

Denoising Autoencoder as an Effective Dimensionality Reduction and Clustering of Text Data

Milad Leyli-Abadi^(✉), Lazhar Labiod, and Mohamed Nadif

LIPADE, Paris Descartes University, 75006 Paris, France
mleyliabadi@gmail.com, {lazhar.labiod,mohamed.nadif}@parisdescartes.fr

Abstract. Deep learning methods are widely used in vision and face recognition, however there is a real lack of application of such methods in the field of text data. In this context, the data is often represented by a sparse high dimensional document-term matrix. Dealing with such data matrices, we present, in this paper, a new denoising auto-encoder for dimensionality reduction, where each document is not only affected by its own information, but also affected by the information from its neighbors according to the cosine similarity measure. It turns out that the proposed auto-encoder can discover the low dimensional embeddings, and as a result reveal the underlying effective manifold structure. The visual representation of these embeddings suggests the suitability of performing the clustering on the set of documents relying on the Expectation-Maximization algorithm for Gaussian mixture models. On real-world datasets, the relevance of the presented auto-encoder in the visualisation and document clustering field is shown by a comparison with five widely used unsupervised dimensionality reduction methods including the classic auto-encoder.

Keywords: Auto-encoder · Deep learning · Cosine similarity · Neighborhood · Document clustering · Unsupervised learning · Dimensionality reduction

1 Introduction

Analyzing sparse high-dimensional point clouds is a classical challenge in visualization. Principal component analysis (PCA), one of the traditional techniques, is certainly the best known. More efficient in nonlinear cases, a number of techniques have been proposed, including Isometric Feature Mapping (**Isomap**), Locally Linear Embedding (**LLE**), and Stochastic Neighbor Embedding (**SNE**). Nevertheless these nonlinear techniques tend to be extremely sensitive to noise, sample size, choice of neighborhood and other parameters (for details see for instance [1]). On the other hand, **t-SNE** [2] and its parametric version [3] is better than existing techniques at creating a single map that reveals structure at many different scales. Parametric **t-SNE** learns the parametric mapping in such

a way that the local structure of the data is preserved as well as possible in the latent space. Generally, it works better in the case of image datasets but it is very dependent on the adjustments of the hyper parameters, e.g. learning rate noted η . Laplacian Eigenmap (LE) [4] is another interesting method where the laplacian graph is used and has relatively the same objective as \mathfrak{t} -SNE, i.e. preserving the local structure of data.

The auto-encoders, a special method of deep learning architecture, have received more attention recently for dimensionality reduction tasks; their abilities to adapt to different domains are promising. They make it possible to embed the high dimensional data in a latent space of lower dimensionality while preserving the original structure of the data. In its traditional version, each data point is used to reconstruct itself from the code layer. If we have the same number of neurons in the code layer as in the input layer, the method learns the identity function. In order to avoid this trivial solution, there are many different approaches. The two most used consist in (1) using fewer number of neurons in the code layer so as to force the auto-encoder to compress the features in a lower space, (2) introducing some noise to data, for instance with a Gaussian noise applied to the whole data or randomly replacing with zeros a percentage of data entries. It is proved that some denoising auto-encoders (DAEs) correspond to a Gaussian RBM (Restricted Boltzmann Machine) in which minimizing the denoising reconstruction error estimates the energy function [5,6]. They generally give better results in comparison to classic auto-encoders without any denoising step. We make use of the former type of auto-encoders in the following. In this paper, we concentrate on the case of sparse high dimensional data and in particular on document-term matrices. The cells of such matrices contain the frequency counts of the terms belonging to the corresponding documents. We know that the auto-encoders aim to find the low dimensional embeddings in data by preserving the structure of the data as well as possible. Herein, with the proposed auto-encoder, we aim to capture the relations among documents while preserving the original structure. Therefore, the proposed method focuses on the dimensionality reduction and the main contributions of the paper, presented schematically in Fig. 1, are as follows:

- we propose a suitable normalization of document-term matrices;
- we introduce a weighted criterion where the weights rely on cosine similarity, and derive an appropriate autoencoder able to effectively reduce the dimension;
- finally in order to cluster the set of documents, we perform the Expectation-Maximization [7] algorithm for Gaussian mixture models on the reduced space instead of the k-means algorithm which is commonly used, and assess the number of clusters relying on the *Bayes Information Criterion* [8].

