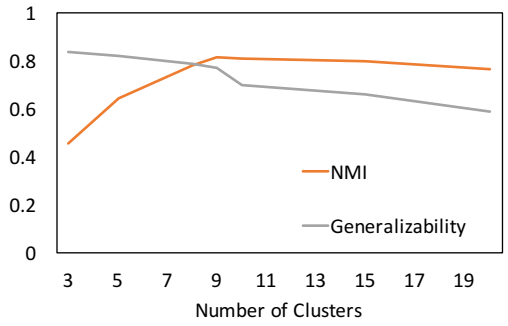


Figure 6. Selection of the centroid count, k. This is a plot of Normalized Mutual Information (NMI) and Generalizability vs. number of clusters. Note that there is a sharp drop of generalizability from 9 to 10 which means that 9 is the optimal number of clusters. Indeed, we observe that 9 gives the highest NMI.

So far we have assumed that the number of natural clusters is given to simplify comparison between algorithms. However, in practice this quantity is often unknown. Therefore a method for determining the optimal number of clusters is needed. To this end, we define two metrics: (1) the standard metric, Normalized Mutual Information (NMI), for evaluating clustering results with different cluster number:

$$NMI(l,c) = \frac{I(l,c)}{\frac{1}{2}[H(l)+H(c)]},$$

where I is the mutual information metric and H is entropy, and (2) generalizability (G) which is defined as the ratio between training and validation loss:

$$G = \frac{L_{train}}{L_{validation}}.$$

G is small when training loss is lower than validation loss, which indicate a high degree of overfitting.

Fig. 6 shows a sharp drop in generalizability when cluster number increases from 9 to 10, which suggests that 9 is the optimal number of clusters. We indeed observe the highest NMI score at 9, which demonstrates that generalizability is a good metric for selecting cluster number. NMI is highest at 9 instead 10 because 9 and 4 are similar in writing and DEC thinks that they should form a single cluster. This corresponds well with our qualitative results in Fig. 3.

6. Conclusion

This paper presents Deep Embedded Clustering, or DEC—an algorithm that clusters a set of data points in a jointly optimized feature space. DEC works by iteratively optimizing a KL divergence based clustering objective with a self-training target distribution. Our method can be viewed as an unsupervised extension of semisupervised self-training. Our framework provide a way to learn a representation specialized for clustering without groundtruth cluster membership labels.

Empirical studies demonstrate the strength of our proposed algorithm. DEC offers improved performance as well as robustness with respect to hyperparameter settings, which is particularly important in unsupervised tasks since cross-validation is not possible. DEC also has the virtue of linear complexity in the number of data points which allows it to scale to large datasets.

7. Acknowledgment

This work is in part supported by ONR N00014-13-1-0720, NSF IIS- 1338054, and Allen Distinguished Investigator Award.

References

Aggarwal, Charu C and Reddy, Chandan K. *Data clustering: algorithms and applications*. CRC Press, 2013.

Alelyani, Salem, Tang, Jiliang, and Liu, Huan. Feature selection for clustering: A review. *Data Clustering: Algorithms and Applications*, 2013.

Bellman, R. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, New Jersey, 1961.

Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Representation learning: A review and new perspectives. 2013.

Bishop, Christopher M. *Pattern recognition and machine learning*. springer New York, 2006.

Boutsidis, Christos, Drineas, Petros, and Mahoney, Michael W. Unsupervised feature selection for the k-means clustering problem. In *NIPS*, 2009.

Coates, Adam, Ng, Andrew Y, and Lee, Honglak. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 215–223, 2011.

De la Torre, Fernando and Kanade, Takeo. Discriminative cluster analysis. In *ICML*, 2006.

Doersch, Carl, Singh, Saurabh, Gupta, Abhinav, Sivic, Josef, and Efros, Alexei. What makes paris look like paris? *ACM Transactions on Graphics*, 2012.

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Halkidi, Maria, Batistakis, Yannis, and Vazirgiannis, Michalis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 2001.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Hornik, Kurt. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

Kuhn, Harold W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Le, Quoc V. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8595–8598. IEEE, 2013.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lewis, David D, Yang, Yiming, Rose, Tony G, and Li, Fan. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 2004.

Li, Tao, Ma, Sheng, and Ogihara, Mitsunori. Entropy-based criterion in categorical clustering. In *ICML*, 2004.

Liu, Huan and Yu, Lei. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2005.

Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. *arXiv preprint arXiv:1411.4038*, 2014.

MacQueen, James et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pp. 281–297, 1967.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Nie, Feiping, Zeng, Zinan, Tsang, Ivor W, Xu, Dong, and Zhang, Changshui. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 2011.

Nigam, Kamal and Ghani, Rayid. Analyzing the effectiveness and applicability of co-training. In *Proc. of the ninth international conference on Information and knowledge management*, 2000.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Steinbach, Michael, Ertöz, Levent, and Kumar, Vipin. The challenges of clustering high dimensional data. In *New Directions in Statistical Physics*, pp. 273–309. Springer, 2004.

Tian, Fei, Gao, Bin, Cui, Qing, Chen, Enhong, and Liu, Tie-Yan. Learning deep representations for graph clustering. In *AAAI Conference on Artificial Intelligence*, 2014.

van der Maaten, Laurens. Learning a parametric embedding by preserving local structure. In *International Conference on Artificial Intelligence and Statistics*, 2009.

van Der Maaten, Laurens. Accelerating t-SNE using tree-based algorithms. *JMLR*, 2014.

van der Maaten, Laurens and Hinton, Geoffrey. Visualizing data using t-SNE. *JMLR*, 2008.

Vincent, Pascal, Larochelle, Hugo, Lajoie, Isabelle, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 2010.

Von Luxburg, Ulrike. A tutorial on spectral clustering. *Statistics and computing*, 2007.

Xiang, Shiming, Nie, Feiping, and Zhang, Changshui. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 2008.

Xing, Eric P, Jordan, Michael I, Russell, Stuart, and Ng, Andrew Y. Distance metric learning with application to clustering with side-information. In *NIPS*, 2002.

Yan, Donghui, Huang, Ling, and Jordan, Michael I. Fast approximate spectral clustering. In *ACM SIGKDD*, 2009.

Yang, Yi, Xu, Dong, Nie, Feiping, Yan, Shuicheng, and Zhuang, Yueting. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 2010.

Ye, Jieping, Zhao, Zheng, and Wu, Mingrui. Discriminative k-means for clustering. In *NIPS*, 2008.

Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional networks. In *ECCV*. 2014.