ESANN 2019

Deep Embedded SOM: Joint Representation Learning and Self-Organization

Florent Forest[1,2], Mustapha Lebbah[1], Hanane Azzag[1] & Jérôme Lacaille[2]
1 - Laboratoire d'Informatique de Paris-Nord (LIPN), Université Paris 13, France
2 - Safran Aircraft Engines, France

forest@lipn.univ-paris13.fr
http://florentfo.rest/
FlorentF9

April 24, 2019

Table of contents

Deep Representation Learning for Clustering

Deep Clustering

Dimensionality reduction (e.g. PCA, autoencoders) improves high-dimensional data clustering for algorithms relying on euclidean distance (e.g. k-means, SOM).

Principles of deep clustering

- Clustering is performed in the intermediate feature space of deep neural networks (*autoencoders*).
- Representation learning and clustering are considered as a single task.
- The objective is to learn *cluster-friendly* representations that improve clustering performance.

Generally achieved via a hybrid loss function that trades off between representation quality and clustering (regularization).

Baselines

Common external metrics: purity, NMI, clustering accuracy.

Baseline	Accuracy (%)
k-means	58
AE + k-means/GMM	82
Song et al. [Song et al., 2014]	76.0
DEC [Xie et al., 2015]	84.30
DCN [Yang et al., 2016]	83.0
IDEC [Guo et al., 2017]	88.06
VaDE [Jiang et al., 2017]	94.46
WaMiC [Harchaoui et al., 2018]	98.42

Table 1: Clustering baselines on MNIST

General notations

$\mathsf{x}_i, i = 1 \dots N$	data points
$\mathsf{m}_k, k = 1 \dots K$	cluster centers
χ	cluster assignment mapping
	$\chi(i) = k$ iff x_i belongs to cluster k
$\mathsf{W_e}, \mathsf{W_d}$	encoder and decoder parameters
$\mathsf{z}_i = \mathsf{f}_{\mathsf{W_e}}(\mathsf{x}_i)$	encoded data point (latent space)
$\tilde{\mathsf{x}}_i = \mathsf{g}_{\mathsf{W_d}}(\mathsf{z}_i)$	decoded data point (reconstruction)

Related work

Methods based on k-means

Principle: jointly optimize reconstruction and k-means loss.

$$\mathcal{L}(\mathbf{W_e}, \mathbf{W_d}, \mathbf{m}_1, \ldots, \mathbf{m}_K, \chi) = \mathcal{L}_r(\mathbf{W_e}, \mathbf{W_d}) + \gamma \mathcal{L}_{km}(\mathbf{W_e}, \mathbf{m}_1, \ldots, \mathbf{m}_K, \chi)$$
$$= \frac{1}{N} \sum_i ||\tilde{\mathbf{x}}_i - \mathbf{x}_i||^2 + \gamma \frac{1}{N} \sum_i ||\mathbf{z}_i - \mathbf{m}_{\chi(i)}||^2$$

where:

$$\chi(i) = \operatorname*{argmin}_k ||\mathbf{z}_i - \mathbf{m}_k||^2$$

- Alternating procedure: (1) update cluster assignments, (2) update network parameters and centroids [Song et al., 2014, Yang et al., 2016]
- Continuous relaxation of k-means loss [Fard et al., 2018]

Related work

Deep Embedded Clustering (DEC) [Xie et al., 2015]

1. Pretrain only with reconstruction loss using SAE (Stacked Autoencoder) [Vincent et al., 2010].
2. Learn the cluster centers and fine-tune the encoder using only *soft hardening loss* (KL-divergence between soft cluster assignment and *hardened* distribution).

$$q_{ik} = \frac{(1 + ||\mathbf{z}_i - \mathbf{m}_k)||^2)^{-1}}{\sum_{k'}(1 + ||\mathbf{z}_i - \mathbf{m}_{k'})||^2)^{-1}}, \; p_{ik} = \frac{q_{ik}^2 / \sum_i q_{ik}}{\sum_{k'} \left(q_{ik'}^2 / \sum_i q_{ik'}\right)}$$

$$\mathcal{L}(\mathbf{W_e}, \mathbf{m}_1, \dots, \mathbf{m}_K) = D_{KL}(p||q) = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}}$$