The rest of the paper is organised as follows. In Sect. 2, we first introduce different types of preprocessing that are mandatory in order to get the auto-encoders work and then the role of the denoising procedure is described. Section 3, is devoted to the introduction of the mathematical formulation of the proposed auto-encoder. The experimental results on different text datasets and

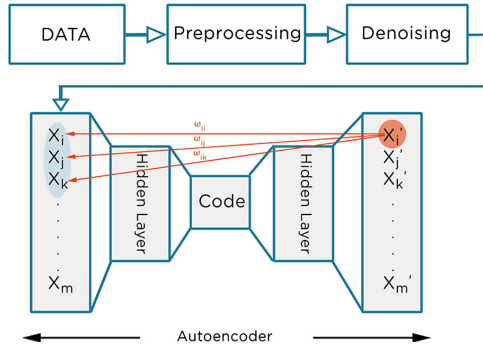


Fig. 1. Proposed method scheme (Color figure online)

clustering are presented in Sect. 4. Section 5 concludes the paper and presents directions for future research.

2 Data Pre-processing

Let $\mathbf{x}_1, \dots, \mathbf{x}_m$ be a set of m objects where \mathbf{x}_i is a n -dimensional vector on \mathbb{R}^n . In practice, \mathbf{x}_i contains the variables corresponding to p measurements made on the i th recording of some features on the phenomenon under study. Then data will be denoted by an m by n data matrix $\mathbf{x} = (x_{ij})$. Before applying of any clustering or dimensionality reduction algorithm, a preprocessing step is necessary in order to reduce the effect of the outliers and prepare the data for a better and more faithful analysis. In the context of dimensionality reduction of document-term data, different normalization methods are available which increase the performance of such methods e.g. TF-IDF, mutual information or χ^2 normalization. While this type of normalizations is not widely employed for deep neural networks, the use of TF-IDF normalization followed by centering, yields good results. On the other hand it is shown in [9] that the other types of normalizations could contribute to the deep architectures to work better and that consists in (1) centering (2) applying KL-Expansion or PCA (3) using covariance equalization. The two steps (2) and (3) can be combined by applying a PCA with *whitening*; see for instance [10]. As the networks learn the fastest from the most unexpected sample, it is recommended to choose a sample at each iteration which is the most unfamiliar to the system [9]. In order to rely on this hypothesis, after the normalization step above, training data are shuffled to ensure that the successive examples are not drawn from the same class. Finally we train the deep auto-encoder as a pretraining step demonstrated in [11], by considering each layer separately as a simple auto-encoder. The activations of a layer below become the input for the next layer as in Fig. 2. In each layer we add some noise to the data by replacing randomly 30% of the data by zeros, a procedure called corruption or denoising. With this approach, we approximate the performance of RBMs with a lower cost in terms of complexity.

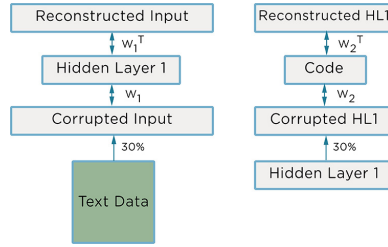


Fig. 2. Layer wise pretraining of a denoising autoencoder

3 Auto-encoder for Text Analysis

3.1 Classic Auto-encoders

Auto-encoders are traditionally composed of two parts, an encoder where the high dimensional feature space of data is encoded and compressed to a lower dimensional feature space by a function h as $\mathbf{y}_i = h(W\tilde{\mathbf{x}}_i + b)$ where $\tilde{\mathbf{x}}$ is a corrupted input obtained by following the denoising procedure explained in the last section, W is the weight matrix between the input and hidden layer as in Fig. 2 where $W \in \mathbb{R}^{d \times p}$ (d number of neurons in hidden layer, p number of input features), b is the bias term where $b \in \mathbb{R}^{d \times 1}$ and \mathbf{y}_i contains hidden layer values in middle layers and code values in the last layer. The parameters W and b are estimated using mini batch gradient descent algorithm in each iteration of optimization and h is an activation function that can be a linear or non linear such as *sigmoid* or *hyperbolic tangent*. In this paper we opt for the *hyperbolic tangent* which generally provides better results. The second part of the auto-encoder consists of a decoder which tries to reconstruct the original data from the code layer \mathbf{y}_i by $\mathbf{x}'_i = g(W'\mathbf{y}_i + b')$. The layer of reconstructions has the same dimensionality as the input layer. As the encoder part, the decoder layer has also the same type of parameters ($W' \in \mathbb{R}^{p \times d}$ and $b' \in \mathbb{R}^{p \times 1}$) that must be adjusted, and an activation function g ; like h it can be a linear or non linear function. In classic auto-encoders, each example \mathbf{x}'_i tries to reconstruct the original input \mathbf{x}_i from the code or hidden layer \mathbf{y}_i . Therefore, the cost function takes the following form

$$C(\theta) = \arg \min \sum_{i=1}^m \|g(W'h(W\tilde{\mathbf{x}}_i + b) + b') - \mathbf{x}_i\|^2 \tag{1}$$

where C is a cost function with $\theta = (W, b, W', b')$ the unknown parameters to estimate by minimizing this function. The symbol $\|\cdot\|$ denotes an euclidean distance between the reconstructed examples and original input.

3.2 The Proposed Unsupervised Auto-encoder

The classic auto-encoder, presented above and referred to as **C-autoencoder** in the sequel, is not able to capture the original structure in the data, and to reveal

the latent structure in the case of complex data. In order to achieve this, one can modify the cost function (1) in a way where each example \mathbf{x}'_i , in addition to reconstructing the correspondent original input \mathbf{x}_i , also reconstructs the data points that are in the neighborhood of \mathbf{x}_i , using *cosine similarity* metric. Moreover, each reconstruction term has a weight; this leads to the construction of a weighted graph with edges connecting nearby documents to each other. At first glance the idea is relatively similar to that in [12], but the prior normalization step following a novel auto-encoder configuration and regularization make this approach more relevant to cluster sparse text data, where the sparsity is regularized in order to avoid overfitting. This procedure is depicted in Fig. 1, where the example \mathbf{x}'_i designated by a red circle reconstructs its correspondent in input layer i.e. \mathbf{x}_i and its k nearest neighbors in input layer i.e. $\{\mathbf{x}_j, \mathbf{x}_k\}$, that are marked by a blue ellipse. The weights between these reconstruction terms are denoted by ω . So the cost function for the proposed auto-encoder becomes,

$$C(\theta) = \arg \min \sum_{i=1}^m \sum_{\ell \in \Psi_i} \omega_{i\ell} \|\mathbf{x}'_i - \mathbf{x}_\ell\|^2. \quad (2)$$

where Ψ_i denotes the set of the k nearest neighbors of the document \mathbf{x}_i and ω (not to be confused with W) is the weight associated to document \mathbf{x}_i and document \mathbf{x}_ℓ belonging to Ψ_i . The set of parameters θ of the network in Eq. (1) holds also for the new loss function. The weight draws on *Laplacian Eigenmaps* where heat kernels are used to choose the weight decay function (parameter $t \in \mathbb{R}$) and the cosine between two documents is used as a similarity measure between them. It takes the following form

$$\omega_{i\ell} = \exp\left(-\frac{\cos(\mathbf{x}_i, \mathbf{x}_\ell)}{t}\right) \quad (3)$$

where t is a hyper parameter to adjust. The details on the choice of t is discussed in [4]. Note that with $t = 1$, two very similar documents lead to $\omega_{i\ell} \approx 1/e$, and so similar documents in embeddings are less penalized; while two distinct documents lead to $\omega_{i\ell} \approx 1$ and so they are more penalized. Furthermore, we have considered the sparsity regularization term in the cost function as follows,

$$C_{sparse}(\theta) = C(\theta) + \beta \sum_{j=1}^s KL(\rho || \hat{\rho}_j) \quad (4)$$

where β controls the importance of the regularization term and s is the number of neurons in hidden layer. $KL(\rho || \hat{\rho}_j)$ is the Kullback-Leibler divergence between ρ a sparsity parameter and $\hat{\rho}_j$ its approximation by $\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^m \mathbf{y}_i^{(j)}$ the average activation of hidden unit j ; $KL(\rho || \hat{\rho}_j)$ can be thought of as a measure of the information lost when $\hat{\rho}_j$ is used to approximate ρ . The details of this regularization are available in [13] and given by

$$KL(\rho || \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}. \quad (5)$$

Hereafter, we describe in Algorithm 1, referred as **T-autoencoder**, the main steps of the method, optimizing (4); we assume that $W' = W^\top$ as is often the case in the literature.