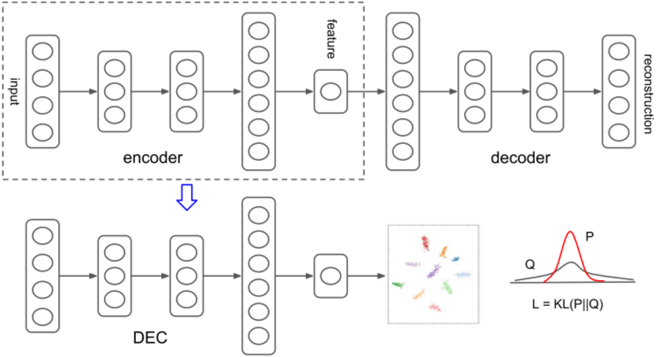

Figure 1: DEC architecture

Related work

Improved Deep Embedded Clustering (IDEC) [Guo et al., 2017]

Principle: training with only a clustering loss corrupts feature space in DEC and hurts clustering performance.

1. Pretrain AE as in DEC.
2. Learn the cluster centers and fine-tune the encoder and decoder using a weighted sum of the reconstruction loss and the clustering loss (KL-divergence).

$$
\begin{aligned}
\mathcal{L}(\mathbf{W_e}, \mathbf{W_d}, \mathbf{m}_1, \dots, \mathbf{m}_K) &= \mathcal{L}_r(\mathbf{W_e}, \mathbf{W_d}) + \gamma \mathcal{L}_c(\mathbf{W_e}, \mathbf{m}_1, \dots, \mathbf{m}_K, \chi) \\
&= \frac{1}{N} \sum_i ||\tilde{\mathbf{x}}_i - \mathbf{x}_i||^2 + \gamma D_{KL}(p||q)
\end{aligned}
$$

Related work

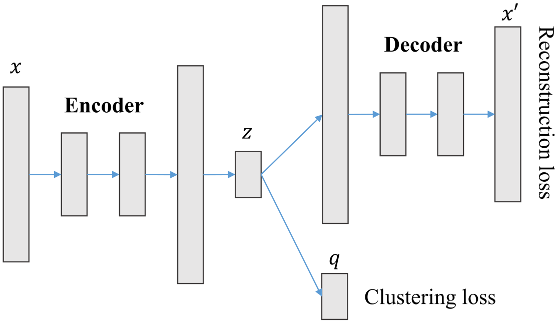

Figure 2: IDEC architecture

DESOM: Deep Embedded Self-Organizing Map

Joint Representation Learning and Self-Organization

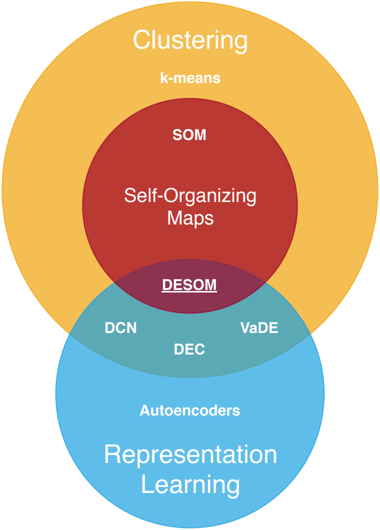

Figure 3: ”Venn” diagram of joint RL and clustering

Related Work

- *Deep Neural Maps*, Pesteie, M., Abolmaesumi, P., & Rohling, R. (2018). [Pesteie et al., 2018] (workshop track ICLR 2018)
- *Deep Self-Organization: Interpretable Discrete Representation Learning on Time Series*, Fortuin, V., Hüser, M., Locatello, F., Strathmann, H., & Rätsch, G. (2018). [Fortuin et al., 2018] (accepted at ICLR 2019)

Principle

Joint training of a deep autoencoder and a self-organizing map [Kohonen, 1982]. The SOM prototypes are learned in latent space, and self-organization and representation learning are achieved as a joint end-to-end task.