Algorithm 1. Unsupervised auto-encoder for Text data (**T-autoencoder**)

Input training set $\{\mathbf{x}_i\}_1^m$

Hypothesis: Tied weights i.e. $W' = W^\top$; fixed sparsity level $\rho = 0.05$

Parameters: $\theta=(W, b, b')$

Notation: Ψ_i reconstruction set for \mathbf{x}_i

ω weights between \mathbf{x}_i and Ψ_i

1. Compute the cosine similarity between documents and determine reconstruction set Ψ by k -nearest neighbor algorithm for each example.
 2. Compute the weights ω between each example \mathbf{x}_i and its reconstruction set Ψ_i , as in (3).
 3. Update θ minimizing the Cost function C_{sparse} in (4).
 4. Update reconstruction set Ψ and weights ω with respect to each hidden layer $\{\mathbf{y}_i\}_1^m$ separately.
 5. Repeat 3 and 4 until convergence.
-

The time complexity of **T-autoencoder** is $\mathcal{O}(n^2)$ for cosine similarity computation, $\mathcal{O}(n \log n)$ for finding k -nearest neighbors and $\mathcal{O}(batch_size \times k)$ to calculate each weighted reconstruction term in (2) in addition to the time complexity of neural networks; where k is the number of neighbors. As $n \rightarrow \infty$ the added time complexity approaches $\mathcal{O}(n^2)$.

4 Experiments

In order to evaluate the performance of **T-autoencoder**, we performed experiments on different document-term datasets. Our implementations are based on *python* and **R** languages, and the *theano* library in order to use the performance of GPU for accelerating computations.

4.1 Experimental Setup

The characteristics of datasets¹ used in experimentation are presented in Table 1. Each dataset presented has its own complexity e.g. excessive number of variables (Curse of dimensionality), high number of clusters or the complexity pertaining to the data structure. We compared the **T-autoencoder** with a linear method (PCA), and three non-linear methods (**Isomap**, **LE** and **t-SNE**). We run the methods which require the hyper parameters as number of neighbors or learning rate

¹ <http://dataexpertise.org/research>.

to be adjusted e.g. `Isomap`, `t-SNE` and `T-autoencoder`, with diverse configurations, and finally we picked the values corresponding to a minimum reconstruction error. As an example, for `t-SNE` we have opted for $\eta = 100$, as it provides better results than other configurations. In addition to the state of the art methods, `C-autoencoder` is also considered in experiments, in order to point out the improvement attained using `T-autoencoder`. We evaluated the performance of these methods by means of three-dimensional plots of embeddings and also by measuring different metrics such as Normalized Mutual Information (NMI) [14], Adjusted Rand Index (ARI) [15] and Purity after applying the Expectation-Maximization algorithm [7] for Gaussian Mixtures Models (GMM) instead of k-means which is based on a restricted Gaussian mixture.

Table 1. Datasets used for experimentation, # denotes the cardinality.

Dataset	#Documents	#Terms	Number of clusters
Classic3	3891	4303	3
CSTR	475	1000	4
20news	3970	8014	4
NG5	500	2000	5
Reviews	4069	18483	5
TR45	690	8261	10
TDT2.10	653	36771	10

For the auto-encoder part a $n - \frac{n}{2} - \frac{n}{2^2} - \dots - d$ architecture is used, where n represents the dimensionality of the data and d represents the dimensionality of the latent space that should be attained in code layer. In this experimentation we opt for three dimensional latent space, so $d = 3$. After extensive numerical experiment trials, $t = 0.5$ in (3) appears appropriate, so we did not get involved as much with tuning of such hyper parameters. The auto-encoders were trained using the layer-wise pretraining procedure explained before, and are fine-tuned by performing back-propagation such as minimizing the weighted sum of squared errors between each example and its reconstruction set. We used a decreasing learning rate, starting from a large value in higher layers and reducing it gradually in lower layers. Weight decay was set to 0.0001 for all the layers. In our experiments, we opted for $\beta = 0.01$ and $\rho = 0.05$ for regularization term in (4); where ρ controls the sparseness of representation, and has a fixed value obtained via experimentation for all the datasets. Furthermore mini batch gradient descent method was used to adjust weights and biases; the batch size was fixed at 100.

4.2 Results

In this section, the results of the above mentioned methods are presented by means of the visualization of the embeddings (Figs. 3, 4 and 5). Furthermore

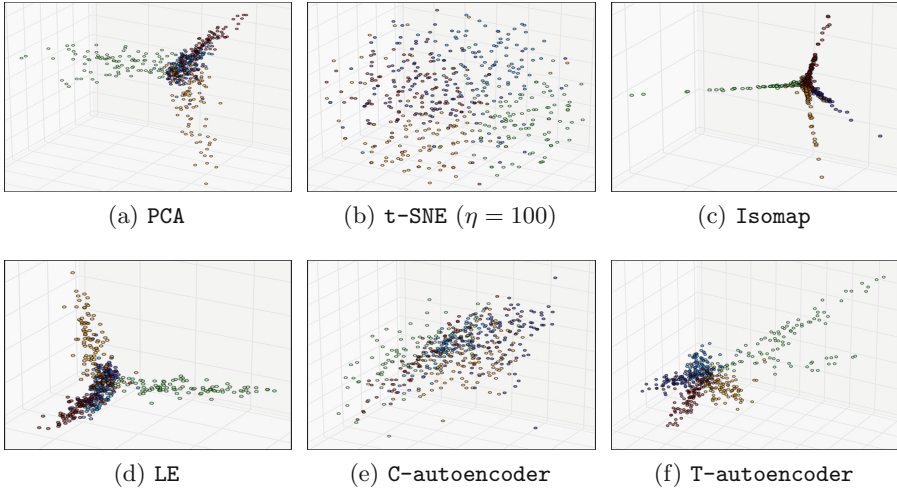


Fig. 3. Visualization of 500 documents from NG5 dataset by different unsupervised dimensionality reduction methods

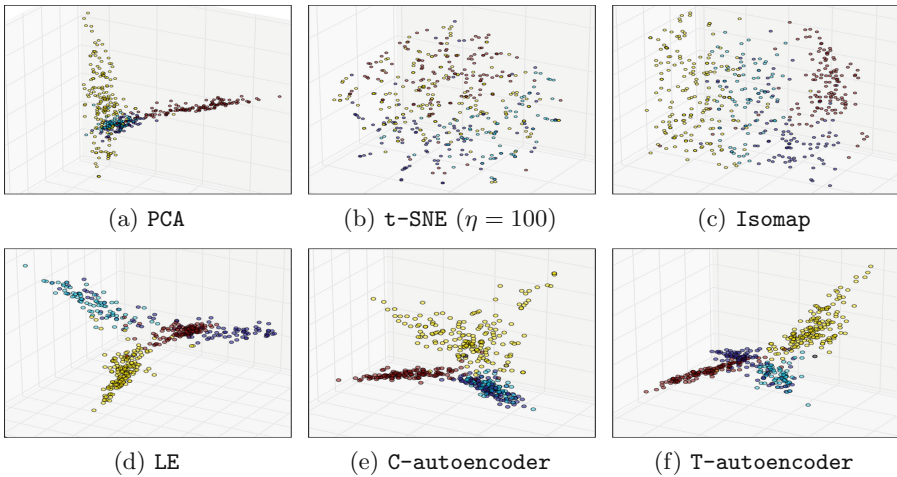


Fig. 4. Visualization of 475 documents from CSTR dataset by different unsupervised dimensionality reduction methods

in order to have more precise comparisons, the embeddings are clustered in homogeneous groups and are compared with true labels (Table 2).

Visualisation and Clustering. To illustrate the interest of T-autoencoder in terms of visualization and clustering, because of the lack of space, we chose to only present the visualizations of NG5, CSTR and TDT2_10 datasets obtained by all the presented methods in Figs. 3, 4 and 5. In Table 2 comparisons are

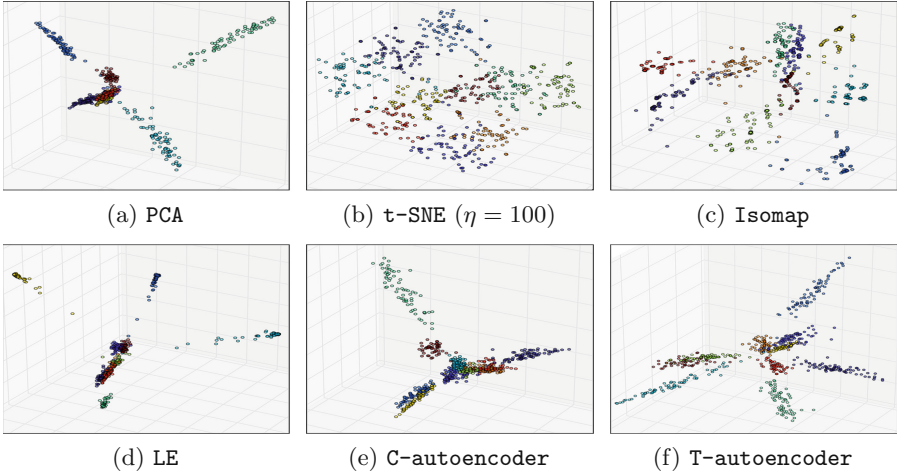


Fig. 5. Visualization of 653 documents from TDT2-10 dataset by different unsupervised dimensionality reduction methods