- Handles high-dimensional data
- Improves clustering performance (*SOM-friendly* space)
- Joint training and no pre-training → cuts training time
- Trains on GPU

- Keras implementation available on Github[1]

[1] `https://github.com/FlorentF9/DESOM`

DESOM Architecture

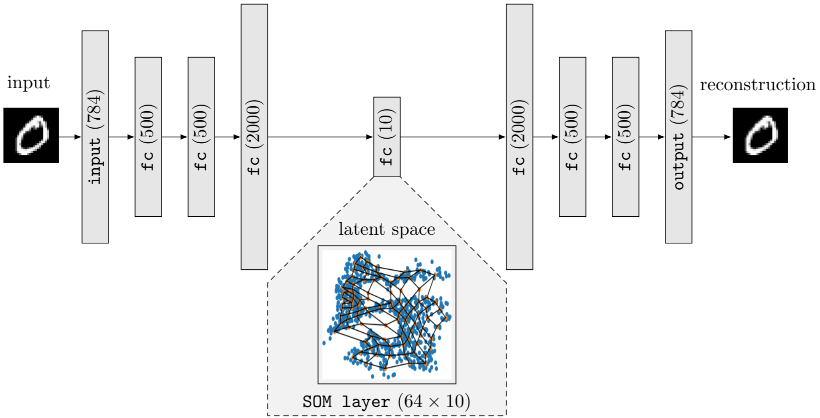

Figure 4: DESOM architecture

Loss Function

Loss function:

$$\mathcal{L}(\mathsf{W_e}, \mathsf{W_d}, \mathsf{m}_1, \ldots, \mathsf{m}_K, \chi) = \mathcal{L}_r(\mathsf{W_e}, \mathsf{W_d}) + \gamma \mathcal{L}_{som}(\mathsf{W_e}, \mathsf{m}_1, \ldots, \mathsf{m}_K, \chi)$$

MSE reconstruction loss:

$$\mathcal{L}_r = \frac{1}{N} \sum_i ||\tilde{\mathsf{x}}_i - \mathsf{x}_i||^2$$

SOM loss:

$$\mathcal{L}_{som} = \frac{1}{N} \sum_i \sum_{k=1}^{K} \mathcal{K}^T\left(\delta(\chi(i), k)\right) ||\mathsf{z}_i - \mathsf{m}_k||^2$$

where $\chi(i) = \underset{k}{\mathrm{argmin}} ||\mathsf{z}_i - \mathsf{m}_k||^2$ and $\mathcal{K}^T(d) = e^{-d^2/T^2}$

Training Procedure

Let us define the weighting terms $w_{i,k} := \mathcal{K}^T(\delta(\chi(i), k))$.
At each iteration, we compute the cluster assignments and fix the terms $w_{i,k}$. Then, all parameters $\{\mathbf{W_e}, \mathbf{W_d}, \mathbf{m}_1, \ldots, \mathbf{m}_K\}$ are optimized using SGD (Adam [Kingma and Ba, 2015]).

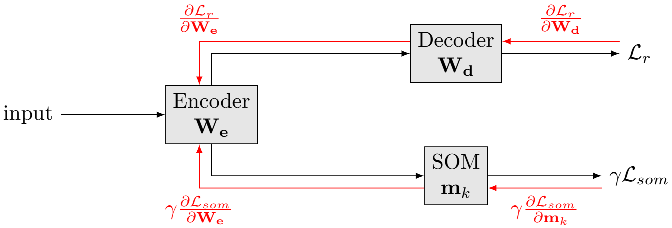

Figure 5: DESOM gradients flow

Training Procedure

Input: training set $\mathbb{X}$; SOM topology; T_{max}, T_{min}; *iterations*; *batchSize*