reported for all the datasets in Table 1 using NMI, ARI and purity metrics after applying the EM algorithm [7]. For each experiment the best performance is highlighted in bold type. Note that EM is conducted considering the General Gaussian Mixture (GMM) Model noted **VVV** in the sequel [16]. In the first step we consider that the number of clusters is known. The different Gaussian models are based on the cluster proportions and three characteristics of clusters (volume, shape, orientation) that can be equal (**E**) or variable (**V**) (for details see, [17]). For instance **EVV** corresponds to the model where the clusters have the same volume but the shapes and orientations are different. In Fig. 3 we observe that the latent structure of NG5 dataset is relatively complex and all the presented methods have difficulties in distinguishing the five existing clusters. For example **PCA** and **LE** can distinguish only three clusters and two remaining clusters are mixed together while **Isomap** can only recognize some documents from the four clusters while fifth group and the rest of documents are mixed in the center. Furthermore it gives the worst performance (see Table 2). Using **t-SNE**, we can see that in almost all the visualizations, the data are more dispersed than with other methods, but the examples of the same cluster remain close to each other; **t-SNE** gives the second best result. **C-autoencoder** cannot identify the frontier between clusters whereas the result from Fig. 3(f) and Table 2 reveals a good performance of **T-autoencoder** compared to the others. Using this method, we can observe a good separability between the five groups of documents and the best results in terms of purity, NMI and ARI.

In Fig. 4 we observe that two clusters are mixed together using most of the methods, proving a complex latent structure of them. Considering **PCA** and **C-autoencoder**, this complexity can be clearly observed. On the other hand **t-SNE** is not able to capture the existing relations with other clusters too while

Table 2. Comparison of all the presented methods in terms of clustering using NMI, ARI and Purity (EM: Expectation-Maximization with the VVV model, KM: K-means)

Datasets	Metrics	PCA	t-SNE	Isomap	LE	C-autoencoder		T-autoencoder	
		EM				KM	EM	KM	EM
Classic3	NMI	0.89	0.75	0.90	0.91	0.71	0.86	0.89	0.96
	ARI	0.93	0.76	0.94	0.94	0.70	0.91	0.93	0.93
	Purity	0.97	0.90	0.98	0.98	0.88	0.97	0.97	0.98
CSTR	NMI	0.70	0.70	0.63	0.76	0.62	0.69	0.72	0.77
	ARI	0.69	0.76	0.56	0.81	0.50	0.64	0.70	0.82
	Purity	0.75	0.88	0.63	0.90	0.67	0.74	0.85	0.91
20news	NMI	0.27	0.52	0.65	0.62	0.22	0.57	0.61	0.71
	ARI	0.06	0.53	0.69	0.53	0.16	0.52	0.54	0.74
	Purity	0.41	0.77	0.86	0.76	0.43	0.66	0.81	0.89
NG5	NMI	0.50	0.49	0.25	0.56	0.31	0.41	0.54	0.60
	ARI	0.38	0.51	0.07	0.49	0.20	0.33	0.38	0.51
	Purity	0.56	0.76	0.38	0.66	0.53	0.55	0.72	0.79
Reviews	NMI	0.31	0.31	0.50	0.59	0.31	0.46	0.45	0.54
	ARI	0.13	0.24	0.43	0.54	0.16	0.44	0.28	0.48
	Purity	0.44	0.49	0.61	0.65	0.47	0.66	0.60	0.66
TR45	NMI	0.49	0.63	0.54	0.56	0.46	0.56	0.64	0.66
	ARI	0.36	0.51	0.48	0.43	0.30	0.37	0.52	0.57
	Purity	0.48	0.59	0.61	0.46	0.49	0.58	0.65	0.67
TDT2.10	NMI	0.70	0.96	0.81	0.80	0.71	0.87	0.80	0.98
	ARI	0.30	0.95	0.70	0.57	0.43	0.81	0.66	0.98
	Purity	0.48	0.97	0.80	0.70	0.52	0.85	0.75	0.99

Isomap shows good performance on this dataset in terms of separability of clusters and clustering. Although the documents are dispersed in different directions there is not a clear separation between four existing groups of documents. The best visualizations are obtained using **LE** and **T-autoencoder**, where we can clearly see that each projected cluster has its own direction. In Table 2 we can also observe their higher performances in terms of all the metrics used.

In Fig. 5, the number of clusters is higher than in the two previous examples. We note that **PCA**, **Isomap** and **LE** are not able to recognize all the ten obtained clusters while **LE** has shown a good performance and **t-SNE** provides a good result in terms of clustering but not in terms of visualisation (Table 2). Finally, the good performance of **T-autoencoder** in terms of visualisation and clustering is easy to observe; **T-autoencoder** clearly outperforms **C-autoencoder**.

Assessing the Number of Clusters. Another reason why we used the GMM for clustering instead of a simple k-means algorithm is that, this approach offers the flexibility to fit the data, using an appropriate model. As we know, estimating the number of clusters for the input of a clustering algorithm is essential and hard to achieve. So to estimate it, we have used the *Bayesian Information Criterion* (BIC) [8] given by $BIC_{M,k} = -2L_{M,k} + v_{m,K} \log m$, where M is the model and k is the number of components. $L_{M,k}$ is the maximum likelihood for M and k and v is the number of free parameters in the model M with k components. This criterion penalizes the number of parameters in model and maximizes the likelihood of data simultaneously; it is efficient on a practical ground. To choose the best model for mixture model with an appropriate number of clusters, we have considered the BIC criterion with different numbers of components (clusters) K on the latent space obtained. Due to the lack of space, we propose to illustrate the contribution of BIC on two data sets TDT2-10 and NG5. In Fig. 6(a), BIC takes the maximum value when the number of components is 10 with the VVV model (see Fig. 6(c)). In Fig. 7(a), we observe that the highest value of BIC is attained for the model VVV with 6 clusters (marked by red vertical dotted line) instead of 5 clusters (marked by green vertical dotted line) with the EVV model; it is also an ellipsoidal model which considers the same volume, but different shape and orientation for clusters. Notice that both suggestions of BIC are interesting; Fig. 7 reinforces them. In Fig. 7(c), we simulated the scheme of NG5 visualization depicted in Fig. 7(b), we have relatively five clusters with different shapes and orientations. The orientations are shown by arrows, and shapes by dotted ellipses.

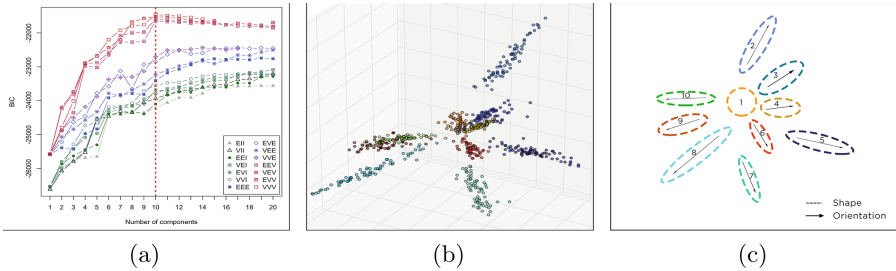


Fig. 6. TDT2-10 dataset: Bic plot related to the latent space obtained by T-autoencoder in (a), Latent space obtained by T-autoencoder with ground truth clusters in (b), Estimated clusters scheme by mixture model using BIC criterion in (c).

In short, we observe that clustering on embeddings obtained via T-autoencoder often outperforms all the presented methods including C-autoencoder. This is due to the main difference in the architecture of the proposed method in comparison with C-autoencoder. As mentioned earlier, T-autoencoder is trained to reconstruct data from the corrupted input. This procedure increases its ability to be less dependent on training data while