Output: AE weights $\mathbf{W_e}$, $\mathbf{W_d}$; SOM code vectors $\{\mathbf{m}_k\}$

Initialize $\mathbf{W_e}$, $\mathbf{W_d}$ (Glorot uniform scheme) ;

Initialize $\{\mathbf{m}_k\}$ (with random data sample) ;

for $iter = 1, \ldots, iterations$ **do**

  $T \leftarrow T_{max} \left(\frac{T_{min}}{T_{max}}\right)^{iter/iterations}$;

  Load next training batch ;

  Encode current batch and compute weights $w_{i,k}$;

  Train DESOM on batch by taking a SGD step ;

end

Algorithm 1: DESOM training procedure

Differences with existing research

Deep Neural Maps [Pesteie et al., 2018]

- Alternating procedure: (1) compute embeddings, (2) update centers using stochastic Kohonen algorithm, (3) fix centers and update AE
- IDEC loss function (using KL-divergence soft hardening loss)
- SAE pre-training
- Accent on visualization, no clustering benchmark

DESOM [Forest et al., 2019]

- Alternating procedure: (1) compute pairwise distances between embeddings and centers, (2) update all parameters (AE and centers)
- Joint reconstruction and SOM loss (using the real SOM loss)
- No pre-training
- Competitive clustering performance

Differences with existing research

SOM-VAE [Fortuin et al., 2018]

- Variational AE
- Discrete latent space (inspired from VQ-VAE [van den Oord et al., 2017])

DESOM [Forest et al., 2019]

- Deterministic AE
- Continuous latent space

Visualization and Quantitative Results

Visualization

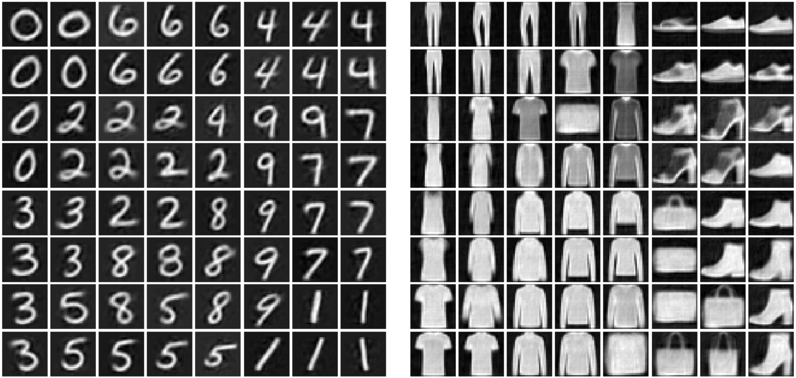

Figure 6: DESOM visualizations of MNIST and Fashion-MNIST

Compared Models

- **minisom**: standard SOM (from minisom module[2]).
- **kerasom**: our implementation of SOM in Keras (equivalent to DESOM without AE) and trained by SGD.
- **AE+minisom**: minisom fit on the encoded dataset using an autoencoder with the same architecture as in DESOM.
- **AE+kerasom**: the same approach but with our kerasom model (equivalent to DESOM without joint optimization of AE and SOM).
- **DESOM**: our proposed Deep Embedded SOM with joint representation learning and self-organization.

[2] `https://github.com/JustGlowing/minisom`

Clustering Performance

Method	MNIST		Fashion-MNIST		REUTERS-10k	
	pur	nmi	pur	nmi	pur	nmi
k-means ($k = 64$)	0.842	0.571	0.716	0.512	0.892	0.427
minisom (8×8)	0.637	0.430	0.646	0.494	0.690	0.230
kerasom (8×8)	0.826	0.565	0.717	0.512	0.697	0.324
AE+minisom (8×8)	0.871	0.616	0.734	0.531	0.690	0.235
AE+kerasom (8×8)	**0.939**	**0.661**	**0.764**	**0.539**	0.777	0.306
SOM-VAE (8×8)	0.868	0.595	0.739	0.520	-	-
DESOM (8×8)	**0.939**	0.657	0.752	**0.538**	**0.849**	**0.381**

Table 2: Purity and NMI (average on 10 runs). Best result and results with no significant difference (p-value > 0.05) in bold.

Method	MNIST	Fashion-MNIST	REUTERS-10k
k-means (k = #classes)	58.34	56.45	59.37
AE+kerasom (8×8) + km	**76.06**	44.87	36.61
DESOM (8×8) + km	**76.11**	**56.02**	**57.18**

Table 3: Unsupervised clustering accuracy (%) (average on 10 runs).

Future Work

Future Work

- Further investigate the influence of hyperparameters and training procedure (parameter γ, learning rate, neighborhood decay, number of iterations)
- Study the properties of the latent space learned by DESOM
- Try different map topologies

Thank you for your attention!

Questions?

Experiments

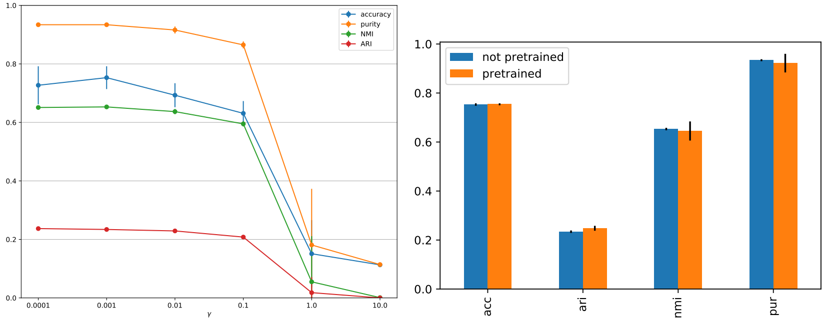

Figure 7: Influence of hyperparameter γ and AE pre-training on several performance metrics

Fard, M. M., Thonet, T., and Gaussier, E. (2018).
Deep k-Means: Jointly Clustering with k-Means and Learning Representations.

Forest, F., Lebbah, M., Azzag, H., and Lacaille, J. (2019).
Deep Embedded SOM: Joint Representation Learning and Self-Organization.
In *ESANN 2019*, pages 1–6, Bruges.

Fortuin, V., Hüser, M., Locatello, F., Strathmann, H., and Rätsch, G. (2018).
Deep Self-Organization: Interpretable Discrete Representation Learning on Time Series.

Guo, X., Gao, L., Liu, X., and Yin, J. (2017).
Improved deep embedded clustering with local structure preservation.
In *IJCAI*, pages 1753–1759.

Harchaoui, W., Mattei, P.-A., Alamansa, A., and Bouveyron, C. (2018).
Wasserstein Adversarial Mixture Clustering.

Jiang, Z., Zheng, Y., Tan, H., Tang, B., and Zhou, H. (2017).
Variational Deep Embedding : An Unsupervised and Generative Approach to Clustering.
In *IJCAI*, pages 1965–1972.

Kingma, D. P. and Ba, J. L. (2015).
Adam: A Method For Stochastic Optimization.
In *ICLR*.

Kohonen, T. (1982).
Self-organized formation of topologically correct feature maps.
Biological Cybernetics, 43(1):59–69.

Pesteie, M., Abolmaesumi, P., and Rohling, R. (2018).
Deep Neural Maps.
pages 1–4.

Song, C., Huang, Y., Liu, F., Wang, Z., and Wang, L. (2014).
Deep auto-encoder based clustering.
Intelligent Data Analysis, 18(6):S65–S76.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017).
Neural Discrete Representation Learning.
In *NIPS*.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010).
Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion.
Journal of Machine Learning Research, 11:3371–3408.

Xie, J., Girshick, R., and Farhadi, A. (2015).
Unsupervised Deep Embedding for Clustering Analysis.
In *ICML*, volume 48.

Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. (2016).
Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering.
In *ICML*.