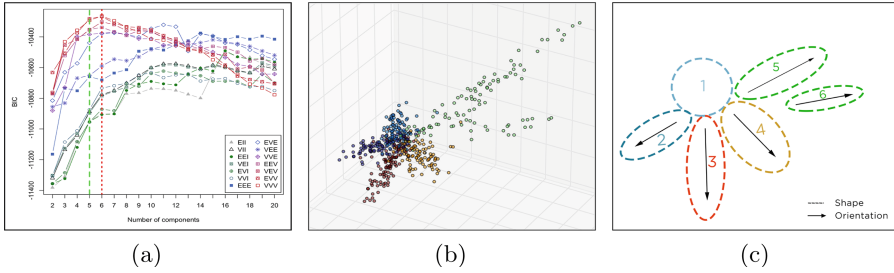


Fig. 7. NG5 Dataset: Bic plot related to the latent space obtained by **T-autoencoder** in (a), Latent space obtained via **T-autoencoder** with ground truth clusters in (b), Estimated clusters scheme by mixture model using BIC criterion in (c). (Color figure online)

promoting close documents; the GMM via EM confirms this performance by providing a better clustering of documents. On the other hand, the autoencoders generally do not construct low-dimensional data representations in which the natural clusters are widely separated. This could also be due to shortcoming of auto-encoders where latent relations in data cannot be discovered. Unlike **C-autoencoder**, **T-autoencoder** learns these relations by reconstructing an example from its k -nearest neighbors according to the more suitable cosine similarity for document-term matrices.

5 Conclusion

In this paper a text specific version of denoising auto-encoders has been proposed. We have seen that appropriate normalization applied on the set of documents combined with the use of a suitable weighted criterion where the weights rely on the cosine similarity among documents is effective. The accuracy of auto-encoders in determining the latent structure of data has been improved for the task of dimensionality reduction and therefore for clustering by exploiting the potential of GMM and BIC. Consequently, auto-encoders do not only aim at maximizing the variance of the data, but also discovering the potential structure in clusters.

The interest of our approach is to demonstrate the accuracy of the proposed method in the buoyant field of visualization and document clustering [18, 19]. The efficiency in terms of time complexity is, however, another issue that could be considered in future works. Although we have used the GPU performance using the *theano* library, the efficiency should be improved by using more recent optimization methods such as BFGSs which converge faster than gradient descent.

References

1. Gittins, R.: *Canonical Analysis - A Review with Applications in Ecology*. Springer, Heidelberg (1985)
2. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. *J. Mach. Learn. Res.* **9**(Nov), 2579–2605 (2008)
3. van der Maaten, L.: Learning a parametric embedding by preserving local structure. *RBM*, 500:500 (2009)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. *NIPS* **14**, 585–591 (2001)
5. Bengio, Y.: Learning deep architectures for ai. *Found. Trends Mach. Learn.* **2**(1), 1–127 (2009)
6. Vincent, P.: A connection between score matching and denoising autoencoders. *Neural Comput.* **23**(7), 1661–1674 (2011)
7. Dempster, A.P., Nan Laird, M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. *J. Roy. Stat. Soc. Ser. B (methodological)* **39**, 1–38 (1977)
8. Schwarz, G., et al.: Estimating the dimension of a model. *Ann. Stat.* **6**(2), 461–464 (1978)
9. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient BackProp. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade*. LNCS, vol. 7700, pp. 9–48. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-35289-8_3](https://doi.org/10.1007/978-3-642-35289-8_3)
10. Jégou, H., Chum, O.: Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) *ECCV 2012*. LNCS, pp. 774–787. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-33709-3_55](https://doi.org/10.1007/978-3-642-33709-3_55)
11. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
12. Wang, W., Huang, Y., Wang, Y., Wang, L.: Generalized autoencoder: a neural network framework for dimensionality reduction. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 490–497 (2014)
13. Ng, A.: Sparse autoencoder. *CS294A Lecture Notes*, vol. 72, pp. 1–19 (2011)
14. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **3**, 583–617 (2003)
15. Hubert, L., Arabie, P.: Comparing partitions. *J. Classif.* **2**, 193–218 (1985)
16. Banfield, J.D., Raftery, A.E.: Model-based gaussian and non-gaussian clustering. *Biometrics* **49**, 803–821 (1993)
17. Fraley, C., Raftery, A.E.: *Mclust version 3: an R package for normal mixture modeling and model-based clustering*. Technical report (2006)
18. Priam, R., Nadif, M.: Data visualization via latent variables and mixture models: a brief survey. *Pattern Anal. Appl.* **19**(3), 807–819 (2016)
19. Allab, K., Labiod, L., Nadif, M.: A semi-NMF-PCA unified framework for data clustering. *IEEE Trans. Knowl. Data Eng.* **29**(1), 2–16 (2017)